MOUNT: Learning 6DoF Motion Prediction based on Uncertainty Estimation for Delayed AR Rendering

Haoran Chen, Lantian Wei, Haomin Liu, Boxin Shi, Guofeng Zhang, Hongbin Zha

Abstract—The delay of rendering on AR devices requires prediction of head motion using sensor data acquired tens of even one hundred milliseconds ago to avoid misalignment between the virtual content and the physical world, where the misalignment will lead to a sense of time latency and dizziness for users. To solve the problem, we propose a method for the 6DoF motion prediction to compensate for the time latency. Compared with traditional hand-crafted methods, our method is based on deep learning, which has better motion prediction ability to deal with complex human motion. In particular, we propose a MOtion UNcerTainty encode decode network (MOUNT) that estimates the uncertainty of input data and predicts the uncertainty of output motion to improve the prediction accuracy and smoothness. Experiments on the EuRoC and our collected dataset demonstrate that our method significantly outperforms the traditional method and greatly improves AR visual effects.

Index Terms—Virtual and augmented reality, Learning environments, Learning technologies

1 Introduction

With the development of computer vision and the gradual maturity of hardware, augmented reality (AR) technology has ushered in rapid development [1]. AR technology can merge virtual content into the physical world and allow users to interact with them. Current smartphones can already provide AR experiences powered by ARKit or ARCore [2], [3], but undoubtedly AR glasses will provide a more immersive experience and natural interaction. Many people believe that AR glasses will replace smartphones as the next generation of intelligent terminals. However, both hardware and software for AR glasses are currently not mature enough.

When an AR application is running on the smartphone, virtual contents are overlaid on the video, which is called video see-through (VST) AR. The commonly used technique is visual-inertial SLAM (VI-SLAM) which fuses visual and inertial measurements to estimate the 6DoF motion for each image frame in order to render the virtual content at the same viewpoint. By contrast, AR glasses allow virtual content to be blended with the direct view of the physical world seen by human eyes, called optical see-through (OST) AR. The OST solution can provide a more immersive experience but also poses challenges on the 6DoF motion estimation. Before rendering, the 6DoF viewpoint at the time human sees the virtual content should be predicted in advance, as shown in Figure 1. Simply using the latest image pose as the VST solution will result in misalignment between virtual content and the physical world. For high-performance hardware such as HoloLens, where the rendering delay is usually short, the quality of the prediction algorithm doesn't make much difference. However, for hardware with low performance and cost, the prediction algorithm will play an important role due to the large rendering latency. The amount of rendering delay depends not only on the performance of hardware, but also the complexity of virtual content. The total rendering delay for medium-size content is typically 30ms, which is tested on the Shadow creator Action One Pro, based on Qualcomm Snapdragon 835, Adreno 540, and the system of Android, with RAM of 6GB. There is always a trade-off between the lightweight of device and quality of content. The trend of cloud-based rendering may be a solution to break this dilemma, but at the cost of a larger amount of rendering delay. Considering the delay of network, the total delay may exceed 100ms before the 5G networks are truly widespread

To compensate delayed rendering, we study the task of 6DoF motion prediction to reduce the error between the true rendering viewpoint and the predicted one using sensor data acquired a period of time ago. Traditionally this is obtained by two phases: 1) Using inertial measurement unit (IMU) measurements to propagate the 6DoF pose from the latest image to the latest IMU measurement and 2) handcrafting a motion model to predict motion from the latest IMU measurement to the rendering time. Due to repaid error accumulation, traditional methods do not respond well to complex human motion patterns and can only perform short-term prediction. By contrast, learning-based methods are able to extract the motion pattern better. Previous works on the prediction and classification of human motion have proved that the learning-based method is more effective than the hand-crafted method [4]. In this work, we propose a learning based method to improve the quality of motion prediction for delayed AR rendering.

Due to the widespread noise in real data, there is a degree of

Haoran Chen is with the AI Innovation Center, School of Computer Science, Peking University, Beijing, China (e-mail: chrer@pku.edu.cn).
Lantian Wei and Haomin Liu are with the Key Lab of Machine Perception (MOE), School of Artificial Intelligence, Peking University, Beijing, China, and Haomin Liu is also with SenseTime Research, Beijing, China (e-mail: weilantian@pku.edu.cn; liuhaomin@sensetime.com).
Boxin Shi is with the National Engineering Research Center of Visual Technology, School of Computer Science, Peking University, Beijing, China (e-mail: shiboxin@pku.edu.cn).
Guofeng Zhang is with the state key lab of CAD&CG, Zhejiang University, Hangzhou, China (e-mail: zhangguofeng@zju.edu.cn).
Hongbin Zha is with the Key Lab of Machine Perception (MOE), School of Artificial Intelligence, Peking University, Beijing, China (e-mail: zha@cis.pku.edu.cn).
This work was partially supported by the National Key Research and Development Program of China under Grant 2020YFF0304300, and the National Natural Science Foundation of China (No. 61932003 and No. 62136001).

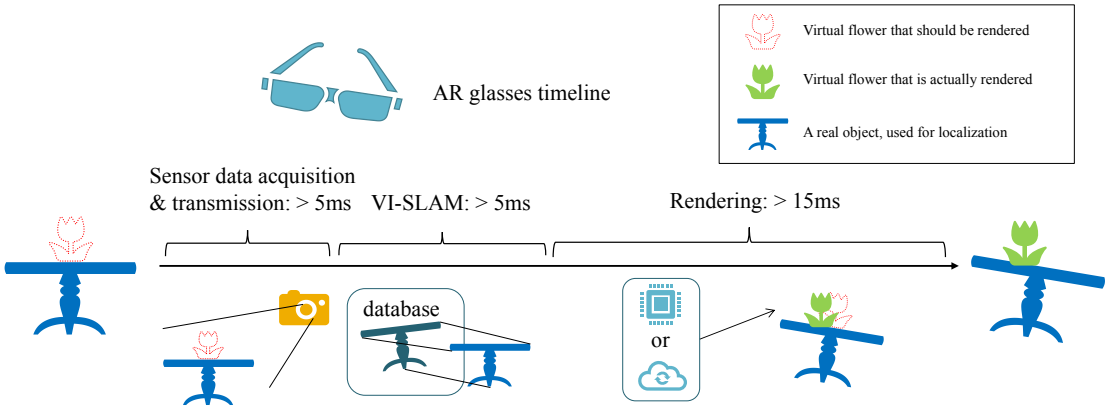


Fig. 1: Illustration of rendering delay in AR. AR glasses take 30 to 60 milliseconds for sensor data acquisition and transmission, VI-SLAM, and rendering. During the period, the motion of human will cause the actual viewpoint to change, resulting in the flower rendered in the figure (green) to be offset from the actual position (red).

uncertainty in input data. Therefore, we design a MOtion UNcerTainty encode decode network (MOUNT) based on quantification of uncertainty, which can estimate uncertainty of input data and learn the uncertainty prediction of the output from training data without supervision. The estimation and prediction of uncertainty can improve the prediction accuracy and make the prediction trajectory smoother through post-processing. Experiments show that MOUNT has more accurate motion prediction ability and can achieve better compensation effect of delayed rendering, compared with traditional methods.

In the further experiments, we find that MOUNT can make long-term motion predictions by anticipating the changes of motion trends in advance, providing a reasonable explanation for its strong prediction ability. At the same time, due to MOUNT's ability of uncertainty estimation, it can also make a reasonable prediction at a lower camera frame rate, which opens a potential to lower the power consumption for lightweight AR devices.

Our contribution can be summarized as:

- We introduce uncertainty to 6DoF motion prediction to improve the task's performance by estimating uncertainty in input data and output prediction.
- We propose a learning framework MOUNT, which has a reasonable data pre-processing method and architecture, and can learn uncertainty prediction of the output from training data without supervision.
- We define the task and evaluation metrics of 6DoF motion prediction, and conduct experiments on the public dataset EuRoC [5] and our collected dataset designed explicitly for the task, which quantitatively proves the effectiveness of MOUNT and shows immersive AR effect that MOUNT can bring.

The rest of this paper is organized as follows. In section 2, related works are reviewed. In section 3, we give the mathematical definition of the task, the evaluation metric, and the traditional method as the baseline. Our method is described in detail in section 4. Experiments and performance of the proposed method in various situations are shown in section 5. We summarize the paper in the section 6 and point out future works. The source code and our collected dataset will be published for the benefit of the community[1].

