3DRing: Enabling Low-Cost 3D Hand Position Tracking by Fusing Inertial and Low-Framerate Optical Sensing

Zhuojun Li
Department of Computer Science and Technology, BNRist
Tsinghua University
Beijing, China
lizj23@mails.tsinghua.edu.cn

Lubin Wang
Weiyang College
Tsinghua University
Beijing, China
wanglb23@mails.tsinghua.edu.cn

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

Chang Liu
Department of Computer Science and Technology
Tsinghua University
BeiJing, China
c-liu21@tsinghua.org.cn

Mingyuan Du
Department of Computer Science and Technology
Tsinghua University
Beijing, China
dmy23@mails.tsinghua.edu.cn

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

Yuanchun Shi*‡
Department of Computer Science and Technology
Tsinghua University
Beijing, China
shiyc@tsinghua.edu.cn

Abstract

Current mobile hand tracking systems primarily rely on high-framerate (HFR) optical sensors to capture hand positions, resulting in high computational cost and limiting the applicability in end devices. We propose 3DRing, a 3D hand position tracking method that requires only low-framerate (LFR, <10 FPS) optical data and a single IMU ring. It consists of two stages: (1) a Deep Extended Kalman Filter module that predicts high-framerate hand positions from LFR optical measurements and a single IMU; (2) a Reinforcement Learning module that adaptively selects minimal keyframes for calibration, further reducing the average optical framerate. Using only 6.61 FPS optical data, 3DRing achieves an average real-time tracking error of 1.75 cm and an interaction efficiency of 86.0% in a 3D target selection task, compared to the 67 FPS hand tracking system of Meta Quest Pro, demonstrating a strong potential to reduce the reliance on optical data in mobile hand tracking tasks.

CCS Concepts

• **Human-centered computing** → **Interaction techniques**.

Keywords

Hand Tracking, Low-Cost, Inertial Sensing, Optical Sensing

ACM Reference Format:
Zhuojun Li, Lubin Wang, Chun Yu, Chang Liu, Mingyuan Du, Weinan Shi, and Yuanchun Shi. 2026. 3DRing: Enabling Low-Cost 3D Hand Position Tracking by Fusing Inertial and Low-Framerate Optical Sensing. 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/10.1145/3772318.3791028

1 Introduction

Human hand tracking is a fundamental task in computer vision and human-computer interaction. It enables natural and efficient spatial interactions in the 3D space, such as target selection, object manipulation, and gaming. Current hand tracking methods heavily depend on camera-based optical data (e.g., RGB, infrared, and depth data), which require high framerate (≥60 FPS) for both image capture and processing. While these methods can achieve accurate hand tracking, they come with significant computational cost, which notably limits their use in mobile devices. For instance, Meta Quest Pro [43] and Apple Vision Pro [26] use 5 and 8 cameras respectively to capture images. The hand tracking framerates reaches 72 FPS and 90 FPS. The resulting battery life is no longer than 2 hours under general use. However, many 3D interactions do not require full degrees of freedom (DoF) or high-accuracy hand tracking, especially for lightweight AR devices that even do not support hand tracking currently. Therefore, developing a low-cost hand tracking method that does not rely heavily on optical sensing, while still maintaining sufficient interaction capabilities, is crucial for mobile spatial interactions.

*Also with Key Laboratory of Pervasive Computing, Ministry of Education.
†Corresponding author.
‡Also with Qinghai University.

CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/26/04
https://doi.org/10.1145/3772318.3791028

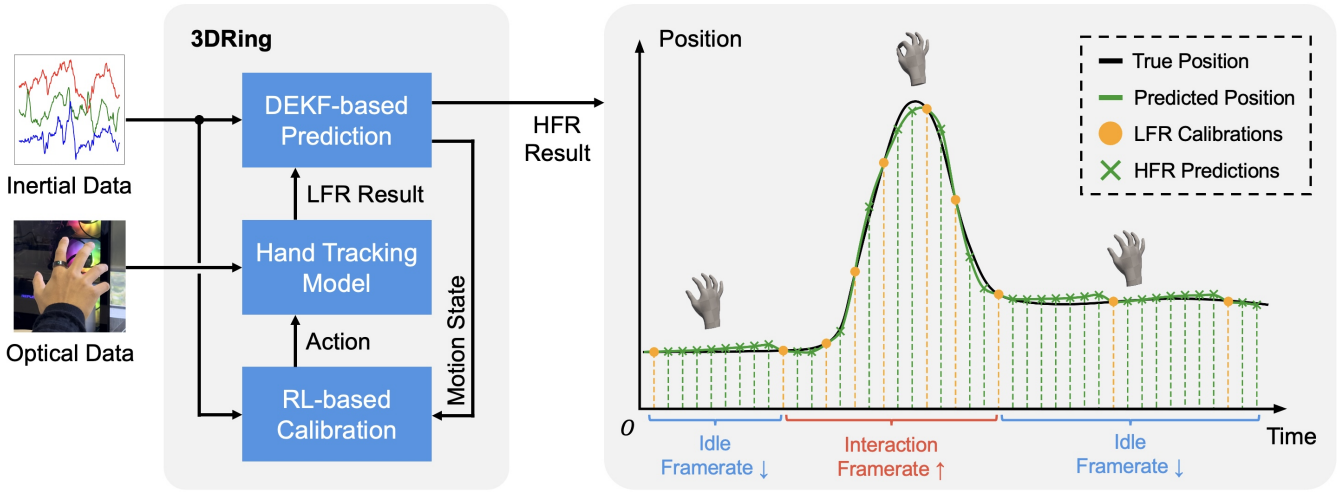


Figure 1: The introduction of 3DRing. Our method aims to lower the optical sensing framerate in 3D hand position tracking as much as possible to reduce the computation costs. To compensate for the information loss, we fuse the optical data with a low-cost IMU ring, which provides rich hand motion information between optical measurements. 3DRing incorporates a DEKF model to predict the hand motion from low-framerate optical data and high-framerate IMU input, and an RL model to adaptively select minimal optical keyframes to calibrate the prediction results (left, detailed later). The tracking curve (right) illustrates the adaptive framerate strategy and the prediction results during real use, where the user moves the hand to select a target in the 3D space. Our method is able to increase the optical sensing framerate during interaction, and reduce it when the hand is static, thus achieving a balance between tracking accuracy and reliance on optical data.

To address the limitations of optical sensing, we focus on reducing its framerate, which lowers the computational costs. If appropriate hardware is available to support variable-framerate sampling, the hardware power consumption can also be reduced, which is further discussed in Section 6.2. This approach, known as low framerate (LFR, <10 FPS) tracking in literature [29, 38, 39, 70], aims to achieve high framerate (HFR, ≥30 FPS) tracking results using only LFR optical data. However, purely relying on LFR optical data face the challenge of balancing tracking accuracy and latency, as significant information is lost and the optical input becomes discontinuous. As a result, these methods are generally applicable only in scenarios where the tracked object moves slowly or predictably, such as tracking pedestrians [39] or vehicles [29]. In contrast, human 3D interactions involve fast and highly unpredictable hand movements, making LFR tracking much more challenging in this context.

To fill this gap, we propose fusing inertial and optical sensing to achieve low computational cost and high accuracy simultaneously, and also reduce the latency caused by LFR optical sensing. The IMU measures accurate hand acceleration and angular velocity and typically samples at a higher frame rate (≥100 FPS). Therefore, it supports hand motion prediction even before the next keyframe is captured, reducing the latency between LFR optical frames. Additionally, the computational cost of the IMU is negligible compared to optical sensors (Section 4.5). Thus, we believe the IMU is an excellent complement to optical sensors, which enables accurate hand tracking while reducing the camera framerate and computational cost. This approach can either enable hand tracking on lightweight AR devices, or serve as a fallback mode that reduces computational overhead on devices with existing hand-tracking capabilities.

We propose 3DRing, a method for 3D hand position tracking that combines a single IMU ring with LFR optical data. We aim to achieve an optimal balance between tracking accuracy and optical sensing framerate, thereby reducing the computational costs. To address information loss, we utilize a 6-axis IMU ring (measuring acceleration and angular velocity), ensuring that hand motions are continuously perceived, thus maintaining high tracking accuracy. The IMU ring is worn on the user's index finger, and its design also enables hand gesture recognition in 3D space (e.g., touch [17], pinch [37], cursor control [52]). While gesture recognition is independent of the hand position tracking presented in this paper, it can be seamlessly integrated to enhance the 3D interaction experience. The input to our method consists of LFR hand position and rotation data (6 DoF, <10 FPS, derived from optical data) and 6-axis IMU data (200 FPS). The output is the high-framerate, continuous hand position tracking results, as illustrated in Figure 1.

3DRing consists of two stages. The first is the **Prediction Stage**, where we introduce a novel method that combines deep learning-based hand velocity prediction with an Extended Kalman Filter (EKF), referred to as DEKF in this paper. This stage predicts continuous hand motion based on LFR hand position and rotation measurements, along with the IMU input. Given that optical hand tracking results are sparse and unreliable for hand velocity estimation, we first train an RNN model to predict hand velocity using only IMU data. We then apply the EKF model to regress hand position and rotation from the predicted hand velocity and angular velocity from the gyroscope. Experiments demonstrate that our

DEKF method significantly outperforms existing EKF-based approaches, reducing the average tracking error from 2.63 ± 0.25 cm by 36.5%, to 1.67 ± 0.12 cm ($p < 0.001$), using an average of just 6.26 FPS optical data.

The second stage is the **Calibration Stage**, where we introduce a new reinforcement learning (RL) task that trains the model to adaptively select minimal optical keyframes to calibrate hand motion predictions. During interaction tasks, users' hand movements are not uniformly distributed, which means the optical tracking framerate can be dynamically adjusted based on the hand motion state. Leveraging this approach, we further reduce the tracking error to 1.39 ± 0.10 cm, which is 47.1% lower than the EKF method and 16.8% lower than the DEKF model ($p < 0.001$). Additionally, the neural networks in the DEKF and RL modules are lightweight, only containing 636.68 K parameters in total and requiring 253.77 M FLOPs of computation per second. As a result, our method is able to reduce the computational cost to about 11% (Section 4.5) compared to CV-based hand tracking methods. Furthermore, a user evaluation study demonstrates that 3DRing achieves 86.0% of the interaction efficiency in a 3D target selection task using only 6.61 FPS (9.9%) optical data, compared to the 67 FPS Meta Quest Pro. It highlights the potential of 3DRing to reduce the computational cost of hand tracking while maintaining high tracking accuracy and interaction efficiency, thereby making hand tracking more accessible to mobile scenarios.

Our contributions in this work are three-fold:

(1) We propose 3DRing, a novel 3D hand position tracking framework that combines inertial and low-framerate optical sensing. It only requires a single IMU ring and LFR optical data, achieving high tracking accuracy and interaction efficiency only with low computational cost.
(2) We introduce a Deep Extended Kalman Filter (DEKF) method to predict continuous hand position from LFR optical frames and IMU input. This method utilizes an RNN-based model to provide reliable hand velocity estimates, which complement the sparse optical data.
(3) We present an RL-based adaptive framerate strategy to calibrate hand motion estimation. This approach adjusts hand tracking based on users' hand movements and interaction states, further enhancing tracking performance.

2 Related Work

2.1 Human Hand Tracking and Reconstruction

Human hand tracking has been extensively studied in computer vision and graphics, supporting a wide range of applications such as human-computer interaction, augmented and virtual reality, and rendering [3, 23, 32, 57]. Existing works primarily focus on detecting hand key points and reconstructing the hand skeleton structure from optical images. In 2D hand images, current methods use convolutional neural networks (CNN) to extract image features, based on which hand joint key points are detected, as 2D coordinates in the image space [15, 33, 54, 69]. The most representative work is MediaPipe Hands [69], which leverages a palm detector and a hand landmark model to detect 21 hand key points, and has been widely used in industry and various applications. In the 3D space, MANO [48] proposes a new hand model that maps hand poses to hand meshes. Based on these 3D hand representations, many works reconstruct 3D hand poses from 2D images [9, 16, 20, 46, 60, 72] or IMUs [8, 25, 35, 44, 67, 68], and have iterated to a good tracking performance. 3D hand mesh and texture models are also proposed to further improve the rendering effect in graphics and animations [5, 14, 34].

The aforementioned works are able to achieve high hand tracking accuracy and fine-grained hand pose reconstruction, which is crucial in graphics and animations. However, in the perspective of human-computer interaction, to support general interactions (e.g., target selection, interface manipulation) in the 3D space, we normally only need the overall 3D hand positions in the global coordinates and gesture events that are related to users' interaction purposes. Furthermore, hand poses from monocular images lack absolute depth information, which is essential for 3D interactions. Therefore, we only use the global 3D hand position tracking [43] in our work, which does not require extensive optical data and can still achieve efficient 3D interactions.