1. https://github.com/braveryCHR/MOUNT-6DoF-Motion-Prediction

2 Related Work

2.1 Asynchronous Timewarp

Asynchronous Timewarp (ATW) is a technology that warps the rendered image before sending it to the display to correct the head movement after the scene is rendered. It is widely used in VR head-mounted display (HMD) devices in order to reduce latency, increase frame rate, and reduce judder-caused missed frames [6]. A common approach is to predict head movements such that stereoscopic images can be rendered from the viewpoint at which the head is going to be in the near future [7], [8]. Since the prediction inevitably has an error, it has to be further corrected when the rendered image is displayed, using the incoming head tracking information to warp images from the predicted viewpoint to the corrected one. However, image warping can only compensate for orientation changes, but not the translation [9], [10].

When ATW is applied to AR, the requirement for motion prediction becomes stricter than VR. Many VR HMD devices only perform 3DoF orientation head tracking. Even for those 6DoF VR devices like Oculus Quest2, the slight error is hardly noticed because the user can only see the virtual content without references. By contrast, for the AR rendering where the virtual content is blended with the physical world, both orientation and translation error will result in misalignment, leading to an uncomfortable experience. The existing methods designed for VR tend to result in an uncomfortable AR experience, especially for low-end AR devices. It is necessary to customize ATW algorithms for delayed AR. Furthermore, a framework based on extended Kalman filtering (EKF) [11] for AR system is proposed to fuse information and predict motion. It has certain similarities to our work. The main difference is that it assumes a certain motion model to predict motion at the future time, while our method is data-driven to adapt to complex motion styles and suitable for long-time prediction.

2.2 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) is the technique of tracking the 6DoF motion in the unknown environment using inside-out sensors like camera and IMU on the mobile device [12]. Compared to the outside-in solutions such as HTC Vive [13], the inside-out solution relaxes the requirement of external sensors in the environment and also achieves better tracking accuracy [14]. It has been successfully used in the most advanced VR and

AR devices like Oculus Quest2 and Microsoft HoloLens and is becoming a standard solution for 6DoF head tracking.

SLAM has been studied for over 30 years. Pioneering works like MonoSLAM [15] and PTAM [16] are based on pure vision, suffering from the robustness issue in textureless environments. ORBSLAM [17], [18] improves accuracy and robustness by using feature descriptors [19], [20] for loop closure and re-localization. Other lines of works [21], [22] replace the sparse features with semi-dense pixels to further improve the robustness. However, the inherent limitation of the pure vision-based method is still difficult to be fully overcome until the emergence of VI-SLAM that fuses the complimentary visual and inertial measurements, achieving the highest robustness and accuracy [23]. To tightly fuse visual and inertial measurements, the filter-based methods [24], [25], [26] use inertial measurement for state propagation and visual measurement for state update, while the optimization-based methods [27], [28] pre-integrate IMU measurements between consecutive frame as constraint [29] and solve the optimization problem [30]. The optimization-based methods are generally more accurate than filter-based methods, at the cost of computational complexity [23]. Recently, many methods based on deep learning [31], [32] have emerged, successfully introducing the latest machine learning methods into the field.

When VI-SLAM is applied on consumer-level mobile devices with constrained resources, efficiency becomes a critical demand. iSAM2 [33] and ICE-BA [34] exploit the incremental nature of SLAM to replace batch optimization with incremental one without sacrificing accuracy. Other works design customized chip for feature extraction [35] or visual-inertial fusion [36], [37]. As the efficiency of VI-SLAM is being optimized to the extreme, the camera is becoming the main source of power consumption. Lowering the camera frame rate will reduce the power consumption naturally, but will also result in tracking drift because of the propagation from an early image frame by IMU measurements. To the best of our knowledge, this is the first work to open this possibility by quantifying and encoding IMU propagation noise.

2.3 Inertial-only Odometry

Inertial-only odometry is widely used in the field of pedestrian dead reckoning (PDR) [38]. Since IMU only provides angular velocity and acceleration measurements that have to be integrated into pose estimates, the noise will also be accumulated quickly, resulting in severe drift after a short period of time. Early approaches rely on the prior knowledge of human walking motion such as Zero-velocity UPdaTe (ZUPT) [39], Zero Angular Rate Update (ZARU) [40] or step counting [41] to alleviate the drift. Recently, the emergence of deep learning provides new possibilities to extract the style of motion pattern from training data to replace the hand-crafted priors and has achieved significantly better results for both 2D [42], [43] and 3D [44] movement estimation.

Compared to the application of PDR, the application of AR requires far more accurate motion estimation. Therefore, the delayed AR rendering needs the extension of the inertial-only method to get the motion in the near future without IMU measurements.

3 Task Definition

3.1 Task

The task of motion prediction is defined as follows. Given a motion sequence up to time n, we have

$$\mathbb{I} = \{I_1, I_2 \cdots I_n\}, \quad \mathbb{P} = \{P_1, P_2 \cdots P_n\}, \tag{1}$$

where $\mathbb{I}$, $\mathbb{P}$ are IMU measurement sequence and pose sequence respectively. At each time i,

$$I_i = (\omega_i, a_i), \quad P_i = (R_i, T_i). \tag{2}$$

The IMU measurement I_i is comprised of the angular velocity ω_i and the acceleration a_i. For simplicity, we assume the IMU bias has been subtracted. The pose $P_i \in \mathrm{SE}(3)$ is comprised of the 3×3 rotation matrix R_i and the translation T_i.

The task is to implement a function F that maps the given sequences up to current time n to a pose P_m at the future time m:

$$P_m = \mathrm{F}(\mathbb{I}, \mathbb{P}). \tag{3}$$

Depending on the hardware capability or cloud transmission, and the complexity of rendered content, the required prediction time between n and m is typically about $30 \sim 100$ milliseconds.

It should be noted that typical AR devices are equipped with 6-axis IMU and VI-SLAM system, which will run in parallel with the 6DoF motion prediction algorithm. Hence the acceleration ω and angular velocity a in this task are derived from IMU, and the IMU bias and pose sequence P can be estimated by the VI-SLAM system. Obviously, the task definition can be adapted to typical AR devices, which align with the actual situation.

3.2 Metric

To ensure a comfortable AR experience, we find that two metrics are essential for motion prediction.

Accuracy measures the error between the predicted trajectory and ground truth. The error will result in misalignment between virtual content and the physical world. Given two pose trajectories $\mathbb{P}^{pred}$ and $\mathbb{P}^{GT}$ representing the predicted and ground truth trajectory respectively, the **average error (AE)** is calculated as

$$AE(\mathbb{P}^{pred}, \mathbb{P}^{GT}) = \sum_{i=1}^{n} \frac{|\mathrm{Log}(P_i^{pred} \ominus P_i^{GT})|}{n}, \tag{4}$$

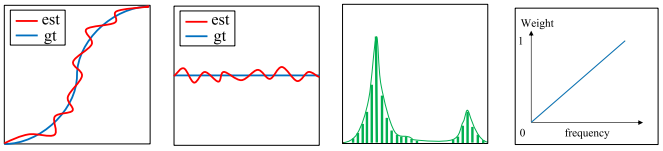
where $\ominus$ is the subtraction operator for $\mathrm{SE}(3)$, and $\mathrm{Log}(P)$ maps a pose $P \in \mathrm{SE}(3)$ to the minimal 6D vector, 3D for rotation and 3D for translation.

Smoothness measures the degree of jitter of the pose trajectory. The jitter of the pose will also cause the jitter of virtual content, leading to an uncomfortable AR experience. In order to measure the smoothness, we propose the **normalized frequency (NF)** as the criterion of smoothness. It follows a simple principle: the more severe the trajectory jitters, the more high-frequency components exist in the error expressed in the frequency domain [45]. Figure 2 shows the calculation process of NF, which is formulated as:

$$\begin{aligned} &\mathbb{P}^{error} = \{|\mathrm{Log}(P_i^{pred} \ominus P_i^{GT})|\}, \\ &\mathbb{P}_{freq}^{error} = \mathrm{DFT}(\mathbb{P}^{error}) = \{P_{freq}^{error}(i) | i = 1 \cdots N\}, \\ &NF = \sum_{i=1}^{N} \frac{w(i) \cdot P_{freq}^{error}(i)}{N}, w(i) = \frac{i}{N}, \end{aligned} \tag{5}$$

where DFT denotes the discrete Fourier transform [46] process, N is the number of frequency components and $P_{freq}^{error}(i)$ denotes the magnitude of response at frequency i. NF is calculated as the weighted mean over all frequency components. The weighting function $w(i)$ is selected as the linear function to eliminate the influence of the trajectory length on numerical results. For example, under the above weighting function, assume $a_1 = \{1, 2, 1\}$,

$b_1 = \{1, 1, 1\}$, $a_2 = \{1, 2, 1, 1, 2, 1\}$, $b_2 = \{1, 1, 1, 1, 1, 1\}$, then $NF(a_1, b_1)$ is equal to $NF(a_2, b_2)$, so that the calculation result NF is only related to the degree of jitter, and has nothing to do with the trajectory length.



(a) (b) (c) (d)

Fig. 2: The calculation steps of NF. (a) The small but frequent vibration between the estimated trajectory and ground truth will be perceived as jitter (abscissa: time, ordinate: value). (b) Calculate the error between the predicted trajectory and ground truth (abscissa: time, ordinate: error). (c) Use Discrete Fourier Transform to convert the error from the time domain to the frequency domain (abscissa: frequency, ordinate: coefficient). (d) Calculate the normalized frequency by the weighted mean overall frequency components.

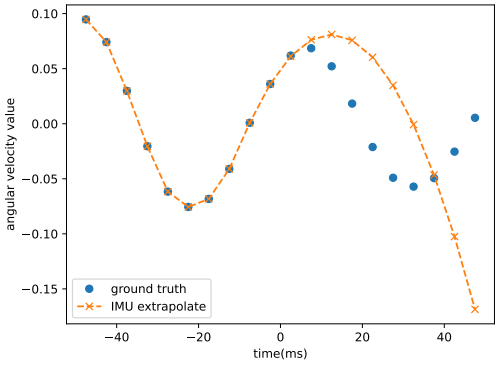


Fig. 3: A typical IMU extrapolation example. The second-order spline method is used to fit data before 0ms and extrapolate data after 0ms.

3.3 Baselines

Since we are the first to use neural networks in the task of 6DoF motion prediction, the performance of our model can only be compared with existing traditional methods. Products already on sale, such as HoloLens, do not provide an interface to obtain the results of their algorithm, nor provide their rendered images through predicted poses. Therefore, its algorithm cannot be compared with our algorithm under fair conditions for the time being.

In the published literature, a sensor fusion framework based on EKF is proposed [11] to predict motion. In the EKF framework, the state of pose, velocity and IMU biases are filtered, using high frequency IMU measurements for state propagation, and low frequency pose measurements for state update. To predict pose at the future time without IMU measurements, it assumes a certain motion model (constant linear and angular velocity) for the pose propagation. We implement the method as baseline to our method.

Another straightforward method is to extrapolate IMU measurements to the future and predict the 6DoF pose using these extrapolated IMU measurements. This method is effective for short-term motion prediction. Precisely, it is calculated as

$$\begin{aligned} &\{I_{n+1}, I_{n+2} \cdots I_m\} = \mathrm{SE}(\mathbb{I}), \\ &P_m = \mathrm{G}(P_n, V_n, \{I_{n+1}, I_{n+2} \cdots I_m\}), \end{aligned} \tag{6}$$

where SE denotes the spline extrapolation algorithm and G represents the process of motion propagation from the latest pose P_n and velocity V_n (calculated by adjacent translation) at time n to the future time m by IMU integration [29].

Figure 3 shows a typical example. It can be seen that it is possible to extrapolate reasonable IMU measurements in a short period of time. As the prediction time increases, the actual motion gradually goes beyond the prediction ability of extrapolation.

4 Our Approach

Figure 4 shows the overview structure of the MOUNT.

4.1 Encoder

In the VI-SLAM system, the IMU frequency (100-1000Hz) is much higher than the image frequency (10-60Hz). Although VI-SLAM can estimate pose at the IMU frequency, the accuracy of IMU pose depends on how long it is away from its latest image, as shown in Figure 5. The VI-SLAM algorithm performs IMU propagation from the latest image pose at IMU rate, accumulating drift over time, and visual-inertial optimization or filter update at camera rate to bound the drift. Accordingly, we classify poses into two categories, **vision pose** and **IMU pose**.

Pose sequence preprocess: For the pose sequence, taking into account limitations of human body and movement speed [47], the movement of humans wearing AR glasses in a short period is restricted to a small range. At the same time, the numerical domain of the absolute translation is too wide, so it is a better choice to use the relative pose as input. This strategy also eliminates the influence of the selection of world coordinate, which can be hardly learned from training data. Specifically, we present each IMU pose P_i at time i in the coordinate of its latest vision pose P_{v_i} at time v_i. The relative translation component is simply $\Delta T_i = T_i - T_{v_i}$. For the rotation component, we use the minimal 3D rotation vector [48] as the presentation, denoted as $\mathrm{Log}(R_{v_i}^T R_i)$, where $\mathrm{Log}(R)$ maps a rotation matrix R to its rotation vector representation. In this way, we preprocess the pose sequence as

$$\begin{aligned} &\mathbb{P}' = \{(\mathrm{Log}(\Delta R_i), \Delta T_i)\}, \\ &\Delta R_i = R_{v_i}^T R_i, \quad \Delta T_i = T_i - T_{v_i}. \end{aligned} \tag{7}$$

IMU sequence preprocess: As in [43], [49], we remove the influence of gravity by subtracting the gravity component from acceleration measurement.

$$\mathbb{I}' = \{(\omega_i, a_i')\}, \quad a_i' = a_i - R_i^T g, \tag{8}$$

where R_i represents the rotation from the body coordinate system to the world coordinate system at time i, and g denotes the gravity vector in the world coordinate.

Pose uncertainty estimation: Due to differences in the accuracy of poses in different positions as shown in Figure 5, we need a way to tell the model uncertainty of each pose in the sequence so that it knows which poses are worthy of trust. As proposed in [29], we assume the true state of (R_i, V_i, T_i) is comprised of the estimated state $(\hat{R}_i, \hat{V}_i, \hat{T}_i)$ and Gaussian noise $(\tilde{r}_i, \tilde{v}_i, \tilde{t}_i)$

$$R_i = \mathrm{Exp}(\tilde{r}_i)\hat{R}_i, \quad V_i = \hat{V}_i + \tilde{v}_i, \quad T_i = \hat{T}_i + \tilde{t}_i, \tag{9}$$

where $\mathrm{Exp}(v)$ maps a 3D vector v to a rotation matrix. The true IMU value is also comprised of the measurement $(\hat{\omega}_i, \hat{a}_i)$ and Gaussian noise $(\tilde{\omega}_i, \tilde{a}_i)$

$$\omega_i = \hat{\omega}_i + \tilde{\omega}_i, \quad a_i = \hat{a}_i + \tilde{a}_i. \tag{10}$$

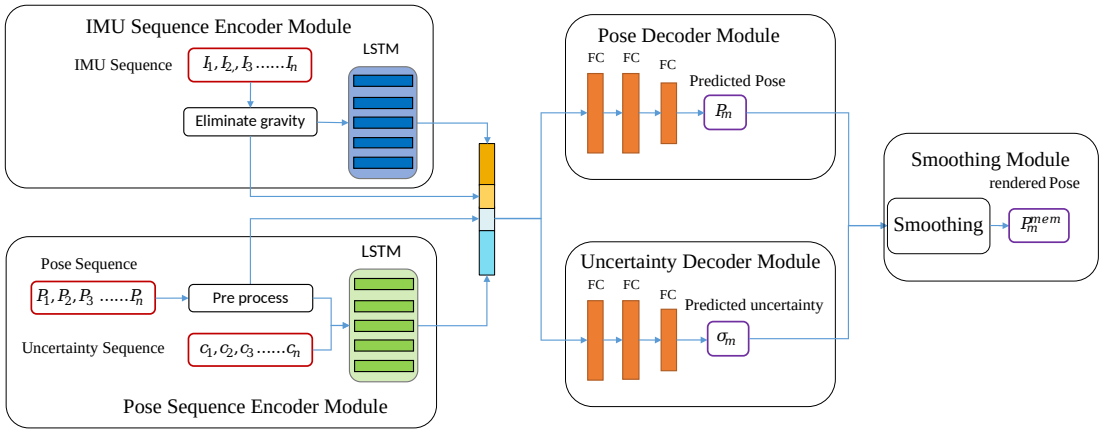


Fig. 4: Overview structure of the MOUNT. The encoder part separately encodes the IMU and pose sequences, and merges all representations to the intermediate vector. The decoder part respectively outputs the predicted pose and the uncertainty of the prediction. After smoothing, the final pose can be used for rendering.