2.2 Low-Cost Object Tracking

CV-based object tracking require extensive optical data to achieve real-time and continuous tracking. The high information density of images leads to significant data processing and computational costs, restricting their applications in mobile computing scenarios. To address this issue, researchers have proposed various low-cost tracking methods. One category of them is to design special tracking systems to reduce the hardware power consumption [1, 41, 47, 64], but they are not general and require specific hardware support. Another category optimizes the tracking algorithms to reduce the computational cost, such as using optical flow [10], feature flow [74], and efficient neural networks [24, 30, 49, 56]. Besides, a more general approach is to use low-framerate (LFR, <10 FPS) optical data, and then recover the high-framerate (HFR) tracking results from LFR data, such that the sensing and computing cost can both be reduced at the same time [6, 28, 29, 38, 39, 70]. However, this approach faces the challenge of balancing the tracking accuracy and latency. By interpolating intermediate tracking results from sparse optical keyframes, the current state estimation can only be updated after the next keyframe comes, such that the minimum tracking latency is limited to $1/FPS$ seconds. By extrapolating (predicting) the tracking results from past states can avoid this issue. However, the tracking accuracy is then compromised, for the reason that the tracking results are only predicted from historical data, which cannot reflect the real-time hand motion status.

To cope with this dilemma, we use inertial sensing to predict the hand positions between optical frames, thus achieving both high tracking accuracy and latency using LFR data. Since the IMU is sensitive to hand motions, it can provide accurate and fast hand position predictions before the next optical frame, and its drift can also be corrected by LFR frames. Therefore, our method achieves a better balance between tracking accuracy and latency, while significantly reducing the reliance on optical data and its power cost.

2.3 Optical and Inertial Sensing Fusion

Optical and inertial sensing are often fused in many tracking and localization applications, due to their complementary sensing capabilities and inevitable shortages [73]. Optical sensors (e.g., RGB, infrared, depth cameras) are able to capture global information such as object positions and orientations, but they are easily affected by lighting conditions and occlusions and have high power consumption. Inertial sensors (e.g., accelerometers, gyroscopes), on the other hand, can provide local motion measurements with extremely low energy costs, but they are sensitive to noises and accumulated errors. Therefore, fusing optical and inertial sensing has the potential to achieve both robust and accurate tracking performance, while maintaining low power consumption. It has been widely applied in various research areas, such as navigation [31, 59, 71], localization [12, 66, 73], and object tracking [13, 21, 41, 53, 55, 58].

In the context of human hand tracking, the most similar work to ours is HOOV [55]. HOOV uses a wrist-worn IMU sensor to predict the hand positions given an initial motion state from optical tracking, allowing users to interact outside the camera's field of view (FOV) for a short period (within 3 seconds), which expands the interactive space of VR devices. Our work, instead, aims to reduce the camera framerate using an IMU sensor. We have two major differences from HOOV: (1) HOOV is only useful when users' hands are out of the tracking FOV. Instead, our method can reduce the camera framerate at all times, which is more general and can be applied to various tracking scenarios. (2) HOOV uses high-framerate (72 FPS) optical data, which gives both accurate position and velocity estimations at the initial tracking state. Our method only relies on LFR (around 6 FPS) optical data. Therefore, the velocity estimations are much more unreliable for predicting arbitrary hand motions. We then designed an RNN-based velocity predictor to tackle this issue, which is technically more challenging and has not been explored in previous works.

3 Method

In this section, we first introduce the problem definition in Section 3.1. Then, we will detail the two key innovations of our work: (1) the prediction stage using DEKF in Section 3.2, and (2) the calibration stage using RL in Section 3.3.

3.1 Problem Definition

Our research goal is to use a single low-cost IMU sensor to complement the optical data in 3D hand tracking tasks, such that the system's reliance on optical sensing (in terms of framerate) and the corresponding computational cost are minimized, without sacrificing much tracking accuracy and interaction efficiency.

To achieve this goal, we propose a low-cost hand tracking framework that is illustrated in Figure 2. Our framework runs on any hand tracking model that estimates the hand position $\boldsymbol{p} \in \mathbb{R}^3$ and rotation $\boldsymbol{r} \in \mathbb{R}^{3\times3}$ from a single frame of raw optical observations $\boldsymbol{o}$ (RGB, infrared, depth images, etc.), which has been widely explored in the literature (Section 2.1). The input to our method is the sequences of IMU data, including accelerations $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_T), \boldsymbol{a}_i \in \mathbb{R}^3$ and angular velocities $\Omega = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_T), \boldsymbol{\omega}_i \in \mathbb{R}^3$ in the device local coordinate system, and raw optical observations $\boldsymbol{O} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T)$. We assume the optical data is synchronized with the IMU data for better clarity, otherwise we can align them by data resampling. The output is the estimated hand overall position $\hat{\boldsymbol{P}} = (\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_T)$ and rotation $\hat{\boldsymbol{R}} = (\hat{\boldsymbol{r}}_1, \ldots, \hat{\boldsymbol{r}}_T)$, while the hidden ground truth is $\boldsymbol{P} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_T)$ and $\boldsymbol{R} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_T)$. Our goal is to adaptively select the optical keyframes to achieve a balance between the average tracking framerate and the tracking accuracy. Therefore, the optimization goal can be formulated as:

$$\begin{aligned} \arg\min_{\boldsymbol{\Phi}, I} \; & ||\boldsymbol{\Phi}(\boldsymbol{A}, \Omega, (\boldsymbol{o}_{i_1}, \ldots, \boldsymbol{o}_{i_N})) - GT||^2 + \lambda \cdot |I| \\ & I = \{i_1, \ldots, i_N\} \subseteq \{1, \ldots, T\} \end{aligned} \tag{1}$$

where I is the keyframe subset with size N. $\boldsymbol{\Phi}$ is the hand tracking model that maps IMU and optical input to the tracking results $\hat{\boldsymbol{P}}$ and $\hat{\boldsymbol{R}}$, and GT is the ground truth tracking results $\boldsymbol{P}$ and $\boldsymbol{R}$. The first term represents the tracking error, and the second term represents the optical sensing cost. λ is a hyperparameter that balances these two terms.

3.2 Prediction: Deep Extended Kalman Filter

3.2.1 Design. The DEKF module regresses the hand position $\hat{\boldsymbol{p}} \in \mathbb{R}^3$ and rotation $\hat{\boldsymbol{r}} \in \mathbb{R}^{3\times3}$ from the IMU input $\boldsymbol{a} \in \mathbb{R}^3$ and $\boldsymbol{\omega} \in \mathbb{R}^3$, under the correction of LFR hand position $\boldsymbol{p}$ and rotation $\boldsymbol{r}$ measurements. In a traditional hand tracking task, the inertial and optical inputs are both captured at a high framerate. The EKF model regresses the hand motion state from IMU data by performing a double integration on accelerations, a single integration on angular velocities, and the transformation from the IMU local coordinate system to the world global coordinate system. Therefore, it needs to maintain the hand velocity as an internal system state for acceleration integration.

However, in our case, the framerate of hand position and rotation measurements (around 6 FPS) is much lower than the IMU data (200 FPS), which makes the hand velocity estimation much more challenging for arbitrary hand motions. To address this issue, we introduce the hand global velocity prediction as an intermediate task and adopt an RNN module to handle it. This design has two advantages: (1) Compared with direct velocity prediction from LFR position measurements, the RNN can provide much more accurate velocity estimation from IMU data, which benefits the integration process in the EKF model. (2) The EKF module will not interact with accelerations anymore, which simplifies the EKF dynamic model to only one integration on the velocity and one integration on the angular velocity. Since the RNN module directly outputs the global hand velocity, the corresponding coordinate transformation is also avoided. As a result, our DEKF design leads to much less error accumulation and better accuracy.

3.2.2 RNN-based Velocity Prediction. We incorporate an RNN model into the traditional EKF model to provide accurate hand velocity estimation under LFR optical measurements. The RNN input is local acceleration $\boldsymbol{a} \in \mathbb{R}^3$, local angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$, and the predicted global rotation $\hat{\boldsymbol{r}} \in \mathbb{R}^{3\times3}$, 15 dimensions in total. For the velocity estimation, the RNN model is naturally suitable for capturing the temporal dependencies in the IMU data and integrating the acceleration to velocity. For the local-to-global coordinate transformation, we add the angular velocity and the predicted hand rotation to the input to help the RNN model better extract the coordinate

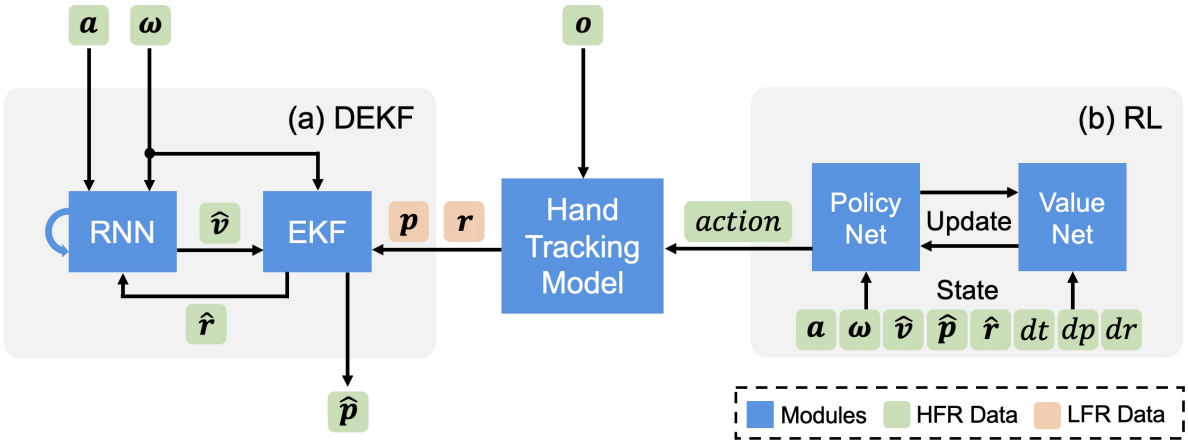


Figure 2: Method overview. Our framework runs on any hand tracking model that estimates hand overall position p and rotation r from raw optical observations o. It contains two stages: (a) DEKF-based prediction stage and (b) RL-based calibration stage. The prediction stage uses an RNN and an EKF model to jointly regress hand position $\hat{p}$ and rotation $\hat{r}$ from the IMU input, including acceleration a and angular velocity ω. The prediction results are dynamically corrected by the LFR position p and rotation r measurements. The RNN module is designed for regressing the hand velocity $\hat{v}$ as an intermediate task, which improves the prediction accuracy under LFR measurements. The calibration stage further uses an actor-critic RL model to assess the necessity to calibrate the hand tracking results using the optical measurements. It adaptively selects minimal keyframes to calibrate the prediction results based on the system state, achieving a better balance between the tracking accuracy and the system's reliance on optical data.

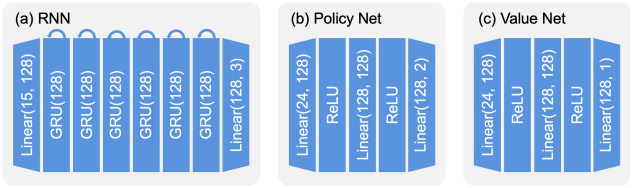


Figure 3: The the structure of neural networks in our method, including (a) the RNN velocity prediction model, (b)&(c) the policy net and value net in the RL algorithm.

features. The output is the global hand velocity $\hat{v} \in \mathbb{R}^3$. With this design, we successfully improved the hand tracking accuracy as well as simplified the EKF dynamic model.

The RNN model structure is shown in Figure 3a. It is designed to be efficient and lightweight. We first use a linear layer to expand the input $(a, \omega, \hat{r}) \in \mathbb{R}^{15}$ to 128 dimensions, then use a 6-layer unidirectional GRU network [7] to process the hidden features. A final linear layer then outputs the global hand velocity. This Linear-GRU-Linear structure is super lightweight and runs in real-time. It only contains 597K parameters and is suitable for mobile devices.

3.2.3 Extended Kalman Filter. Based on velocity prediction, the EKF module is adopted to regress the final hand position and rotation. The input to the EKF module is the LFR optical measurement, including position $p \in \mathbb{R}^3$ and $r \in \mathbb{R}^{3\times3}$, the predicted global hand velocity $\hat{v}$, and angular velocity ω. The output is the estimated HFR hand position $\hat{p}$ and rotation $\hat{r}$. Generally, given the intial state p and r at time n, we can leverage $\hat{v} \in \mathbb{R}^3$ and $\omega \in \mathbb{R}^3$ to predict the successive hand position and rotation at time $n+1$. The core of EKF is the system's dynamic model:

$$\begin{aligned} p_{n+1} &= p_n + \hat{v}_n \Delta t && (Position\ Prediction) \\ r_{n+1} &= matrix(r_n \omega_n \Delta t) r_n && (Rotation\ Prediction) \end{aligned} \tag{2}$$

where $matrix$ is the transformation from rotation vector to rotation matrix.

Based on this derivation, the EKF method is introduced to dynamically model the uncertainties of the measurements and the dynamic model, and leverage the Kalman gain to balance these uncertainties, thus giving a more robust and accurate tracking result. We followed the EKF design from [11] and modified it to fit our problem. The full process can be divided into prediction, update, and the calculation of the Kalman gain, which dynamically adjusts the weight of the prediction and measurement results. Unlike the traditional EKF model that predicts and updates jointly at every time step, the LFR measurements p and r in our system are significantly sparse compared with $\hat{v}$ and ω. Therefore, we only use the prediction process at every time step, but calibrate the system state using the update process only when the RL module decides to use an LFR measurement. Therefore, the hand tracking model is able to run at a low framerate, which significantly reduces the reliance on optical data. Since EKF is a well-studied algorithm, we provide a detailed mathematical formulation in the Appendix A. Readers can also refer to [11] for more details.

3.3 Calibration: RL-based Adaptive Framerate

3.3.1 Design. Based on the DEKF module, our method can already run on any LFR optical data with a fixed framerate. However, a

fixed framerate may not be an optimal choice, since it lacks the awareness of users' hand motion status. For example, intuitively, when the user's hand is moving in a fast and random pattern during interaction, the hand tracking error may be significantly larger than when the hand is moving in a slow and smooth pattern. Therefore, we need an adaptive framerate strategy that selects the minimal keyframes to achieve maximum tracking performance.

We formulated a new reinforcement learning (RL) task to achieve this goal. Instead of using a fixed optical framerate, the RL model needs to estimate the tracking error from the current hand motion state, and decide whether to use an optical measurement to calibrate the hand tracking results at each time step. Therefore, the RL problem can be formulated into a binary classification task. At each time step, the RL model outputs a probability of calibration. Then, a boolean calibration action is sampled from this probability distribution and passed to the DEKF module. Only when the action is positive, an optical frame is processed to get a ground truth hand motion measurement, and the DEKF module updates the hand position and rotation using this measurement. Leveraging this RL strategy, our method could adaptively select the minimal optical keyframes that are most beneficial for improving the hand tracking accuracy, thus reducing the sensing and computational cost drastically.

For the RL algorithm, we adopt the standard Proximal Policy Optimization (PPO) algorithm [51] with clipped surrogate objective. The PPO algorithm and training procedure follows [2], which readers are referred to for more details. For the RL model, we use an MLP actor-critic model, as illustrated in Figure 3b&c. The model consists of two parts: (1) the policy net, which outputs the probability of whether to calibrate at each time step or not, and (2) the value net, which estimates the value of the current state. These two models only contain 40K parameters in total, which is lightweight and efficient. To complete the PPO training process, we further define the system state input to these models and the corresponding reward of taking an action. We will then detail these two parts in the following sections.

3.3.2 State Representation. In RL problems, the input to the RL model is the system state vector. The RL model then decides the action to take, and the system will transit to the next state based on this action. In our task, the system state should represent the hand motion status, such that the RL model can decide whether to calibrate the hand tracking results or not. To achieve this goal, we formulate the system state as:

$$\boldsymbol{s} = (\boldsymbol{a}, \boldsymbol{\omega}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{p}}, \hat{\boldsymbol{r}}, dt, dp, dr) \in \mathbb{R}^{24} \tag{3}$$

where $\boldsymbol{a}, \boldsymbol{\omega}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{p}}, \hat{\boldsymbol{r}}$ are the same as in the DEKF module (Section 3.2), representing the hand motion status. Beyond that, we add three more dimensions of temporal features dt, dp, dr to the state vector. dt is the elapsed time since the last calibration (i.e., when RL sampls a positive action). Similarly, dp and dr are the RMS hand position and rotation changes between the last calibration and the current predicted value. Intuitively, when the optical calibration pauses for a long time, or when the hand motion changes significantly, the tracking error may be prone to accumulate, leading to a higher probability of calibration.

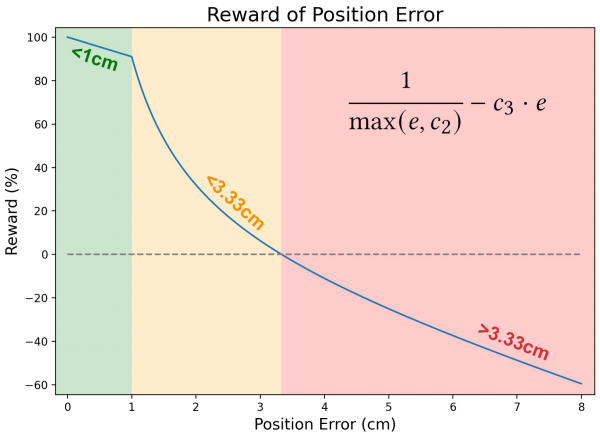


Figure 4: The position error reward. When the position error e<1.0cm, the reward reaches almost 100%. Then, the reward decreases drastically to zero when e reaches 3.33cm, and continues to decrease (almost linearly) when e>3.33cm.

3.3.3 Reward Function. The RL reward is the key to determine the quality of an action, thus guiding the RL model to converge to an optimal policy. In our task, the reward design should be able to balance the tracking error and the optical tracking cost. Therefore, when the tracking error is small, or when the calibration rate is low, the RL model should receive a high reward. However, the tracking error and the calibration rate are two contradictory factors, which is a typical trade-off problem. They should be properly weighted in the reward function to achieve an optimal balance.

We design the RL reward function as follows. At each time step, the RL model samples a binary action $a \in \{0, 1\}$ (1 for calibration, 0 for not), and the DEKF module will update a new hand position $\hat{\boldsymbol{p}}$ with or without the optical measurement. Then, the corresponding tracking error $e = \sqrt{||\hat{\boldsymbol{p}} - \boldsymbol{p}||^2}$ will be calculated (we only consider the position error here). We define the reward at each time step as a function of e and a as:

$$reward(e, a) = c_1 \cdot \left(\frac{1}{\max(e, c_2)} - c_3 \cdot e\right) - c_4 \cdot a \tag{4}$$

The reward is divided into two terms: the *position error reward* and the *calibration reward*, with weights c_1 and c_4 respectively. For the position error reward, it is generally designed to be inversely proportional to the tracking error, with two extra modifications. First, we add a clipping term c_2 to avoid the reward explosion when the tracking error is small. Otherwise, the RL strategy may be prone to calibrate frequently to reach a high reward, which is not desired for our LFR tracking goal. Second, we add a penalty term c_3 to penalize it when the tracking error is large, such that the RL strategy will restrict the tracking error within an acceptable range. When e is represented in cm, we empirically set $c_2 = 1.0$ and $c_3 = 0.09$ in practice, such that the reward reaches almost 100% when $e < 1.0cm$, and decreases to zero when e reaches $3.33cm$. It is illustrated in Figure 4.

For the calibration reward, using an optical measurement means a cost in sensing and computation. Therefore, we add a negative

reward for each calibration to balance the cost. In practice, c_4 is fixed to 1.0, and c_1 is set to 1.2×10^{-3} empirically, which achieves an average calibration rate of 6 FPS. Note that we tuned our model to converge to the 6 FPS calibration rate only to demonstrate its capability. For different requirements on the tracking error and calibration rate, it could be adjusted by using a different c_1 value.

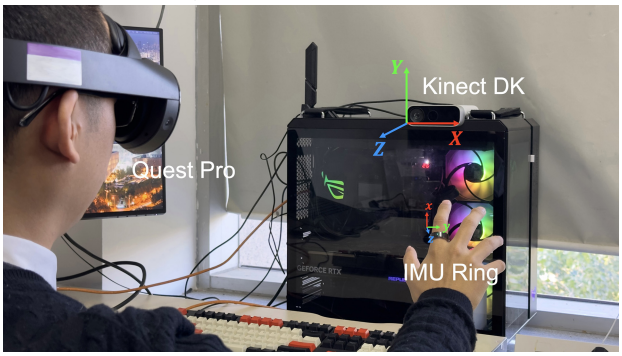
4 Study One: Data Collection

In this section, we conducted a user study to collect the dataset for model training. We introduce the apparatus setup in Section 4.1, the study design and procedure in Section 4.2, and the model training details in Section 4.3. Then, we conducted an offline evaluation of our method in Section 4.4, and compare the computation cost of our method with previous CV-based hand tracking method in Section 4.5.

4.1 Apparatus

(a) The IMU ring.

(b) The Quest Pro headset.



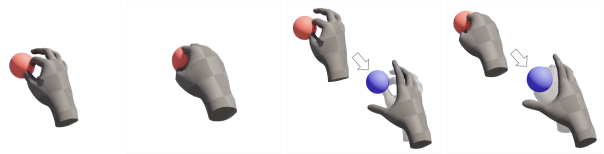
(c) The apparatus setup.

Figure 5: The apparatus setup in the user study. The user wears an IMU ring on the index finger and a Meta Quest Pro on the head. The Meta Quest Pro and the IMU ring capture the optical and inertial data respectively. We also add a Kinect DK camera to record the study process and use it as a marker of the world coordinate frame. All optical data are captured and converted to the world coordinate frame XYZ, while the IMU data are in the local coordinate frame xyz.

To train the DEKF and RL model (Section 3), we aim to build an inertial and optical dataset of human hands during 3D interaction. For the inertial part, we embedded an InvenSense ICM-42607-C IMU sensor [27] into a customized ring device. It is worn on the index finger of the user's dominant hand, for the reason that the index finger triggers most of the hand gestures (e.g., pinch) in 3D interaction. Thus, it has become a common practice in ring-based interaction tasks [18, 36, 37, 52]. The IMU sensor is located on the palm side of the ring. It samples 6-axis acceleration and angular velocity data at 200 FPS. The orientation of the IMU ring is fixed on the finger to ensure consistent data collection across users, and no calibration is required before use. The IMU ring and sensor coordinates are shown in Figure 5a.

For the optical part, we directly used the Meta Quest Pro headset [43] (Figure 5b) to recognize and record the hand skeleton data (positions and rotations of 24 joints) in the 3D space. We choose Meta Quest Pro for two reasons: (1) It is a widely used commercial VR headset with state-of-the-art hand tracking capability, which ensures the data quality and user experience in our study. (2) It directly provides the hand skeleton data in the 3D space, which is universal across most CV-based hand traking models. Therefore, the DEKF and RL modules in our method will not directly interact with the raw optical images, which ensures the transferability of our method to other optical hand tracking systems. We use the root joint of the index finger to represent the overall hand position $\boldsymbol{p} \in \mathbb{R}^3$ and rotation $\boldsymbol{r} \in \mathbb{R}^{3\times3}$ in the DEKF model (Figure 2). The Quest data are sampled at 72 FPS and upsampled to 200 FPS to be synchronized with the IMU data. In our experiments, we use the calibration actions from the RL module to select the Quest keyframes (other frames are ignored), thus simulating the LFR hand tracking results. In real deployment, the tracking results can be obtained from LFR or event-based cameras, as discussed in Section 3.3.1. The full apparatus setup is shown in Figure 5c.

4.2 Design and Procedure



(a) Pinch. (b) Clench (c) PinchMove. (d) ClenchMove.

Figure 6: Hand gestures in the data collection study. (a) Pinch at the red ball's location. (b) Clench at the red ball's location. (c) Pinch at the red ball's location, move to the blue ball and release. (d) Clench at the red ball's location, move to the blue ball and release. We use them to simulate the target selection, moving, and dragging operations in 3D interaction.

In this study, we aim to collect users' hand movements in general 3D interactions. To achieve this, we designed a target selection task in the 3D space, implemented by Unreal Engine 5 [4] and rendered in the Meta Quest Pro headset. The task incorporats both hand gestures and hand movements. We do not only collect the hand movement data for the reason that hand gestures are naturally combined with hand movements in 3D interaction. A hand gesture may cause extra noises to the IMU ring, but not disturb users' intended hand movements. Therefore, our model must be robust to the gesture noises for recovering the true hand movements.

We designed five gestures in total: *Pinch*, *Clench*, *PinchMove*, *ClenchMove*, (shown in Figure 6), and *Free* (i.e., any arbitrary gesture performed at the user's discretion). For each gesture, each user performs three sessions of data collection trials. In each session, a small red ball would appear at a random location (in a cuboid of 80cm width, 50cm height, and 50cm depth) in the VR space in front of the user. Each time, the user is required to move the dominant hand to the target ball's position and perform the corresponding gesture. After that, the ball will randomly refresh to the next random position, and the user needs to move their hand and complete the gesture again, and so on. Differently, for the *Pinch Move* and *Clench Move* gestures, a red ball and a blue ball would appear simultaneously in front of the user at each time. The user is required to first move the hand to the red ball, perform and maintain the corresponding gesture, and then move to the blue ball and release the gesture, which simulates a task involving target selection and dragging. This process repeats for 30 times per session.