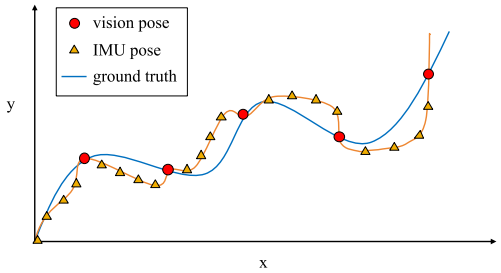


Fig. 5: Two different types of poses in VI-SLAM. The vision pose is relatively accurate, while the IMU pose gradually diverges as the noise error accumulates.

The error of state $\tilde{x}_i = (\tilde{r}_i^T, \tilde{v}_i^T, \tilde{t}_i^T)$ can be propagated as

$$\tilde{x}_{i+1} = A_i \tilde{x}_i + B_i \tilde{n}_i, \tag{11}$$

where $\tilde{n}_i = (\tilde{\omega}_i^T, \tilde{a}_i^T)$ and

$$\begin{aligned} A_i &= \begin{pmatrix} \mathrm{Exp}(\hat{\omega}_i \Delta t) & 0 & 0 \\ -\hat{R}_i [\hat{a}_i]_\times \Delta t & I & 0 \\ -\frac{1}{2}\hat{R}_i [\hat{a}_i]_\times \Delta t^2 & I\Delta t & I \end{pmatrix}, \\ B_i &= \begin{pmatrix} J_r(\hat{\omega}_i \Delta t)\Delta t & 0 \\ 0 & \hat{R}_i \Delta t \\ 0 & \frac{1}{2}\hat{R}_i \Delta t^2 \end{pmatrix}, \end{aligned} \tag{12}$$

where Δt is the time interval from i to $i+1$, $[v]_\times$ is the skew symmetric matrix of vector v, and J_r is the right Jacobian of SO3. The covariance matrix of $\tilde{x}_i$ is propagated accordingly as

$$C_{i+1} = A_i C_i A_i^T + B_i Q B_i^T, \tag{13}$$

where Q is the pre-defined covariance matrix of $\tilde{n}_i$. Finally, the error is propagated from absolute pose to the relative pose defined in Equation 7

$$\Delta\tilde{x}_i = D_i \tilde{x}_i = \begin{pmatrix} J_r^{-1}(\mathrm{Log}(\Delta\hat{R}_i))\Delta\hat{R}_i & 0 & 0 \\ 0 & 0 & I \end{pmatrix} \tilde{x}_i. \tag{14}$$

The covariance matrix of $\Delta\tilde{x}_i$ is propagated accordingly as

$$C_i' = D_i C_i D_i^T. \tag{15}$$

In this way, we obtain the covariance sequence

$$\mathbb{C}' = \{c_1', c_2' \cdots c_n'\}, \tag{16}$$

where c_i' is the 6D diagonal vector of C_i'.

Encoding: We use two independent LSTMs [50] to encode several sequences.

$$M_I = \mathrm{LSTM}_I(\mathbb{I}'), \quad M_P = \mathrm{LSTM}_P(\mathbb{P}', \mathbb{C}'). \tag{17}$$

4.2 Decoder

Representation: LSTM does well in learning trends and patterns. However, it is more difficult to use it to perform accurate numerical regression directly. Therefore, we need to directly provide the latest l poses and IMU measurements to the neural network.

$$\begin{aligned} &\mathbb{P}_l = (P'_{n-l+1} \cdots P'_n), \quad \mathbb{I}_l = (I'_{n-l+1} \cdots I'_n), \\ &M = (M_I, M_P, \mathbb{P}_l, \mathbb{I}_l). \end{aligned} \tag{18}$$

The decoding representation vector contains two parts. The first part (M_I, M_P) is called **motion pattern encoding**, which is derived from the LSTM encoding and represents the mode and trend of the motion. The second part $(\mathbb{P}_l, \mathbb{I}_l)$ is called **value encoding**, which provides a direct numerical reference for decoding.

Decoding: The part of the decoder is comprised of two sequences of fully connected layers, which will decode the representation vector to obtain the predicted pose and the uncertainty prediction for the output.

Taking into account the numerical stability of the model output, the output pose predicted by the model is the relative pose of P_m to P_n.

$$\Delta P_m^{pred} = \mathrm{FC}_P(M), \quad P_m^{pred} = P_n \oplus \Delta P_m^{pred}, \tag{19}$$

where $\oplus$ is the addition operator for $\mathrm{SE}(3)$. Uncertainty of the prediction can be obtained by the following formulas

$$s^{pred} = \mathrm{FC}_\sigma(M), \quad \sigma^{pred} = \mathrm{Norm}(\exp(\frac{1}{2}s^{pred})), \tag{20}$$

where Norm represents the function that normalizes σ to [0,1].

4.3 Loss Function

Obviously, if the model only needs to predict the pose, minimizing the mean square loss of the predicted pose and the actual pose (L2 loss) is enough.

$$\begin{aligned} L_{pose} &= ||\mathrm{Log}(P_m^{GT} \ominus P_m^{pred})||^2 \\ &= ||\mathrm{Log}(\Delta P_m^{GT} \ominus \mathrm{F}_\theta(\mathbb{I}, \mathbb{P}))||^2, \end{aligned} \tag{21}$$

where θ represents the parameters of our proposed model and ΔP_m^{GT} represents the label of relative pose $P_m \ominus P_n$.

However, our model not only predicts the pose but also predicts the uncertainty of the output pose, which will be used to smooth the trajectory. The smoothing strategy will be described in section 4.4. We introduce our method for pose prediction with uncertainty here, which is similar to the method proposed in [51].

The error of the prediction result is assumed to be normally distributed as $\mathcal{N}(0, \sigma^2)$, where σ^2 is the variance of prediction error. The probability is formulated as

$$\begin{aligned} \mathrm{P}_\theta &= \mathrm{P}(\Delta P_m^{GT} | \mathrm{F}_\theta(\mathbb{I}, \mathbb{P})) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{||\mathrm{Log}(\Delta P_m^{GT} \ominus \mathrm{F}_\theta(\mathbb{I}, \mathbb{P}))||^2}{2\sigma^2}). \end{aligned} \tag{22}$$

The goal is to make a maximum a posterior (MAP) inference by optimizing the model parameters, which is formulated as

$$\begin{aligned} \theta^* &= \arg\max_\theta \mathrm{P}_\theta = \arg\min_\theta(-\log \mathrm{P}_\theta) \\ &= \arg\min_\theta \frac{||\mathrm{Log}(\Delta P_m^{GT} \ominus \mathrm{F}_\theta(\mathbb{I}, \mathbb{P}))||^2}{2\sigma^2} + \frac{1}{2}\log\sigma^2. \end{aligned} \tag{23}$$

Consider the numerical stability, our model predicts the logarithm of the variance $s = \log\sigma^2$. The final loss function is

$$L_{final} = \frac{1}{2}\frac{||\mathrm{Log}(\Delta P_m^{GT} \ominus \mathrm{F}_\theta(\mathbb{I}, \mathbb{P}))||^2}{\exp(s)} + \frac{1}{2}s. \tag{24}$$

4.4 Smoothing Strategy

In section 3.2, we propose two metrics to evaluate the quality of motion prediction, considering accuracy and smoothness. Although LSTM naturally has a certain smoothing ability, we found that is not enough and the jitter still exists. Our model has the ability to predict uncertainty. With the help of uncertainty, we propose a simple smoothing strategy that can improve smoothness.

Let $\Delta P_m^{mem} = (\Delta T_m^{mem}, \Delta R_m^{mem})$ be the prediction memory, which contains the previous prediction information. Whenever a new prediction comes, use the following strategy to update:

$$\begin{aligned} \Delta T_m^{mem} &= (1-\sigma) \cdot \Delta T_m^{pred} + \sigma \cdot \Delta T_m^{mem}, \\ \Delta R_m^{mem} &= \mathrm{slerp}(\Delta R_m^{mem}, \Delta R_m^{pred}, 1-\sigma), \end{aligned} \tag{25}$$

where slerp represents spherical linear interpolation for rotation. The less uncertain the prediction ΔP_m^{pred} is, the closer ΔP_m^{mem} will be to this prediction. Conversely, ΔP_m^{mem} will believe more in historical predictions. The final rendered pose is obtained by

$$P_m^{pred} = P_n \oplus \Delta P_m^{mem}. \tag{26}$$

5 Experiments

5.1 Training Details

5.1.1 Dataset

We first verify MOUNT's performance on the EuRoC dataset [5], one of the most influential indoor VI-SLAM benchmark datasets. It contains 11 motion sequences, including three scenes, Machine Hall (MH) and two Vicon Room (V1 and V2). Each sequence contains 20Hz image data, 200Hz IMU measurements, and 200Hz ground truth poses.

However, EuRoC [5] is a dataset collected by drones, which is different from human motion in AR application. Therefore, we use a pair of AR glasses (Shadow Creator Action One Pro) to collect a dataset of human motion wearing AR glasses. The ground truth poses are obtained by the method of LSFB [52]. The method is based on an accurate HD map, visual localization and offline visual-inertial optimization. Details are referred to [52].

As shown in Table 1, we classify AR motion according to two dimensions: environment and motion purpose. Environments are divided into four categories according to their spaciousness: "narrow", "near", "far" and special "stair". Motion purposes are divided into five categories: "forward", "inspect", "patrol", "run", "focus". It needs to be further explained that "forward" represents normal walking forward, "inspect" represents standing still and observing the surrounding environment, "patrol" represents exploring the environment around, "focus" represents careful observation of specific objects, and "run" represents the state of running forward. It can be seen that the environments and purposes mentioned above can cover common AR scenarios.

Through a reasonable combination of environment and purpose, we simulate user's behaviour wearing AR glasses, collect trajectories and process the raw data into a form suitable for the task. Some environments and purposes are unsuitable for the combination and are therefore abandoned. Finally, our collected dataset is constructed for the validation of the task. It contains 14 motion sequences, including 5 kinds of purposes and 4 kinds of environments. For each sequence, it contains 30Hz image data, 300Hz IMU measurements, and 300Hz ground truth poses.

TABLE 1: Trajectories in our collected dataset. 14 trajectories are collected from two dimensions of environment and purpose, which is designed specifically for human motion scenes in AR.

environment|purpose	forward	inspect	patrol	run	focus
narrow	✓	✓	✓		
near	✓	✓	✓		✓
far	✓	✓	✓	✓	✓
stair	✓		✓		

5.1.2 Train, test, and validation split

For EuRoC [5], we use the MH and V1 sequences as the training set and V2 sequences as the test set. For our dataset, we use the first 70% of sequences as the training set, the next 30% as the test set. During training, the last 20% of the training set is used to validate the performance of model.

5.1.3 Simulation

In order to simulate the real situation, we use ground truth and IMU propagation to simulate input data to the model as shown in Figure 6. For high-frequency ground truth poses, only vision poses synchronized with image data are retained. The remaining poses will be replaced by results of the IMU propagation from the latest vision pose using IMU measurements in between. The above method ensures that data input to the model contains original noise. It should be noted that propagated pose sequences with noise are only used as the model's input, while ground truth poses are still used to calculate loss function. At the same time, to further ensure the universality of the real scene, we add an experiment in which both state estimation of ORB-SLAM3 [23] and simulated data are used to train and only estimated states are used to test.

5.1.4 Implementation details

Our model is implemented by PyTorch [53] on an NVIDIA 1080Ti GPU and Intel Xeon E5-2695 v4 CPUs, using Adam [54] optimizer with β_1 =0.9, β_2 = 0.99. The hyperparameters of MOUNT

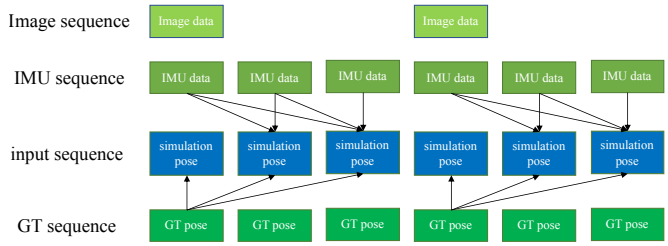


Fig. 6: Simulation process to meet the real situation. For sequences input to the model, only ground truth poses synchronized with image data will be retained, and remaining poses are replaced by IMU propagation from the latest vision pose using IMU measurements in between, which is consistent with the real situation.

are obtained by hyperparameter search. The input sequence length is set to 30, and the input sequences are first passed through the linear layer to become a 70 dimensions embedding and then encoded by the LSTM. The amount of parameters is on the order of 10^5, and for each input sequence of length 30, FLOPS is on the order of 10^6. All parts of the network are initialized with kaiming uniform [55]. The network is trained with a batch size of 1024, 100 epochs in total. The initial learning rate is set to 3×10^{-4} and drops to 0.3 times if the performance does not increase in 3 epochs. Each group of experiments runs 5 times under different random seeds, and the average result will be reported.

5.2 Results

In the following series of experiments, two variables will be further explored: **prediction time** and **camera frame rate**.

- The prediction time represents the time difference between the last input pose and the predicted pose. This variable can be used to test how the prediction ability changes over prediction time, which is also helpful to explore the source of the prediction ability of MOUNT.
- The camera frame rate represents the frame rate of the camera's recorded images and is also the frequency of VI-SLAM. This variable can be used to test the model's ability to deal with noisy inputs, which helps us explore the impact of uncertainty encoding on the model.

Therefore, we first show the performance of our model under basic settings and then show the performance changes of our model for the above two variables. Finally, the validity of the uncertainty prediction and user study will be discussed. In terms of speed, our model can run at 150fps on the CPU and 400fps on the GPU, proving its applicability for real-time prediction on AR devices.

5.2.1 Performance compared with baselines

According to the medium rendering delay time and camera frame rate, we test the performance of our model at the prediction time of 60ms and camera frame rate of 20Hz (EuRoC [5]) or 30Hz (our collected dataset). Specifically, for each pose at time t, we use IMU measurements and VI-SLAM output from $t-0.06-(t_n-t_1)$ s to $t-0.06$ s to predict it, where t_i represents the time at pose i. "w/o predict" represents the performance of not using motion prediction methods and rendering according to the current pose, which indicates the average motion of during the prediction time. The other methods are described in section 3.3.

As shown in Table 2, MOUNT can reduce the translation error by about 95% and the rotation error by 80% compared to "w/o predict" on the EuRoC [5] dataset. Compared with the best of other methods, MOUNT's translation and rotation errors can also be reduced by about 20% and 30%. At the same time, from the metric NF, the trajectory predicted by MOUNT is generally better than other methods in smoothness. Only in a relatively simple scene "V2_01_easy", MOUNT is as smooth as the EKF, which is because the EKF algorithm also has a certain smooth effect.

Since the dataset collected by drones doesn't obtain general characteristics of human motion, the performance shown in our collected dataset is more representative. Experimental results on our collected dataset are shown in Table 3. It can be seen that in the real AR motion dataset, MOUNT also has a strong prediction ability, which can greatly reduce the prediction error. Due to the unique characteristics of human motion, MOUNT performs better on our collected dataset than on EuRoC [5], thus bringing a greater improvement in AR experience. Compared with the best of other methods, MOUNT can reduce the translation error by about 35% and the rotation error by 60%, which shows that MOUNT does learn human motion patterns and applies them to prediction.

We further explore the performance of MOUNT according to the classification of environment and motion purpose. As shown in Table 4, under different environments and motion purposes, characteristics of motion are quite different. For example, motion in "far" environments is significantly faster than in "near" or "narrow" environments, because narrow environments are not conducive to fast motion, while the motion for "inspect" has slower translation and faster rotation, which is also in line with characteristics of in situ observation. It can be seen from Table 4 that MOUNT performs better than the methods based on extrapolation or EKF in any environment and motion purpose. In the mean time, MOUNT can at least reduce the translation error by more than 91% and the rotation error by 75% regardless of environments and motion purposes compared with w/o predict, which shows the generality of the algorithm. Obviously, the traditional methods including extrapolation and EKF depend on the current sensor data and certain motion model to obtain motion prediction, without considering the complex motion characteristics of people in different AR scenarios, such as the periodicity and tendency of human motion. With the prior information about characteristic of human behavior, MOUNT is obviously better than the traditional methods in the prediction ability.