We recruited 20 participants from the campus (aged 20 to 35, M = 25.10, SD = 4.25, 11 females). They have never experienced 3DRing before. During the study, we set up a Kinect DK camera to record the study process to check potential errors (Figure 5c, the camera data are not used in our method). It also serves as a marker of the origin of the world coordinate system. Before the experiment, we manually annotated the center of the Kinect DK camera and the global Z direction to determine the world coordinate system XYZ in the VR space. Then, we explained the above task design to the participants and arranged a warm-up session with the *Pinch* gesture for them to get familiar with the task procedure. After that, each participant performed 15 sessions (5 gestures $\times$ 3 sessions per gesture) of the data collection task. The gesture order is randomized for each participant. We recorded the IMU data from the ring, the hand motions from the Quest headset. As a result, we collected $20 \times 5 \times 3 \times 30 = 9000$ times (5.45 hours) of 3D target selection trials and the corresponding hand motion data in total. Each participant spent about 50 minutes and was paid 15 USD.

4.3 Model Training

4.3.1 Data Preprocessing. After the data collection study, we have 300 data sequences (20 participants $\times$ 5 gestures $\times$ 3 sessions) in total, each containing 30 times of target selection trials. Since the Quest and IMU data are sampled at different framerates (72 FPS and 200 FPS) and on different devices, we first upsampled the Quest data to 200 FPS to be consistent with the IMU data, and converted the Quest data to the world coordinate frame XYZ using matrix transformation. Then, to strictly align them in the time dimension, we first synthesized the virtual angular velocity data by differentiating the Quest hand rotations and matching them with the real IMU angular velocities. After that, we derived the final hand position $\boldsymbol{p}$, rotation $\boldsymbol{r}$, and IMU data $\boldsymbol{a}, \boldsymbol{w}$ in 200 FPS for model training. These data are further cut into sequences with a length of 1000 frames (5 seconds) for batch operations.

4.3.2 DEKF Model Training. As introduced before (Section 3.2), the DEKF module in 3DRing contains two parts: an RNN model to regress hand velocity from the IMU input and the predicted hand rotation, and an EKF model to predict the hand position and rotation based on the system dynamic model (Equation 3.2.3). They jointly depend on each other to predict the hand motion between LFR tracking frames. The EKF model is pure mathematical and does not require any training, while the RNN model is trained on the dataset from the user study (Section 4.2), detailed as follows.

For the RNN input, we directly used the IMU acceleration $\boldsymbol{a}$, angular velocity $\boldsymbol{\omega}$, and the ground truth hand rotation $\boldsymbol{r}$. We use $\boldsymbol{r}$ to replace the predicted hand rotation $\hat{\boldsymbol{r}}$ for the reason that (1) the prediction accuracy of $\hat{\boldsymbol{r}}$ depends on the measurement framerate of $\boldsymbol{r}$, which is not determined without the RL module, and (2) the rotation prediction $\hat{\boldsymbol{r}}$ is relatively reliable from the LFR rotation measurement and the following angular velocity integration, thus being close to the ground truth $\boldsymbol{r}$. For the RNN output, we synthesized the ground truth hand velocity $\boldsymbol{v}$ by taking differentials of the ground truth hand position $\boldsymbol{p}$ (collected at 72 FPS and upsampled to 200 FPS). We further used a low-pass Butterworth filter with a cutting frequency of 8 Hz to smooth the velocity data, as most of the natural hand movements are below this frequency. It also filters out high-frequency velocity fluctuations caused by hand gestures (e.g., pinch), ensuring that the training objective of the velocity estimation model focuses only on the overall hand motion rather than being affected by high-frequency finger movements. The IMU data are normalized to mean = 0 and standard deviation = 1 before input to the RNN model, but the velocity data are not normalized and are represented in m/s.

To train the RNN model, we used 16 users as the training dataset and 4 users as the validation dataset. We trained the RNN model with 4000 epochs, a batch size of 32, and a learning rate starting at 10^{-4}, and decaying with a factor of 0.998 for each epoch. We used the Adam optimizer and MSE loss and evaluated the model every 10 epochs on the validation dataset. We saved the model with the best validation loss for further evaluations. The full training process takes around 30 minutes on an NVIDIA RTX 4090 GPU.

4.3.3 RL Model Training. Given the state representation in Equation 3.3.2, the goal of the RL module is to decide whether the system should trigger a calibration action (i.e., processing an optical frame to generate a hand position and rotation measurement) or not. Different from other RL tasks, our RL environment is defined on the users' hand movement data and the trained DEKF model. We define an hand movement data clip (containing 1000 frames of $\boldsymbol{a}$, $\boldsymbol{\omega}$, $\boldsymbol{p}$, and $\boldsymbol{r}$) as an episode. For each episode, we simulate the hand movement process, let the RL agent to decide whether to trigger the calibration action, and use the trained DEKF model to predict the hand movements. For each step, we recorded the calibration action, the system state (Equation 3.3.2), and the reward (Equation 3.3.3) as the interaction history. After the episode ended, we calculated the accumulated future rewards and used the history data to update the policy and value networks, as specified in the PPO algorithm [51]. As a result, the RL agent is converged to maximize the rewards, which achieves a balance between calibration frequency and hand tracking accuracy.

To train the RL agent, we used the same dataset as the DEKF model (Section 4.3.2). The weights of the position error reward c_1 and the calibration reward c_4 is set to 1.2×10^{-3} and 1.0 respectively (Equation 3.3.3), resulting in a calibration framerate of around 6 FPS. We trained the model with 40 epochs and a batch size of 50. We use 4 concurrent CPU processes to sample a batch of episode

history, based on which we update the policy and value networks with 50 iterations per batch. The learning rate is 5×10^{-5}, and decays with a factor of 0.9 for each epoch. We evaluated the model for every epoch on the validation dataset and saved the model with the highest reward for further evaluations. The full training process takes around 3.5 hours on an Intel i9-14900K CPU and an NVIDIA RTX 4090 GPU.

4.4 Offline Evaluation

4.4.1 Settings. In this study, we conducted an offline evaluation of our method based on the dataset we collected. We design four baselines, including two vision-only methods and two ablation methods. They are defined as:

(1) **Baseline1 (linear extrapolation, vision-only):** only using LFR optical data. It predicts future hand positions from the last two LFR tracking frames by linear extrapolation.
(2) **Baseline2 (quadratic extrapolation, vision-only):** similar to Baseline2, it predicts future hand positions from the last three LFR tracking frames by quadratic extrapolation. These two baselines represent vision-only forward prediction methods without IMU data.
(3) **Baseline3 (3DRing removing RNN and RL, optical-inertial fusion):** fusing inertial and optical data, but only containing an EKF model, with a fixed optical sensing framerate and no RNN-based hand velocity prediction. In this setting, the hand velocity is added to the EKF state. And, the velocity prediction is replaced by $\hat{\boldsymbol{v}}_{\boldsymbol{n}} = \hat{\boldsymbol{v}}_{\boldsymbol{n-1}} + (\boldsymbol{r}_n\boldsymbol{a} - \boldsymbol{g})\Delta t$ ($\boldsymbol{g}$ is the gravity), which is also added to the system dynamics in Equation 3.2.3. This setting represents the current EKF-based method (e.g., the EKF model in HOOV[55]).
(4) **Baseline4 (3DRing removing RL, optical-inertial fusion):** fusing inertial and optical data, containing the full DEKF module, but with a fixed calibration framerate (i.e., no RL module).

To our best knowledge, there is no existing work that involves temporal prediction in LFR hand tracking tasks. Therefore, we design the first two baselines which only use LFR optical data to predict future hand positions. To minimize latency in realtime use, we use forward extrapolation but not interpolation to predict hand positions before the next LFR tracking frame arrives (the same as 3dRing), otherwise the latency would increase at least the interval between two LFR frames (e.g., 167 ms at 6 FPS).

For optical-inertial fusion methods, most temporal prediction approaches leverage the EKF model to smooth the tracking result, like in HOOV [55], which is comparable to Baseline3. For our task, the biggest challenge is that LFR tracking looses too much information on hand velocities, which makes it hard to determine the initial velocity value for integrating the acceleration data. Therefore, we introduce the DEKF method to complement the velocity information with learning-based method (RNN), leading to Baseline4. Baseline3 and Baseline4 also serve as two ablation settings to evaluate the effectiveness of the DEKF and RL modules.

4.4.2 Evaluation Method. Based on the dataset we collected in Section 4, we simulated the real-time hand tracking process with the baselines and our method, and compared the tracking result with the ground truth. To evaluate the tracking accuracy, we measured the following metrics: the **Mean Error**, **Median Error**, and **95% Error**, representing the average, median, and 95% percentile error of the Euclidean distance between the predicted hand position and the ground truth. Since the tracking accuracy is correlated with the calibration framerate, we evaluate the baselines under 15 different framerates within 2-10 FPS, forming a curve of the tracking error over the calibration framerate (shown in Figure 7). But for our method, since the calibration action is decided by the RL model, which would converge to a particular average framerate during training, we can only get a pair of a tracking error and a framerate from one model. Therefore, we compare 3DRing with the baselines by interpolating the error curve linearly, and find the baseline tracking error under the same framerate as 3DRing to make the comparison fair.

To get a more reliable result, we adopted a cross-user validation method. We split the dataset of 20 users P1-P20 into 20 groups, from (P1, P2, P3, P4), (P2, P3, P4, P5), …, to (P20, P1, P2, P3), denoted as group 1-20, each containing 4 users. For each group, we used the 4 users as the validation dataset and the rest as the training dataset. Therefore, we trained 20 DEKF models and 20 RL models in total. We do not split a test dataset for the reason that our method will be further tested in a real-time user study (Section 5).

4.4.3 Result. We visualize the tracking errors over framerates of the baselines and 3DRing in Figure 7. After training, our RL models (of 20 groups) reach an average calibration framerate of 6.26 ± 0.28. For each group, we find the corresponding baseline tracking errors under the same framerate as 3DRing using linear interpolation. As a result, the mean errors of four baselines and 3DRing are 2.57 ± 0.27 cm, 3.41 ± 0.41 cm, 2.63 ± 0.27 cm, 1.67 ± 0.12 cm, and 1.39 ± 0.10 cm respectively. The median errors of them are 1.28 ± 0.12 cm, 1.72 ± 0.18 cm, 1.71 ± 0.17 cm, 1.11 ± 0.08 cm, and 0.98 ± 0.06 cm respectively. The 95% percentile errors of them are 9.52 ± 1.11 cm, 12.45 ± 1.67 cm, 8.33 ± 0.92 cm, 5.06 ± 0.40 cm, and 4.03 ± 0.33 cm respectively. We further conducted a one-way RM ANOVA test on these five methods. Results show a significant difference among four baselines and 3DRing for all metrics ($p < 0.001$). Post-hoc pair-wise t-tests with Bonferroni correction also prove that for all metrics, 3DRing all significantly outperforms the four baselines ($p < 0.001$), and Baseline4 also significantly outperforms Baseline3 ($p < 0.001$).

From these results, we draw two major conclusions: (1) the vision-only methods (Baseline1 and Baseline2) both fail in this task, indicating that the hand movement is highly dynamic and unpredictable thus proving the task difficulty. Without relying on IMU data, hand movements are hard to be accurately predicted from sparse LFR tracking results. (2) the DEKF and RL modules both contribute significantly to our method (from Baseline3 and Baseline4). The DEKF significantly improves the tracking accuracy compared with current EKF models for better velocity prediction, and the RL module further improves the performance by adaptively selecting the keyframes for calibration.

4.5 Computational Cost

The main research goal of 3DRing is to reduce the computational cost of the hand tracking system in AR/VR. Therefore, we need to quantify the reduction of computational cost when 3DRing is

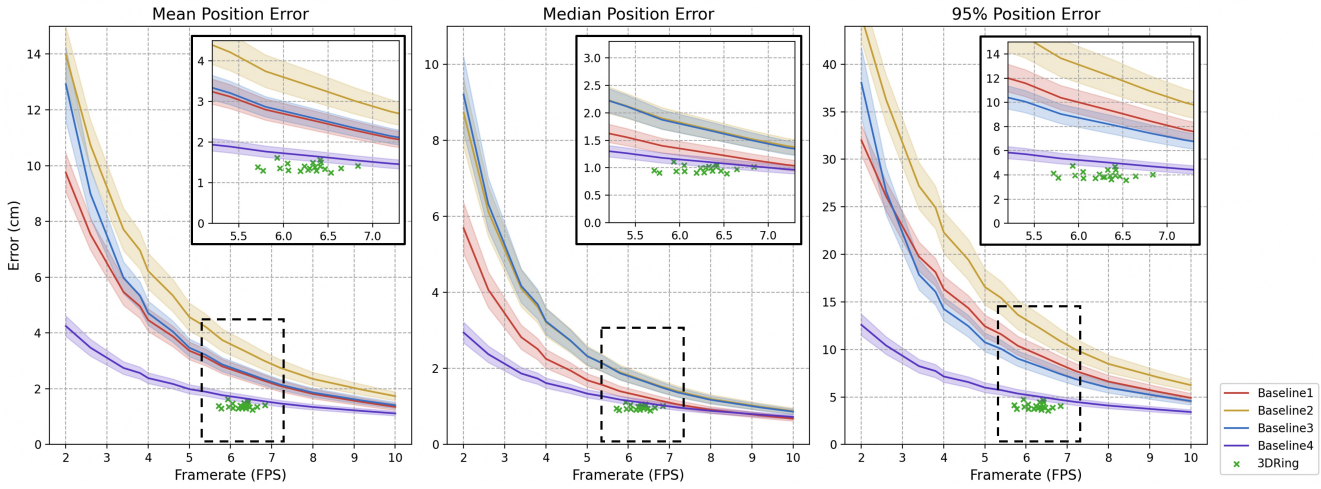


Figure 7: The mean, median, and 95% position error of baselines and 3DRing. We use 20 groups of cross-user datasets to train and validate these methods. The baseline methods are evaluated under 15 calibration framerates from 2 to 10 FPS. The red, yellow, blue, and purple curves show the average error and the standard deviation across the 20 groups, while the green Xs mark the error-framerate pairs of the 3DRing full models. Results show that our method outperfoms all baselines.

Method	Params	FLOPS w/o 3DRing	FLOPS w/ 3DRing	Proportion
Mediapipe[69]	3.78 M	133.75 G	14.21 G	10.62%
MEgATrack[19]	2.28 M	39.74 G	4.40 G	11.07%
UmeTrack[20]	4.25 M	235.84 G	24.86 G	10.54%
MinimalHand[72]	12.97 M	1655.35 G	172.96 G	10.45%
HandGraphCNN[16]	21.77 M	1956.50 G	204.37 G	10.45%
POEM[63]	35.43 M	12.51 T	1.31 T	10.44%

Table 1: The computational cost of existting CV-based hand tracking methods, and the cost when they are combined with our method. We calculated the total parameters of these models, the FLOPs per second (FLOPS) when they are running at 60 FPS, and the FLOPS when they are running at 6.26 FPS and combined with 3DRing (200 FPS). The proportion shows the ratio of the FLOPS with 3DRing to the original FLOPS.

applied to existing hand tracking systems. Since our method can be applied to any CV-based hand tracking methods (explained in Section 3.1), we calculated the computational cost (measured in FLOPS, i.e., floating-point operations per second) of several state-of-the-art CV-based hand tracking methods. These methods include Mediapipe Hands [69], MEgATrack [19], UmeTrack [20], MinimalHand [72], HandGraphCNN [16], and POEM [63], representing recent progress in CV-based hand tracking.

We firstly investigated the computational cost of our method. As introduced in Section 3, our method contains two neural networks: the RNN-based velocity prediction model in the DEKF module and the policy and value networks in the RL module. These three models contain 636.68 K parameters in total, and are all running at 200 FPS, which is the same as the IMU sampling rate. Therefore, given the network structure, the working FPS, and the input data size, we are able to calculate that the total FLOPS of 3DRing is only 253.77 M, which is significantly more efficient than hand tracking itself, mainly due to the sparsity of IMU data compared with optical data.

Then, we reimplemented the hand tracking models in the aforementioned methods, and calculated the model parameters and the FLOPS without 3DRing respectively, which are show in Table 1. To ensure a smooth hand tracking result, we assume that these methods are all running at 60 FPS (the frequencies of Meta Quest Pro and Apple Vision Pro are 72 FPS and 90 FPS respectively). Additionally, when these methods are combined with 3DRing, their framerate can be drastically reduced with the complement of the IMU data. We assume that the calibration rate of these methods combined with 3DRing is also 6.26 FPS (Section 4.4.3), and further calculated the FLOPS combined with 3DRing together. The results show that the main computational cost in our framework comes from the hand tracking part. Our method can be viewed as a strategy to schedule the hand tracking model more properly leveraging the IMU information. As a result, the computational cost can be reduced to around 11% compared with the original cost.

Last but not least, we need to point out that the Meta Quest Pro used in our experiment is a closed-source platform. We are unable to directly calculate that hand tracking FLOPS of Meta Quest Pro.

Therefore, we choose to use the existing open-source hand tracking methods to estimate the cost of the hand tracking part. Table 1 shows that the reduction of computational cost of our method is universal when combined with these methods. Thus, we conclude that 3DRing has a good potential to reduce the computational cost of general hand tracking systems, especially for AR/VR applications.

5 Study Two: Evaluation

In this section, we conduct an online evaluation on 3DRing to investigate its real-time performance and user experience. We introduce the study design and procedure in Section 5.1, and analyze the study results in Section 5.2.

5.1 Design and Procedure

The general hand interactions in the 3D space can be divided into two parts: hand gestures and hand movements. For hand gestures, numerous previous work has studied hand gesture recognition using an IMU sensor [17, 18, 37, 45, 52, 62, 65]. It can be well supported on the IMU ring, which is already low-cost and does not rely on optical data. Therefore, in this study, we focus on the evaluation of the hand movement tracking performance of 3DRing.

To achieve this, we designed a 3D target selection task in the VR space, as illustrated in Figure 8. We implemented a VR application that reads the IMU data and Quest hand data, then predicts the hand's overall position and rotation in real-time using our proposed method and the same apparatus as in Section 4. The Quest hand tracking frames are used as the DEFK measurements only when the RL model decides to do so, thus simulating the LFR optical tracking. The tracking result is rendered as a green ball with $r = 3cm$ in the VR space, representing the hand's overall position. The green ball follows the hand movement in real-time. Additionally, a grey spherical area with $r = 5cm$ will appear at a random position in front of the user (in a cuboid of $80cm$ width, $50cm$ height, and $50cm$ depth, the same as the data collection study in Section 4.2) as the target area. The goal of the target selection task is to move the green ball into the target area as fast as possible (overlap not being counted). Therefore, the user's hand movement has only a $5cm - 3cm = 2cm$ tolerance space, representing 3D spatial interaction tasks with a high precision requirement. Once the green ball is moved into the target area, the target area will refresh to the next random position, and the user need to repeat the above procedure and so on.

To compare our method with existing hand tracking methods, we defined three systems in the study:

(1) **Full Quest Pro (67 FPS):** the built-in hand tracking system of Meta Quest Pro. We directly read the hand tracking results from Quest Pro, and upsample it to 200 FPS to align with our method (only for convenience). Then, we use the root joint of the index finger to represent the overall hand position, and render it in the VR space for every 3 frames, (limited by the screen freshrate), forming a 67 FPS hand tracking system.
(2) **3DRing (adaptive framerate):** the proposed method. The average optical sensing framerate (around 6.25 FPS) depends on the RL calibration strategy. The position prediction results are rendered at 67 FPS as well.
(3) **LFR Quest Pro (6.25 FPS):** the down-sampled hand-tracking system of Meta Quest Pro. Similar to the full Quest Pro system, we render the hand position for every 32 frames. Therefore, the tracking framerate is 6.25 FPS, which is similar to our method, but without any position prediction.

We recruited 20 participants from the campus (aged 18 to 30, M = 22.30, SD = 3.25, 10 males and 10 females). They have neither experienced 3DRing before nor joined the data collection study. We explained the task design to the participants and required them to use the 3 hand tracking systems to complete 9 sessions (3 for each system) of the target selection task. The 3 systems was arranged in a rotating order (i.e., 123, 231, 312, 123, ...) to counterbalance the learning effect. Additionally, the system design was not disclosed to the participants to avoid bias. For each system, we also arranged a warm-up session to let participants get familiar with the system. For each session, the target area would refresh at 30 random positions one by one, and the users are required to complete the task as fast as possible. Therefore, each user has to complete 12 sessions (3 systems $\times$ (1 warm-up session + 3 formal sessions)) in total. We recorded all tracking results of the formal sessions for further analysis. After the study, the participants are invited to fill in a questionnaire to provide subjective feedback on the 3 systems. Each participant was paid 15 USD for their time.

5.2 Result

5.2.1 Calibration Framerate. The average calibration framerate of 3DRing running on the validation dataset is 6.26 FPS (Section 4.4.3). Therefore, we set the tracking framerate of LFR Quest Pro to 6.25 FPS (rendering one time every 32 frames) to make it comparable with 3DRing. In the evaluation, the average, minimum, and maximum framerates during real-time use turned out to be 6.61 ± 0.32 FPS, 2.75 ± 0.24 FPS, and 9.68 ± 0.49 FPS respectively. The average framerate is slightly higher than before, for the reason that users are required to complete the task as fast as possible in the evaluation, but not in the data collection study. Therefore, for faster hand movements, the RL model is prone to increase the calibration framerate for better tracking accuracy. However, it is still only 9.9% of the Full Meta Quest Pro (67 FPS), and is only higher than the LFR Quest Pro (6.25 FPS) by 5.8%, which is still considered fair for comparison.

5.2.2 Interaction Efficiency. The interaction efficiency is measured by the time cost of the 3D target selection task. We recorded the total time cost of each interaction session and divided it by the number of trials (30) to get the average time cost. As shown in Figure 9, the average target selection time costs of the Full Quest Pro, 3DRing, and the LFR Quest Pro are 1.65 ± 0.24 s, 1.91 ± 0.37 s, and 2.31 ± 0.50 s respectively. A One-way RM ANOVA and post-hoc test with Bonferroni correction (the same for all following statistical tests, if not specified) shows that both the Full Quest Pro and the LFR Quest Pro have significant differences with 3DRing ($p < 0.001$). The remaining gap to the Full Quest Pro is largely caused by the IMU drifting problem, as also discussed in IMU-based motion capture works [67, 68]. To conclude, using only 9.9% of optical frames, our method achieves 86.4% of the interaction efficiency of the Meta Quest Pro, and is 21.0% faster than the LFR Quest Pro with a similar

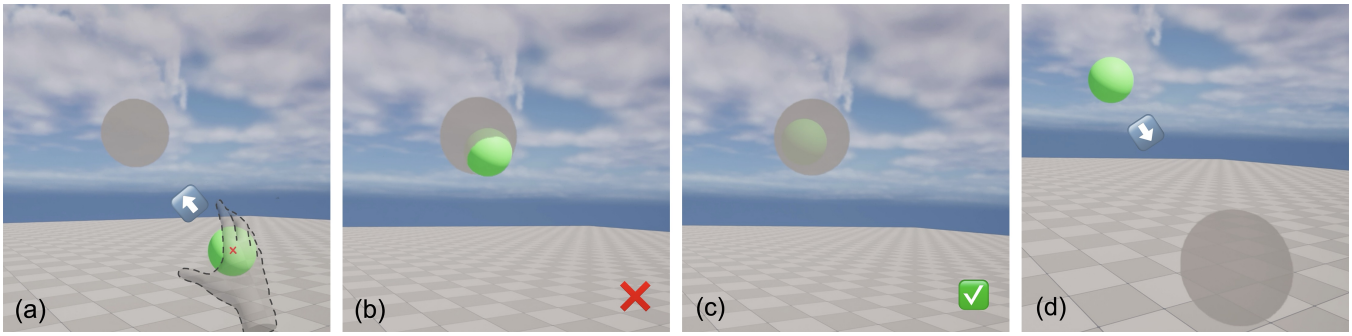


Figure 8: The target selection task in the evaluation study. (a) We use a green ball with r=3cm to represent the hand's overall position. The green ball is centered at the root of the index finger. The task objective is to move the green ball into a grey spherical area with r=5cm as fast as possible. (b) An unsuccessful trial. The green ball overlaps with the target area but is not inside it. (c) A successful trial. The green ball is moved into the target area. (d) Once a target selection is completed, the target area will move to a new random location, indicating the start of the next trial. The hand mesh and the arrow markers are invisible to users in the actual experiment.