It should be noted that even with the state estimates of ORB-SLAM3 [23] as input, MOUNT is still able to make relatively accurate predictions, with only a 10% increase in error compared to using simulated data. Compared with traditional methods, MOUNT still has a great advantage in accuracy. On the one hand, modern VI-SLAM system can make very accurate state estimation in a short time, on the other hand, MOUNT can effectively model the uncertainty, so as to reduce the influence of noisy input.

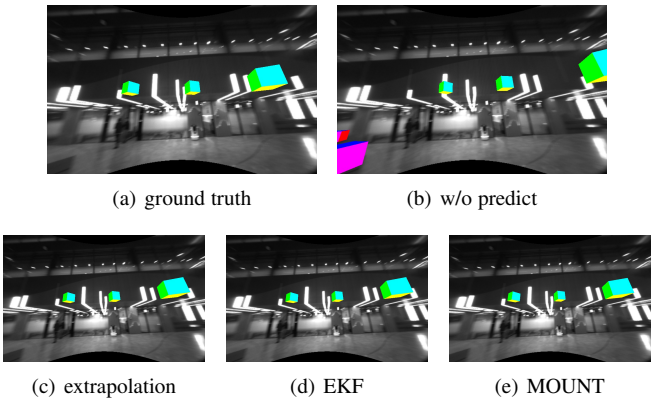
Figure 7 shows a typical comparison of images rendered with different prediction methods, the viewpoint in the image is being rotated, thus causing a significant rotation error in w/o predict method. Taking the lamp in the background as a reference, the prediction result of MOUNT is significantly more accurate than other methods and w/o predict. The full AR effects in different conditions are shown in the supplementary video.

5.2.2 Accuracy about prediction time

Since the traditional method based on a hand-crafted motion model can only make predictions in a short time due to its mathematical principles. Therefore, we should pay attention to our

TABLE 2: The performance of MOUNT. Prediction time: 60ms, camera frame rate: 20Hz, dataset: EuRoC [5]. O3 represents using state estimates of ORB-SLAM3 [23] as input. The other experiments use simulated data as input.

Sequence	method	AE_T(cm)	$AE_R(°)$	NF_T	NF_R
V2_01 _easy	w/o predict	1.957	0.8630	7.421	8.359
	extrapolation	0.6825	0.3010	24.92	8.520
	EKF	0.1225	0.2838	**3.310**	6.891
	MOUNT	**0.1087**	**0.2258**	3.428	**6.300**
	MOUNT(O3)	0.1154	0.2527	3.749	7.021
V2_02 _medium	w/o predict	4.331	1.975	16.39	15.88
	extrapolation	0.5913	0.4866	22.05	13.25
	EKF	0.2543	0.5174	3.228	10.98
	MOUNT	**0.1835**	**0.3469**	**2.920**	**9.162**
	MOUNT(O3)	0.2021	0.3756	3.012	9.421
V2_03 _difficult	w/o predict	4.504	2.222	17.78	17.91
	extrapolation	0.5757	0.5727	22.24	14.19
	EKF	0.2547	0.5880	3.746	12.10
	MOUNT	**0.1959**	**0.4043**	**3.596**	**11.24**
	MOUNT(O3)	0.2212	0.4512	3.723	11.85
whole	w/o predict	3.612	1.694	13.86	13.86
	extrapolation	0.6158	0.4547	23.07	11.98
	EKF	0.2112	0.4646	3.428	9.993
	MOUNT	**0.1631**	**0.3265**	**3.314**	**8.903**
	MOUNT(O3)	0.1775	0.3550	3.343	9.257



(a) ground truth (b) w/o predict

(c) extrapolation (d) EKF (e) MOUNT

Fig. 7: A typical example of an image (454th image in inspect_far) rendered under different prediction methods.

model's ability to different prediction time. We test performances of the above models at future 10ms, 30 ms, 60 ms, and 90 ms.

Results are indicated in Table 5. As the prediction time increases, the error of the no motion prediction increases linearly. Although the error of the predicted pose of our proposed MOUNT is also increasing, the improvement of MOUNT relative to the best of other methods is in an increasing trend, which shows that MOUNT has the ability to predict the pose after a long time. Even when the prediction time is 90ms, MOUNT can still produce prediction results with millimetre accuracy. Compared with other methods, it can reduce the translation error by 54% and the rotation error by 45%.

It can also be seen that even the traditional methods perform well when the prediction time is very short (10ms), and MOUNT is not able to pull far behind them. This indicates that for devices with high hardware performance and low latency such as HoloLens, the prediction algorithm with high performance has little effect, while for devices with low performance and high latency and cloud devices, the prediction algorithm will play a more important role.

The Figure 8 shows two typical examples. The turning point of the trajectory predicted by MOUNT is significantly earlier than other methods, which shows that MOUNT has a strong ability to predict motion tendency in advance. It can be seen from the above result that MOUNT has learned motion patterns contained in training data, which explains to a certain extent the reason why the model's predictions are so accurate.

TABLE 3: The performance of MOUNT. Prediction time: 60ms, camera frame rate: 30Hz, dataset: ours.

Sequence	method	AE_T(cm)	$AE_R(°)$	NF_T	NF_R
focus_far	w/o predict	5.800	1.689	10.25	8.221
	extrapolation	0.2665	1.055	5.870	9.241
	EKF	0.3327	0.8277	2.012	6.300
	MOUNT	**0.1752**	**0.4079**	**1.551**	**4.992**
focus_near	w/o predict	4.719	1.182	9.488	4.256
	extrapolation	0.1963	0.8179	4.034	5.985
	EKF	0.2473	0.6188	1.174	3.832
	MOUNT	**0.1380**	**0.3009**	**0.9353**	**2.981**
forward_far	w/o predict	5.071	2.142	12.56	11.67
	extrapolation	0.2498	0.8996	4.052	8.894
	EKF	0.3062	0.7812	**1.982**	6.601
	MOUNT	**0.1689**	**0.3404**	1.983	**4.938**
forward_narrow	w/o predict	4.912	1.902	11.29	9.194
	extrapolation	0.2418	0.9696	7.767	6.894
	EKF	0.2743	0.8505	**4.542**	5.225
	MOUNT	**0.1514**	**0.2894**	7.480	**3.202**
forward_near	w/o predict	4.550	1.257	10.72	5.295
	extrapolation	0.1359	0.7364	2.326	5.729
	EKF	0.2415	0.5433	1.129	3.849
	MOUNT	**0.1287**	**0.2015**	**0.9437**	**2.097**
forward_stair	w/o predict	4.093	2.479	7.943	10.53
	extrapolation	0.261	1.026	5.382	8.053
	EKF	0.4476	0.8830	2.005	5.542
	MOUNT	**0.2167**	**0.3894**	**1.643**	**4.615**
inspect_far	w/o predict	1.068	3.391	2.540	8.821
	extrapolation	0.1468	0.3378	0.7600	2.253
	EKF	0.1628	0.4366	0.6892	2.617
	MOUNT	**0.0946**	**0.1772**	**0.6610**	**1.899**
inspect_narrow	w/o predict	0.6400	1.575	2.546	5.995
	extrapolation	0.05891	0.2619	0.4909	2.267
	EKF	0.05938	0.3097	0.4885	2.125
	MOUNT	**0.05890**	**0.1542**	**0.3879**	**1.748**
inspect_near	w/o predict	2.211	3.166	6.810	13.79
	extrapolation	0.3489	0.5891	3.734	6.669
	EKF	0.3341	0.5807	**2.456**	5.021
	MOUNT	**0.1825**	**0.2688**	2.715	**4.219**
patrol_far	w/o predict	4.710	3.596	10.29	13.64
	extrapolation	0.3517	1.097	9.089	12.24
	EKF	0.3576	0.9815	**2.557**	7.941
	MOUNT	**0.2167**	**0.4159**	2.852	**5.789**
patrol_narrow	w/o predict	5.479	4.397	6.962	18.62
	extrapolation	0.2351	1.103	4.977	9.610
	EKF	0.3343	1.078	**1.730**	7.198
	MOUNT	**0.1750**	**0.4598**	1.808	**5.737**
patrol_near	w/o predict	5.024	2.285	8.845	8.522
	extrapolation	0.1468	0.5651	3.297	4.659
	EKF	0.1928	0.4836	1.289	3.191
	MOUNT	**0.1034**	**0.2174**	**1.018**	**2.773**
patrol_stair	w/o predict	4.859	3.533	11.49	15.34
	extrapolation	0.3104	1.184	7.259	10.08
	EKF	0.4157	1.103	2.345	7.424
	MOUNT	**0.2219**	**0.4850**	**2.144**	**5.951**
run_far	w/o predict	10.83	3.933	21.58	17.18
	extrapolation	0.7962	2.364	23.22	19.67
	EKF	0.9975	1.924	5.859	12.85
	MOUNT	**0.4780**	**0.7158**	**3.301**	**7.753**
whole	w/o predict	4.557	2.700	9.514	10.78
	extrapolation	0.2737	0.9235	5.876	8.019
	EKF	0.3370	0.8161	2.155	5.693
	MOUNT	**0.1806**	**0.3473**	**2.101**	**4.192**

TABLE 4: The performance of MOUNT in different environments and motion purposes. Prediction time: 60ms, camera frame rate: 30Hz, dataset: ours. The $*$ symbol stands for any.