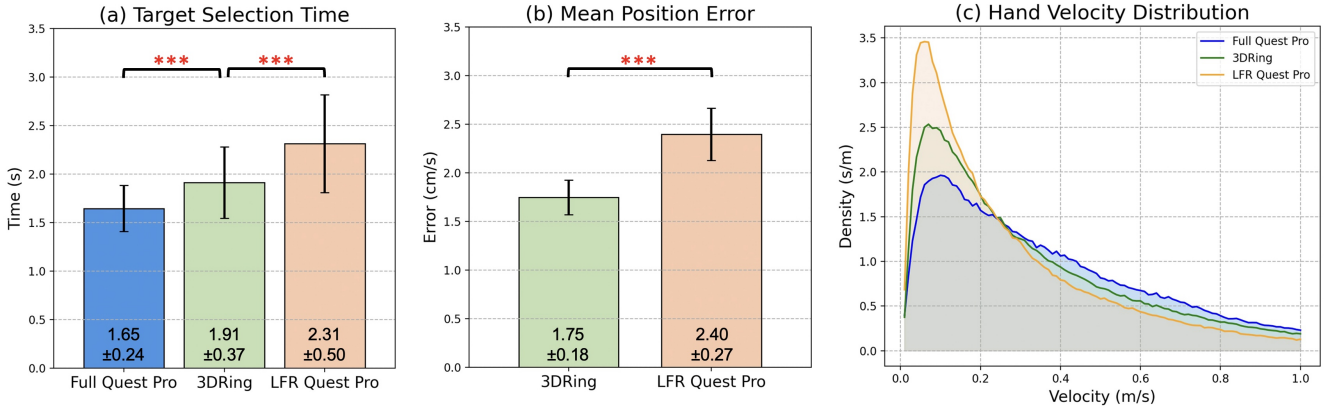


Figure 9: The quantitative results of study two. (a) The target selection time cost of the three systems. * indicates a significance level of p<0.001 (same for the following statistical tests). (b) The real-time tracking error of 3DRing and the LFR Quest Pro. The tracking results from the Full Quest Pro are used as the ground truth. (c) The distributions of users' hand velocity when using the three systems. The area below the distribution curve is normalized to 1. The velocity range is truncated to [0.0, 1.0] m/s for better illustration.**

framerate. It demonstrates a strong potential of 3DRing to be applied in low-cost hand tracking devices.

5.2.3 Tracking Error. We further measured the real-time tracking error of our method. We use the hand tracking results from Full Quest Pro as the ground truth, whose position error is considered zero. The position errors of 3DRing and LFR Quest Pro are 1.75±0.18 cm and 2.40 ± 0.27 cm respectively. It reduced by 27.1% with our method, showing a significant difference ($p < 0.001$). Therefore, we prove that fusing inertial sensing with optical sensing is able to improve the tracking accuracy, even only using LFR optical data.

5.2.4 Velocity Distribution. To find a plausible explanation for the differences of the tracking performance, we visualized users' hand velocity distributions using them in Figure 9c. The average hand velocity using Full Quest Pro, 3DRing, and LFR Quest Pro are 42.76± 5.96 cm/s, 37.83 ± 3.88 cm/s, and 30.46 ± 3.59 cm/s respectively, and the differences between each two of them are all significant ($p < 0.001$). The distribution shows that users tend to move their hands more slowly when the tracking framerate is lower, because the tracking error and latency may cause more uncertainty and cautious in the user's perception, thus reducing their hand movement speed. Our method leverages the IMU to predict hand positions between optical frames, which alleviates the uncertainty problem. Therefore, users are able to move their hands faster during interaction, thus achieving a higher efficiency.

5.2.5 Subjective Feedback. After the evaluation, we collected subjective feedback from the participants. They are required to rate the 3 systems in 7 different aspects using a 7-point Likert scale (higher is better). The aspects include:

(1) **Speed:** the interaction efficiency when doing the target selection task. Higher is better.

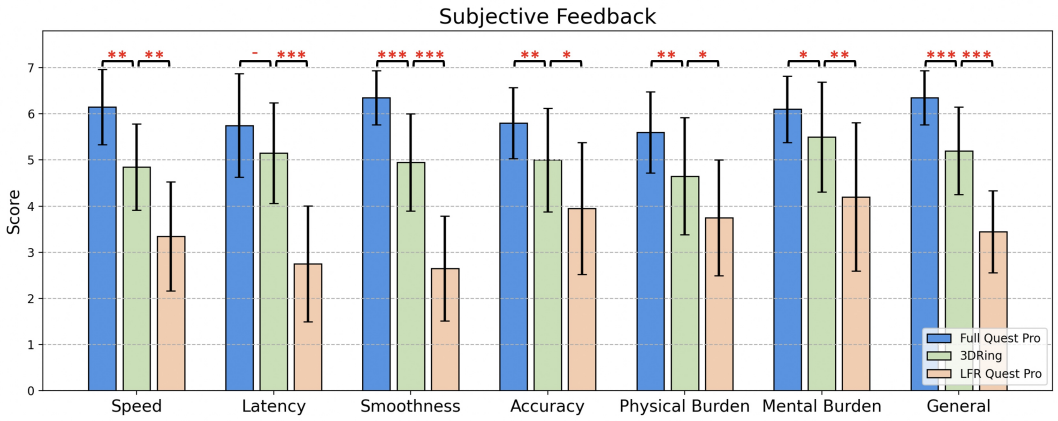


Figure 10: Users' subjective feedback of the 3 systems in terms of speed, latency, smoothness, accuracy, physical burden, mental burden, and general satisfaction. The score is rated on a 7-point Likert scale (higher is better). The significance level is marked as * for p<0.05, ** for p<0.01, and * for p<0.001. '-' indicates no significant difference.**

(2) **Latency:** the tracking latency when moving the hand. Lower is better.
(3) **Smoothness:** the smoothness of the rendered tracking result. Higher is better.
(4) **Accuracy:** the tracking accuracy of the hand position. Higher is better.
(5) **Physical Burden:** the physical effort required to finish the task. Lower is better.
(6) **Mental Burden:** the mental effort required to finish the task. Lower is better.
(7) **General:** the overall satisfaction of the system. Higher is better.

We visualized the feedback results and significance levels (using the Wilcoxon signed-rank test) in Figure 10. Generally, the Full Quest Pro received the highest scores in all aspects, the LFR Quest Pro was the lowest, and 3DRing was in the middle, which is consistent with previous quantitative results. We also observed all significant differences between the 3 systems in all aspects, except for the tracking latency between 3DRing and the Full Quest Pro. It demonstrates an outstanding real-time performance of 3DRing to predict hand positions only from LFR optical data. To conclude, though there is still a gap between 3DRing and the Full Quest Pro, the LFR tracking ability of 3DRing is already able to provide an acceptable user experience in mobile scenarios. Despite only using LFR optical data, 3DRing received positive feedback in all aspects, which indicates users' willingness to use our method in 3D interactions.

5.2.6 Fitts' Law Study. In the previous 3D target selection study, the radii of the cursor (green ball) and the target area (grey sphere) are fixed to 3 cm and 5 cm respectively, resulting in a target selection precision of 2 cm. To further investigate the generalizability of 3DRing to tasks with varying difficulty, we designed a follow-up study to examine its performance under different target selection precisions, and reported the relationship between target selection time and task difficulty, following the design principles of Fitts' Law [40, 42, 52]. Specifically, we varied the radius of the target area to 4 cm, 5 cm, and 7 cm, while keeping the cursor radius fixed at 3 cm. The resulting target selection precisions are changed to 1 cm, 2 cm, and 4 cm. Other settings remain exactly the same as in the main study (Section 5.1).

We recruited 18 participants from the campus (aged 21 to 28, M = 23.06, SD = 1.89, 9 males and 9 females). We explained the task design to them, and required them to use the 3 hand tracking systems (LFR Quest Pro, 3DRing, and Full Quest Pro) to complete 9 sessions. Each system contains 3 sessions with different target selection precisions (1 cm, 2 cm, and 4 cm). The order of the 3 systems was arranged in a rotating order, and the order of the 3 precisions was randomized to counterbalance the learning effect. Other procedures remain the same as before.

To fit the relationship between target selection time and task difficulty, we calculate the Index of Difficulty (ID) of each target selection trial as $ID = \log_2(D/W + 1)$, where D is the distance between the target area and the previous one, W is the target selection precision (i.e., 1cm, 2cm or 4cm). For each session, we calculate the average target selection time and ID over 30 trials, and visualize the results in Figure 11. For precision 1 cm, the average target selection times of LFR Quest Pro, 3DRing, and Full Quest Pro are 3.34 ± 0.93 s, 3.10 ± 0.82 s, and 2.33 ± 0.60 s; for precision 2 cm, they are 2.21 ± 0.40 s, 1.88 ± 0.44 s, and 1.57 ± 0.33 s; for precision 4 cm, they are 1.59 ± 0.35 s, 1.27 ± 0.28 s, and 1.18 ± 0.28 s. Similar to the main study, we found 3DRing shows consistent improvement over LFR Quest Pro under all precisions. Moreover, from Figure 11, we found this improvement tend to be relatively more significant under lower task difficulties (precision 4 cm), but is weaker under higher task difficulties (precision 1 cm). This could be explained by the motion jitter caused by IMU drifts, which may affect target selection with high precision requirements. However, although 3DRing still has room for improvement in high-precision target selection tasks, it substantially outperforms LFR Quest Pro in lower-precision tasks and can even approach the performance of Full Quest Pro. This suggests that 3DRing can be effectively applied to lightweight AR/VR applications for fundamental hand-based interactions such

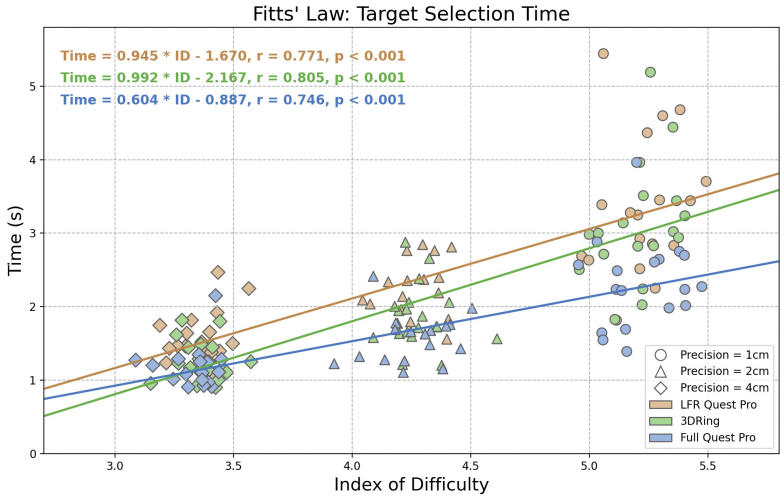


Figure 11: The Fitts' Law results. For each session, we visualized the average target selection time against the Index of Difficulty (ID). The shape of each point indicates the target selection precision, and the color represents the hand-tracking system.

as menu navigation, capabilities that are not yet well supported on lightweight devices.

6 Discussion

6.1 IMU Ring Design

The inertial sensing device in our method is an IMU ring, worn on the index finger. We would like to discuss the rationale behind this design. The general principle is to achieve an optimal balance between simplicity and sensing ability. For simplicity, the ring is the smallest possible IMU wearable device compared with others like smartwatches [55]. Our prototype (Figure 5ca) only weights 4g, minimizing the user burden in daily long-term use. For sensing ability, the IMU ring is the closest device to sense finger motions, which carries rich information in interactions. We demonstrate in this paper that a single IMU ring can enable low-cost hand position tracking when fused with LFR optical sensing. The IMU ring can also support a wide range of tasks, like gesture recognition [37], text input [18], pointing [52], hand writing [22], and other applications [61]. These gesture-based sensing capabilities can all be integrated with our hand position tracking method in parallel, without affecting the usage of 3DRing. Therefore, we envision that the IMU ring will become a widely used input device and sensing platform in future AR/VR systems for spatial interaction. Industrial products like Samsung Galaxy Ring [50] have also proved its potential. In conclusion, the sensing capability and simplicity of the IMU ring are sufficient to calirify its use in our method. Our method further extends its applications to enhance hand tracking.

6.2 Computational Cost

Our method aims to reduce the computational cost of the hand tracking system by lowering the optical sensing framerate. It has several advantages, including but not limited to (1) reducing the power consumption, (2) increasing the battery life, and (3) being more suitable for light-weight devices. We would like to clarify some concerns about them.

First, for the power consumption in hand tracking systems, it could be divided into two parts: the hardware sensing energy cost and the computational energy cost. For the hardware sensing part, a lower framerate means fewer images need to be captured. However, not all camera sensors support capturing images at a variable framerate. In such cases, using a fixed low framerate is more appropriate. Nevertheless, we still envision that adaptive-framerate sensing will be valuable in future low-power hardware. For example, event cameras can naturally achieve event-triggered sensing. For the computational cost, regardless of the hardware, only a small number of frames need to be processed, which greatly reduces the computational load of the system, while unused frames can simply be buffered and discarded. In addition, it is worth noting that for hand-tracking systems whose per-frame tracking results depend on multiple historical frames, our method faces a greater trade-off that such systems may need to forgo historical data and rely on single-frame inference, which can lead to larger tracking errors. Even so, the adaptive framerate strategy can still prioritize processing only the most informative frames to minimize the computational load as much as possible. For the actual power consumption, since our hardware platform Meta Quest Pro is a closed-source system, we are unable to directly measure it. We regard it as a limitation of our work for future work to address, and only use the reduction of computational cost (Section 4.5) to reflect the potential power consumption reduction.