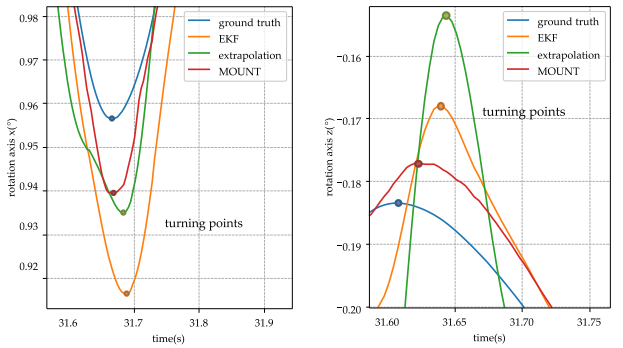
environment	purpose	w/o predict AE_T(cm)	w/o predict $AE_R(°)$	extrapolation AE_T(cm)	extrapolation $AE_R(°)$	EKF AE_T(cm)	EKF $AE_R(°)$	MOUNT AE_T(cm)	MOUNT $AE_R(°)$
narrow	$*$	3.909	2.813	0.1841	0.8196	0.2365	0.7865	**0.1343**	**0.3191**
near	$*$	3.774	2.291	0.2324	0.6421	0.2657	0.5525	**0.1445**	**0.2470**
far	$*$	5.615	2.844	0.3561	1.1532	0.4224	0.9871	**0.2243**	**0.4139**
stair	$*$	4.523	3.071	0.2888	1.1145	0.4297	1.0067	**0.2196**	**0.4431**
$*$	forward	4.734	1.984	0.2282	0.9097	0.3153	0.7716	**0.1669**	**0.3118**
$*$	inspect	1.569	2.781	0.2299	0.4528	0.2273	0.4801	**0.1323**	**0.2204**
$*$	patrol	5.026	3.450	0.2587	0.9807	0.3221	0.9056	**0.1777**	**0.3917**
$*$	run	10.83	3.933	0.7962	2.3644	0.9975	1.9237	**0.4780**	**0.7158**
$*$	focus	5.369	1.486	0.2385	0.9602	0.2986	0.7444	**0.1603**	**0.3652**

5.2.3 *Accuracy about camera frame rate*

Since MOUNT quantitatively measures the uncertainty of input data and output prediction, it should have a certain degree of

TABLE 5: The accuracy of our model about different prediction times. Prediction time: changeable, camera frame rate: 30Hz, dataset: ours. The last row of the table shows our method's error reduction percentage relative to the best of other methods.

time	10ms		30ms		60ms		90ms	
	AE_T(cm)	$AE_R(^\circ)$	AE_T(cm)	$AE_R(^\circ)$	AE_T(cm)	$AE_R(^\circ)$	AE_T(cm)	$AE_R(^\circ)$
w/o predict	0.7618	0.4565	2.279	1.3620	4.557	2.700	6.830	4.004
extrapolation	0.03499	0.03181	0.08230	0.3058	0.2737	0.9235	0.7111	1.872
EKF	0.03741	0.07065	0.1117	0.2885	0.3370	0.8161	0.6907	1.525
MOUNT	**0.02937**	**0.03121**	**0.06947**	**0.09698**	**0.1806**	**0.3473**	**0.3161**	**0.8328**
ratio	**16.06%**	**1.886%**	**15.58%**	**33.61%**	**34.01%**	**57.44%**	**54.23%**	**45.30%**



(a) (b)

Fig. 8: Two partial enlarged views of trajectories after extending prediction time. Prediction time: 60ms, camera frame rate: 30Hz, shown sequence: patrol_narrow. The dot in the figure represents the turning point of the trajectory, which illustrates the change in the motion trend.

ability to deal with noise. Therefore, we investigate the model's prediction ability at different camera frame rates.

As mentioned in section 5.1.3, the input of our model is not ground truth but the simulated VI-SLAM output sequence. In VI-SLAM, due to the presence of noise in the IMU, as the camera frame rate drops, the IMU propagation error will be accumulated, thus causing the increasing error in the input pose sequence. In this section, we show the robustness of the model to noise by testing the model's motion prediction performance under different camera frame rates.

As shown in Table 6 , when the camera frame rate decreases, the performance of our model also decreases due to the increase in the noise of the input sequence, but it is always much more accurate than no motion prediction and other methods.

TABLE 6: The performance of our model about camera frame rate. Prediction time: 60ms, camera frame rate: changeable, dataset: ours. The last row of the table shows the percentage of error reduction of our method relative to the best of other methods.

camera frame rate	30Hz		15Hz		10Hz		6Hz	
	AE_T(cm)	$AE_R(^\circ)$	AE_T(cm)	$AE_R(^\circ)$	AE_T(cm)	$AE_R(^\circ)$	AE_T(cm)	$AE_R(^\circ)$
w/o predict	4.557	2.700	4.557	2.700	4.557	2.700	4.557	2.700
extrapolation	0.2737	0.9235	0.2964	0.9246	0.3235	0.9260	0.3910	0.9281
EKF	0.3370	0.8161	0.3541	0.8173	0.3753	0.8184	0.4318	0.8204
MOUNT	**0.1806**	**0.3473**	**0.1940**	**0.3614**	**0.2225**	**0.3618**	**0.3044**	**0.3417**
ratio	**34.01%**	**57.44%**	**34.5%**	**55.78%**	**30.44%**	**55.79%**	**22.14%**	**58.34%**

Figure 9 shows a typical example when the camera frame rate is reduced to 10 Hz. It can be seen that the error between the predicted trajectory and the real trajectory has apparent periodicity, which is due to the drift of the input pose caused by the noise contained in the IMU. However, the robustness of our model makes it possible to produce more accurate poses compared with other methods.

More surprisingly, we found that 10Hz is a good choice that balances the prediction accuracy and the power consumption of AR devices. At this camera frame rate, the error of the motion prediction will be lower than other methods of 30Hz. However, the power consumption of the camera and VI-SLAM will be reduced to about 35% of the latter, which will significantly improve the AR device's endurance.

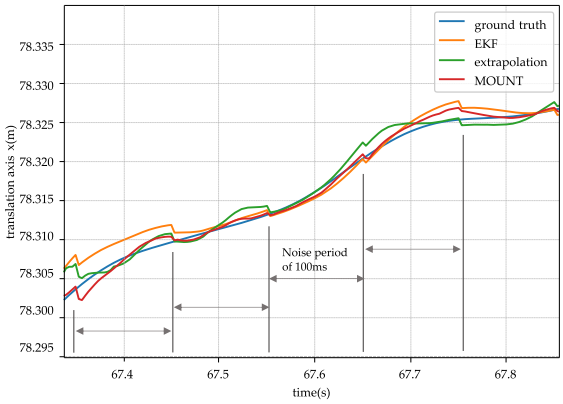


Fig. 9: Partial enlarged view of trajectories after reducing camera frame rate. Prediction time: 60ms, camera frame rate: 10Hz, shown sequence: run_far. The error of the predicted trajectory has an apparent periodicity of 100ms because of the periodicity of the noise in the input pose sequence.

5.2.4 *Uncertainty prediction*

We test the Bayesian model proposed in section 4.3 to evaluate its ability for uncertainty prediction of current output.

Figure 10 illustrates the relationship between the uncertainty prediction of the model's output and the actual prediction error. It can be seen from the top half of the two subplots that there is a visual positive correlation between the two variables, which shows that the model has a certain uncertainty prediction ability.