Second, a lower power consumption could directly lead to a longer battery life of mobile devices. However, since our system also contains a lightweight IMU ring, a potential concern is that though the total power consumption is reduced, the IMU ring may become a new bottleneck of the whole system. To clarify this, we further measured the battery life of our IMU ring. The prototype (Figure 5a) is embedded with a 19.5 mAh battery, and its battery life is over 5 hours for continuous use and data transmission, which

is already longer than the current VR headset (e.g., Meta Quest Pro and Apple Vision Pro, 2-3 hours). For real use senarios, users' active interaction is relatively sparse, which means the battery life could be further extended with a proper power management strategy.

Last, a lower computational cost means a lower requirement on the capability of the system hardware. Our method is suitable for more lightweight devices, like AR glasses and other wearable devices. For example, hand tracking has not been widely applied in AR glasses due to the strict weight and energy constraints. Our method could be a potential solution to this problem, thus supporting more emersive and interactive AR applications.

6.3 Method Generalizability

In our work, to validate this low-cost hand position tracking pipeline, we implemented it on the Meta Quest Pro headset and a customized IMU ring, and evaluated its tracking performance through a 3D target selection task. However, our method is not limited to this apparatus setting and task. For the hand tracking part, we only used the overall hand position and rotation (6 DoF) tracking results captured and processed by the Meta Quest Pro. This requirement can also be supported by other hand tracking models that provide similar tracking results [9, 16, 20, 46, 60, 72]. For the interaction task, our current implementation focuses on the overall hand position tracking and validates it using a 3D pointing task. We acknowledge that there is still a gap to general 3D spatial interactions. However, 3D pointing is one of the most fundamental tasks, enabling basic 3D interface operations such as menu navigation, which most lightweight AR devices currently lack. More complex interaction tasks, such as fine-grained object manipulation, mid-air text entry, or gaming, may require richer sensing capabilities, for example, multi-joint positions and rotations. Therefore, we do not claim that our current implementation can support the full spectrum of 3D spatial interaction tasks. Instead, it is better suited to providing lightweight devices with essential pointing capabilities. Our core contribution lies in proposing a LFR hand-tracking framework based on optical-inertial fusion, while leaving the support for more complex tasks to future work. Beyond hand tracking, since we only use the 6 DoF hand position and rotation, this low-cost tracking pipeline can also be adapted to other object tracking tasks.

6.4 Limitations and Future Work

Despite the promising results, our work still has limitations, which provide directions for future work:

(1) **Accuracy:** Our method achieves an average tracking error of 1.39 ± 0.10 cm (6.26 FPS) on the validation dataset, and 1.75 ± 0.18 cm (6.61 FPS) during real-time evaluation. Although the tracking error is acceptable for most 3D interaction tasks, it still has a gap compared to Meta Quest Pro. The interaction efficiency is also lower than the Meta Quest Pro for 14.0%. Future work still has room to improve it under the same tracking framerate.

(2) **The Hand Position:** In our work, the IMU ring is worn on the proximal phalanx of the index finger, and we use the ring's position as a proxy for the overall hand position. Although we apply filtering techniques to mitigate the influence of high-frequency finger movements on estimating the overall hand velocity (Section 4.3.2), the finger position still differs from the true hand position (e.g., the palm or the wrist), which may introduce perceptual discrepancies for users. To remain compatible with existing ring-based interaction capabilities, we chose to retain this design despite the trade-off. Future work could consider inferring the overall hand motion indirectly from the ring to eliminate this discrepancy.

(3) **IMU Ring Orientation:** Our ring does not require calibration before use, but it must be worn on the index finger in a fixed orientation. This is because the velocity estimation model needs to establish a mapping between the IMU signals and the hand velocity. In practical use, this requirement may introduce some inconvenience. Future work may explore ID-oriented designs that naturally guide users to wear the ring in a consistent orientation (e.g., adding a protrusion or an indicator light). In addition, automatic detection of the orientation could be investigated, introducing either a brief calibration procedure or a dynamic orientation recognition algorithm.

(4) **Hardware Latency:** The Quest Pro and IMU ring data are transmitted to a host computer via Quest Link and Bluetooth Low Energy (BLE). The data streams are processed by our model to predict hand positions, which are then sent back to Quest Pro for rendering. The end-to-end latency (measure by a 120 FPS camera capturing rendered images and hand motions simultaneously) is about 120 ms, which is acceptable for most users (Figure 5.2.5). However, it still has room for improvement in future work. Limited by the BLE bandwidth, 10 IMU frames are transmitted at each time, resulting in a 50 ms latency. The BLE transmission latency is measured to be around 30 ms. But the data processing and computation only takes less than 10 ms. Transmission on Quest Link and rendering also takes around 30 ms. Therefore, we conclude that the latency mainly comes from data sampling and transmission, which is limited by the hardware prototype, but not by the method itself. Future work can optimize the hardware (e.g., using on-device processing) to further reduce the overall latency.

(5) **Application Scenarios:** As mentioned in 6.3, we demonstrate the tracking performance of 3DRing using a 3D target selection task. Though target selection is the most basic and fundamental task in 3D interaction, future work can apply it to more scenarios to validate the generalizability more thoroughly.

7 Conclusion

We present 3DRing, a low-cost 3D hand position tracking method that is able to achieve accurate and real-time tracking performance using only LFR optical data with adaptive framerate and a single IMU ring. Our key innovations include the DEKF-based position prediction and the RL-based adaptive framerate calibration. The DEKF model fuses an RNN model to predict the hand velocity, and an EKF model to further regress the hand position, which enables accurate hand motion prediction from LFR optical data. The RL

model further adaptively selects minimal optical keyframes to calibrate the prediction results. It is aware of users' hand motion state and thus achieves a balance between tracking accuracy and framerate. Evaluations show that both these two modules contribute significantly to our method, and drastically reduce the computational cost of the hand tracking system. In a 3D target selection task, 3DRing achieves an average tracking error of 1.75 ± 0.18 cm from 6.61 FPS optical data, which is only 9.9% of the Meta Quest Pro's framerate (67 FPS). The interaction efficiency is also 86.0% of the Meta Quest Pro, demonstrating a strong potential of our method to reduce the reliance on optical data while maintaining a satisfying tracking and interaction performance.

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, Beijing Key Lab of Networked Multimedia, Institute for Artificial Intelligence, Tsinghua University (THUAI), College of AI, Tsinghua University.

References

[1] Lizy Abraham, Andrea Urru, Niccolò Normani, Mariusz P Wilk, Michael Walsh, and Brendan O'Flynn. 2018. Hand tracking and gesture recognition using lensless smart sensors. *Sensors* 18, 9 (2018), 2834.

[2] Joshua Achiam. 2018. Spinning Up in Deep Reinforcement Learning. (2018).

[3] Ammar Ahmad, Cyrille Migniot, and Albert Dipanda. 2019. Hand pose estimation and tracking in real and virtual interaction: A review. *Image and Vision Computing* 89 (2019), 35–49.

[4] Alex Becker. 2024. Kalman Filter from the Ground Up. https://www.kalmanfilter.net/default.aspx

[5] Xingyu Chen, Baoyuan Wang, and Heung-Yeung Shum. 2023. Hand avatar: Free-pose hand animation and rendering from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8683–8693.

[6] Yohan Chon, Elmurod Talipov, Hyojeong Shin, and Hojung Cha. 2011. Mobility prediction-based smartphone energy optimization for everyday location monitoring. In *Proceedings of the 9th ACM conference on embedded networked sensor systems*. 82–95.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[8] Nathan DeVrio, Vimal Mollyn, and Chris Harrison. 2023. Smartposer: Arm pose estimation with a smartphone and smartwatch using uwb and imu data. In *Proceedings of the 36th annual ACM symposium on user interface software and technology*. 1–11.

[9] Haoye Dong, Aviral Chharia, Wenbo Gou, Francisco Vicente Carrasco, and Fernando De la Torre. 2024. Hamba: Single-view 3D Hand Reconstruction with Graph-guided Bi-Scanning Mamba. *arXiv preprint arXiv:2407.09646* (2024).

[10] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 2758–2766.

[11] Inc. Epic Games. 2025. Unreal Engine 5. https://www.unrealengine.com/en-US/unreal-engine-5

[12] Wei Fang, Lianyu Zheng, and Huanjun Deng. 2016. A motion tracking method by combining the IMU and camera in mobile devices. In *2016 10th international conference on sensing technology (ICST)*. IEEE, 1–6.

[13] Wei Fang, Lianyu Zheng, Huanjun Deng, and Hongbo Zhang. 2017. Real-time motion tracking for mobile augmented/virtual reality using adaptive visual-inertial fusion. *Sensors* 17, 5 (2017), 1037.

[14] Daiheng Gao, Yuliang Xiu, Kailin Li, Lixin Yang, Feng Wang, Peng Zhang, Bang Zhang, Cewu Lu, and Ping Tan. 2022. DART: Articulated hand model with diverse accessories and rich textures. *Advances in Neural Information Processing Systems* 35 (2022), 37055–37067.

[15] Srujana Gattupalli, Ashwin Ramesh Babu, James Robert Brady, Fillia Makedon, and Vassilis Athitsos. 2018. Towards deep learning based hand keypoints detection for rapid sequential movements from rgb images. In *Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference*. 31–37.

[16] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 2019. 3d hand shape and pose estimation from a single rgb image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10833–10842.

[17] 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.

[18] Yizheng Gu, Chun Yu, Zhipeng Li, Zhaoheng Li, Xiaoying Wei, and Yuanchun Shi. 2020. Qwertyring: Text entry on physical surfaces using a ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–29.

[19] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. 2020. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics (ToG)* 39, 4 (2020), 87–1.

[20] Shangchen Han, Po-chen Wu, Yubo Zhang, Beibei Liu, Linguang Zhang, Zheng Wang, Weiguang Si, Peizhao Zhang, Yujun Cai, Tomas Hodan, et al. 2022. UmeTrack: Unified multi-view end-to-end hand tracking for VR. In *SIGGRAPH Asia 2022 conference papers*. 1–9.

[21] Changyu He, Peter Kazanzides, Hasan Tutkun Sen, Sungmin Kim, and Yue Liu. 2015. An inertial and optical sensor fusion approach for six degree-of-freedom pose estimation. *Sensors* 15, 7 (2015), 16448–16465.

[22] 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.

[23] Jinuk Heo, Hyelim Choi, Yongseok Lee, Hyunsu Kim, Harim Ji, Hyunreal Park, Youngseon Lee, Cheongkee Jung, Hai-Nguyen Nguyen, and Dongjun Lee. 2024. Hand Tracking: Survey. *International Journal of Control, Automation and Systems* 22, 6 (2024), 1761–1778.

[24] Andrew G Howard. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[25] Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J Black, Otmar Hilliges, and Gerard Pons-Moll. 2018. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15.

[26] Apple Inc. 2024. Apple Vision Pro. https://www.apple.com/apple-vision-pro/

[27] InvenSense. 2024. TDK InvenSense. https://invensense.tdk.com

[28] Dongcheng Jiang, Chao Zhang, and Philip C Woodland. 2021. Variable Frame Rate Acoustic Models Using Minimum Error Reinforcement Learning.. In *Interspeech*. 2601–2605.

[29] Pavel Korshunov and Wei Tsang Ooi. 2010. Reducing frame rate for object tracking. In *International Conference on Multimedia Modeling*. Springer, 454–464.

[30] Hyunhoon Lee, Young-Seok Kim, Mijung Kim, and Youngjoo Lee. 2021. Low-cost network scheduling of 3D-CNN processing for embedded action recognition. *IEEE Access* 9 (2021), 83901–83912.

[31] Yongseok Lee, Jaemin Yoon, Hyunsoo Yang, Changu Kim, and Dongjun Lee. 2016. Camera-GPS-IMU sensor fusion for autonomous flying. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 85–88.

[32] Rui Li, Zhenyu Liu, and Jianrong Tan. 2019. A survey on 3D hand pose estimation: Cameras, methods, and datasets. *Pattern Recognition* 93 (2019), 251–272.

[33] Yuan Li, Xinggang Wang, Wenyu Liu, and Bin Feng. 2019. Pose anchor: A single-stage hand keypoint detection network. *IEEE Transactions on Circuits and Systems for Video Technology* 30, 7 (2019), 2104–2113.

[34] Yuwei Li, Longwen Zhang, Zesong Qiu, Yingwenqi Jiang, Nianyi Li, Yuexin Ma, Yuyao Zhang, Lan Xu, and Jingyi Yu. 2022. Nimble: a non-rigid hand model with bones and muscles. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

[35] Zhuojun Li, Chun Yu, Chen Liang, and Yuanchun Shi. 2024. Poseaugment: Generative human pose data augmentation with physical plausibility for imu-based motion capture. In *European Conference on Computer Vision*. Springer, 55–73.

[36] Chen Liang, Chi Hsia, Chun Yu, Yukang Yan, Yuntao Wang, and Yuanchun Shi. 2023. DRG-Keyboard: Enabling subtle gesture typing on the fingertip with dual IMU rings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–30.

[37] Chen Liang, Chun Yu, Yue Qin, Yuntao Wang, and Yuanchun Shi. 2021. DualRing: Enabling subtle and expressive hand interaction with dual IMU rings. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27.

[38] Haibin Ling, Yi Wu, Erik Blasch, Genshe Chen, Haitao Lang, and Li Bai. 2011. Evaluation of visual tracking in extremely low frame rate wide area motion imagery. In *14th International Conference on Information Fusion*. IEEE, 1–8.

[39] Liqiang Liu and Jianzhong Cao. 2020. End-to-end learning interpolation for object tracking in low frame-rate video. *IET Image Processing* 14, 6 (2020), 1066–1072.

[40] I Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139.

[41] Andualem T Maereg, Emanuele L Secco, Tayachew F Agidew, David Reid, and Atulya K Nagar. 2017. A low-cost, wearable Opto-Inertial 6-DOF hand pose tracking system for VR. *Technologies* 5, 3 (2017), 49.

[42] Hongyu Mao, Mar Gonzalez-Franco, Vrushank Phadnis, Eric J Gonzalez, and Ishan Chatterjee. 2025. RestfulRaycast: Exploring Ergonomic Rigging and Joint Amplification for Precise Hand Ray Selection in XR. In *Proceedings of the 2025 ACM Designing Interactive Systems Conference*. 28–39.

[43] Meta. 2024. Meta Quest Pro. https://www.meta.com/quest/quest-pro/

[44] Vimal Mollyn, Riku Arakawa, Mayank Goel, Chris Harrison, and Karan Ahuja. 2023. Imuposer: Full-body pose estimation using imus in phones, watches, and earbuds. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–12.

[45] Seyed Ahmadreza Mousavi and Rastko Selmic. 2023. Wearable smart rings for multi-finger gesture recognition using supervised learning. *IEEE Transactions on Instrumentation and Measurement* (2023).

[46] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. 2024. Reconstructing hands in 3d with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9826–9836.

[47] Giuseppe Placidi, Alessandro Di Matteo, Filippo Mignosi, Matteo Polsinelli, Matteo Spezialetti, et al. 2022. Compact, Accurate and Low-cost Hand Tracking System based on LEAP Motion Controllers and Raspberry Pi.. In *ICPRAM*. 652–659.

[48] Javier Romero, Dimitrios Tzionas, and Michael J Black. 2022. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610* (2022).

[49] Zaid Saeb Sabri and Zhiyong Li. 2021. Low-cost intelligent surveillance system based on fast CNN. *PeerJ Computer Science* 7 (2021), e402.

[50] SAMSUNG. 2024. Samsung Galaxy Ring. https://www.samsung.com/us/rings/galaxy-ring/

[51] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[52] 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 CHI Conference on Human Factors in Computing Systems*. 1–19.

[53] Soyong Shin, Zhixiong Li, and Eni Halilaj. 2023. Markerless motion tracking with noisy video and imu data. *IEEE Transactions on Biomedical Engineering* 70, 11 (2023), 3082–3092.

[54] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 1145–1153.

[55] Paul Streli, Rayan Armani, Yi Fei Cheng, and Christian Holz. 2023. Hoov: Hand out-of-view tracking for proprioceptive interaction using inertial sensing. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–16.

[56] Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 6105–6114.

[57] Eleni Theodoridou, Luigi Cinque, Filippo Mignosi, Giuseppe Placidi, Matteo Polsinelli, João Manuel RS Tavares, and Matteo Spezialetti. 2022. Hand tracking and gesture recognition by multiple contactless sensors: A survey. *IEEE Transactions on Human-Machine Systems* 53, 1 (2022), 35–43.

[58] Yushuang Tian, Xiaoli Meng, Dapeng Tao, Dongquan Liu, and Chen Feng. 2015. Upper limb motion tracking with the integration of IMU and Kinect. *Neurocomputing* 159 (2015), 207–218.

[59] Ya Tian, Jie Zhang, and Jindong Tan. 2013. Adaptive-frame-rate monocular vision and imu fusion for robust indoor positioning. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2257–2262.

[60] Jiayi Wang, Franziska Mueller, Florian Bernard, Suzanne Sorli, Oleksandr Sotnychenko, Neng Qian, Miguel A Otaduy, Dan Casas, and Christian Theobalt. 2020. Rgb2hands: real-time tracking of 3d hand interactions from monocular rgb video. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 1–16.

[61] Zeyu Wang, Ruotong Yu, Xiangyang Wang, Jiexin Ding, Jiankai Tang, Jun Fang, Zhe He, Zhuojun Li, Tobias Röddiger, Weiye Xu, et al. 2025. Computing with Smart Rings: A Systematic Literature Review. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 9, 3 (2025), 1–54.

[62] Xuhai Xu, Jun Gong, Carolina Brum, Lilian Liang, Bongsoo Suh, Shivam Kumar Gupta, Yash Agarwal, Laurence Lindsey, Runchang Kang, Behrooz Shahsavari, et al. 2022. Enabling hand gesture customization on wrist-worn devices. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–19.

[63] Lixin Yang, Jian Xu, Licheng Zhong, Xinyu Zhan, Zhicheng Wang, Kejian Wu, and Cewu Lu. 2023. POEM: reconstructing hand in a point embedded multi-view stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21108–21117.

[64] Hui-Shyong Yeo, Byung-Gook Lee, and Hyotaek Lim. 2015. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. *Multimedia Tools and Applications* 74 (2015), 2687–2715.

[65] 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.

[66] Xinyu Yi, Yuxiao Zhou, Marc Habermann, Vladislav Golyanik, Shaohua Pan, Christian Theobalt, and Feng Xu. 2023. Egolocate: Real-time motion capture, localization, and mapping with sparse body-mounted sensors. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–17.

[67] Xinyu Yi, Yuxiao Zhou, Marc Habermann, Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Feng Xu. 2022. Physical inertial poser (pip): Physics-aware real-time human motion tracking from sparse inertial sensors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 13167–13178.

[68] Xinyu Yi, Yuxiao Zhou, and Feng Xu. 2021. Transpose: Real-time 3d human translation and pose estimation with six inertial sensors. *ACM Transactions On Graphics (TOG)* 40, 4 (2021), 1–13.

[69] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. 2020. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214* (2020).

[70] Xiaoqin Zhang, Weiming Hu, Nianhua Xie, Hujun Bao, and Stephen Maybank. 2015. A robust tracking system for low frame rate video. *International Journal of Computer Vision* 115 (2015), 279–304.

[71] Yinlong Zhang, Jindong Tan, Ziming Zeng, Wei Liang, and Ye Xia. 2014. Monocular camera and IMU integration for indoor position estimation. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 1198–1201.

[72] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. 2020. Monocular real-time hand shape and motion capture using multi-modal data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5346–5355.

[73] Jun Zhu, Hongyi Li, and Tao Zhang. 2023. Camera, LiDAR, and IMU based multi-sensor fusion SLAM: A survey. *Tsinghua Science and Technology* 29, 2 (2023), 415–429.

[74] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. 2017. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2349–2358.

A Details of the Extended Kalman Filter (EKF)

Here we provide a detailed mathematical formulation of the EKF model. Based on the dynamic model described in Equation 3.2.3, the goal of the EKF model is to leverage the uncertainties of the measurements and the dynamic model to achieve a more robust and balanced hand tracking result. The full process can be divided into three stages, including (1) the prediction of the system state and uncertainty from the dynamic model, (2) the update of the system state and uncertainty from the predicted state and the measurement, and (3) the estimation of the Kalman gain, a factor to balance the prediction and measurement. The main equations can be formulated as:

$$
\begin{aligned}
&(Kalman\ Gain) \quad \boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}(\boldsymbol{P}_{n,n-1} + \boldsymbol{R}_n)^{-1} \\
&(State\ Prediction) \quad \hat{\boldsymbol{x}}_{n+1,n} = f(\hat{\boldsymbol{x}}_{n,n}, \boldsymbol{u}_n) \\
&(Covariance\ Prediction) \quad \boldsymbol{P}_{n+1,n} = \frac{\partial f}{\partial \boldsymbol{x}} \boldsymbol{P}_{n,n} (\frac{\partial f}{\partial \boldsymbol{x}})^T + \boldsymbol{Q}_n \\
&(State\ Update) \quad \hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{x}_{n,n-1}) \\
&(Covariance\ Update) \quad \boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n)\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n)^T + \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_n^T
\end{aligned} \tag{5}
$$

where $\boldsymbol{x} \in \mathbb{R}^{12}$ is the system state, including position $\boldsymbol{p} \in \mathbb{R}^3$ and rotation $\boldsymbol{r} \in \mathbb{R}^{3\times3}$. $\boldsymbol{z} \in \mathbb{R}^{12}$ is the LFR measurement, also including position $\boldsymbol{p}$ and rotation $\boldsymbol{r}$, which is the same as the system state. $\boldsymbol{u} \in \mathbb{R}^6$ is the control input, including the global hand velocity $\hat{\boldsymbol{v}} \in \mathbb{R}^3$ predicted by the RNN model and augular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ from the

gyroscope. $\boldsymbol{K} \in \mathbb{R}^{12\times12}$ is the calculated Kalman gain. $\boldsymbol{P} \in \mathbb{R}^{12\times12}$ is the covariance matrix of the system state, which represents its uncertainty. $\boldsymbol{Q} \in \mathbb{R}^{12\times12}$ and $\boldsymbol{R} \in \mathbb{R}^{12\times12}$ are the process noise and measurement noise covariance matrices respectively, representing the uncertainties of the dynamic model and measurements. For the subscripts, a single subscript, like n, represents a variable at time n. While, double subscripts, like $n, n-1$, represent a predicted variable at time n based on the information at time $n-1$. Finally, the function f is the dynamic model, which predicts the next system state from the current state and the control input. It is given by:

$$\begin{aligned} &(Position\ Prediction) \quad \boldsymbol{p}_{n+1,n} = \boldsymbol{p}_{n,n} + \hat{\boldsymbol{v}}_n \Delta t \\ &(Rotation\ Prediction) \quad \boldsymbol{r}_{n+1,n} = matrix(\boldsymbol{r}_{n,n}\boldsymbol{\omega}_n \Delta t)\boldsymbol{r}_{n,n} \end{aligned} \tag{6}$$

which is similar to Equation 3.2.3.

In the above EKF equations, $\boldsymbol{z}$ and $\boldsymbol{u}$ are measured by the Quest Pro and the IMU ring. $\boldsymbol{x}$, $\boldsymbol{P}$, and $\boldsymbol{K}$ are automatically estimated by the EKF process. $\boldsymbol{R}$ and $\boldsymbol{Q}$ are the uncertainties of the measurement and the dynamic model respectively, which still need to be determined manually. For $\boldsymbol{R}$, it is defined as the covariance of the optical measurement noise. Since we do not have access to the error statistics of the Quest Pro, we empirically extract the high-frequency components (above 8 Hz) of the Quest Pro hand-tracking results using a Butterworth filter, treat these components as the noise of the low-frame-rate (LFR) measurements, and compute their covariance matrix as $\boldsymbol{R}$. For $\boldsymbol{Q}$, it is defined as the covariance of the process noise, which is the error of the predicted hand position and rotation from the dynamic model. Since our design uses an adaptive framerate, the number of times the dynamic model predicts the hand position and rotation before the next optical measurement arrives is not fixed. More consecutive predictions lead to larger accumulated errors. Therefore, $\boldsymbol{Q}$ also depends on the number of consecutive predictions. To estimate this uncertainty, we start from given initial hand positions and rotations, perform different numbers of consecutive predictions, and compare each predicted result with the Quest Pro measurements to compute the corresponding errors. This yields a series of $\boldsymbol{Q}$ matrices representing the process noise for different prediction steps. During realtime use, we select the $\boldsymbol{Q}$ associated with the number of consecutive predictions performed as the process noise.

Finally, we define the complete execution procedure of the EKF model: the update process is performed only when an optical measurement arrives, while the prediction process is executed at all steps, during which the Kalman gain is updated as well. This allows the EKF to dynamically estimate the errors of the optical measurements and the system predictions, producing more robust hand-tracking results.