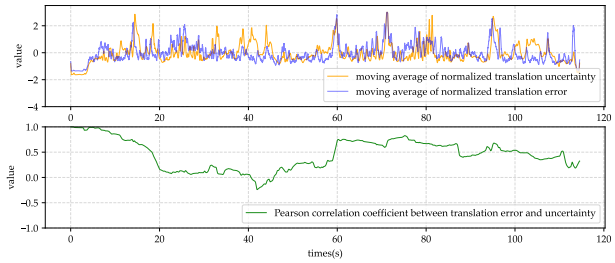
To further quantitatively measure the correlation between uncertainty and error, we use Pearson correlation analysis to calculate the correlation coefficient between the two time series. The bottom half of the two subplots show the two series' correlation within a time window. It can be seen that although the correlation fluctuates with time, it is positive most of the time. The overall translation and rotation correlation coefficients of this scene are 0.486 and 0.661, which shows that there is indeed a close correlation between the uncertainty of the MOUNT output and the actual error, which proves the ability of MOUNT to predict the uncertainty.

It should be noted that a small uncertainty usually means a small error, while a significant uncertainty does not necessarily mean a large error. When the uncertainty increases, the expected value of the prediction error will increase, and the variance will be more significant.

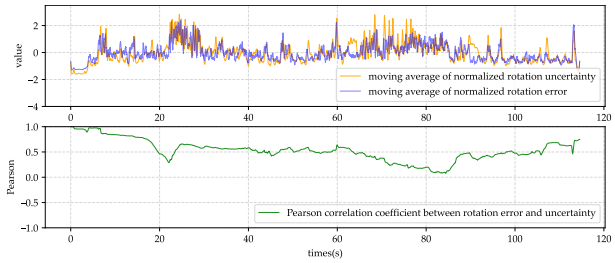
5.2.5 *User Study*

We conduct a user study to evaluate the perceptual quality of our proposed MOUNT in terms of accuracy and smoothness. More specifically, we conduct an experiment with 30 people who watch some AR videos produced by various methods and rate the accuracy of the virtual-real fusion and the smoothness of the position of virtual objects in the video.

Figure 11 shows the results of the user perceptual experiments. It can be found that although the result of MOUNT is still far from that of ground truth, it is obviously superior to the traditional method and can bring better feelings to users.



(a) translation



(b) rotation

Fig. 10: The Pearson correlation between uncertainties and errors. Prediction time: 60ms, camera frame rate: 20Hz, shown sequence: V2_03_difficult.

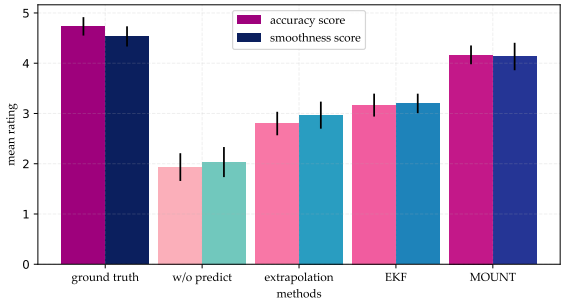


Fig. 11: Mean ratings from the user perceptual experiments with 95% comfidence intervals.

5.3 Ablation Study

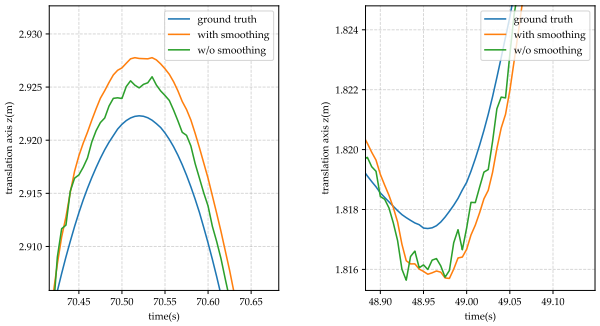
5.3.1 Ablation study on model components

To study the effectiveness of each part of the model, we divide the composition of the model into six parts:

- A: value encoding structure, this part of the network splices the latest poses and IMU measurements values to form the intermediate vector.
- B: motion pattern encoding structure, this part of the network is responsible for extracting the motion pattern information contained in the pose sequence.
- C: uncertainty pattern encoding structure, this part of the network is responsible for encoding the pose uncertainty sequence
- D: IMU pattern encoding structure, this part extracts the motion pattern information contained in the IMU sequence.
- E: uncertainty decoding structure, this part of the network is responsible for predicting the uncertainty of this output. Without this part, the loss function is changed to Equation 21
- F: smoothing structure, this part represents the smoothing strategy shown in section 4.4, which takes the responsibility of making the predicted trajectory smoother.

TABLE 7: Ablation experiment for network structure. Prediction time: 60ms, camera frame rate: 30Hz. dataset: ours

structure						Accuracy		Smoothness	
A	B	C	D	E	F	AE_T(cm)	$AE_R(°)$	NF_T	NF_E
√						0.2572	0.4221	10.68	13.24
√	√					0.2298	0.4059	8.120	10.49
√	√	√				0.2047	0.3862	6.176	9.198
√	√	√	√			0.1952	0.3612	3.842	6.286
√	√	√	√	√		**0.1725**	**0.3340**	3.054	5.073
√	√	√	√	√	√	0.1806	0.3473	**2.101**	**4.192**



(a) (b)

Fig. 12: Two Partial enlarged views of trajectories before and after smoothing strategy is used. Prediction time: 60ms, camera frame rate: 20Hz, shown sequence: V1_01_easy. After adding a smoothing strategy, the smoothness of the trajectory is greatly improved.

Table 7 shows changes in model performance after gradually adding the above model structure. It can be seen that part A,B,C,D and E of the model contribute to the final accuracy and smoothness to a certain extent, which shows the rationality of the model design.

It should be noted that after adding the smoothing strategy(part F), the accuracy has slightly declined by about 4%, for the reason that historical predictions are indeed not as accurate as this prediction for current prediction in a statistical sense. However, smoothness is greatly improved due to the addition of the smoothing strategy. It can be seen from Figure 12 that compared to the original output, the filtered prediction curve appears much smoother, which will significantly improve AR visual effects.

5.3.2 Ablation study on data form

To study the effectiveness of the data pre-processing methods in MOUNT, we experiment the effect of different forms of input and output data on the final performance. For the rotation component, it has three forms worth considering: rotation vector, euler angle and quaternion. At the same time, both rotation and translation can be processed into absolute and relative quantities. The absolute quantity represents the value itself. The calculation method of the relative quantity has been mentioned in Equation 7. For the IMU, we also try not to eliminate gravity.

The Table 8 shows the model performance under different data forms. As mentioned above, the relative quantity guarantees numerical stability and is easier to be learned by the model. Because of the singularity of euler angle and additional constraints of quaternion, the rotation vector becomes to be the best choice. The conclusions in [56] also provide us with strong evidence: "a rotation vector representation might also be suitable when restricted to small angles." The removal of gravity can significantly

reduce the learning burden of the network, so It can be seen that the data form we have chosen is reasonable and valid.

TABLE 8: Ablation experiment for data form. Prediction time: 60ms, camera frame rate: 30Hz, dataset: ours. ABS means absolute, REL means relative, and RAW means unprocessed form. The $\surd$ indicates that the data form here is the same as our method.

rotation form	translation form	IMU form	AE_T(cm)	$AE_R(^\circ)$
ABS rotation vector	$\surd$	$\surd$	0.1883	3.354
ABS quaternion	$\surd$	$\surd$	0.2132	16.27
ABS euler angle	$\surd$	$\surd$	0.1985	2.628
REL quaternion	$\surd$	$\surd$	0.1898	1.192
REL euler angle	$\surd$	$\surd$	0.1867	0.3523
$\surd$	ABS 3D vector	$\surd$	12.97	0.3498
$\surd$	$\surd$	RAW	0.3149	0.3482
$\surd$	$\surd$	$\surd$	**0.1806**	**0.3473**

6 Conclusion and Future Work

In this paper, we define a task called 6DoF motion prediction customized for delayed AR rendering mathematically, and show effective methods MOUNT for this task. Experiments on EuRoC [5] and our collected dataset show that MOUNT has the ability to demonstrate competitive performance on accuracy and smoothness and significantly improve AR visual effects. Furthermore, the experiments of MOUNT under the condition of low camera frame rate and long prediction time provide an explanation for its strong prediction ability to a certain extent.

Currently the prediction time is fixed. It is sufficient to verify the effectiveness of this method. However in practice, variable prediction time is required due to the fluctuations of actual delay time. So we plan to adapt MOUNT to variable prediction time in the future, and deploy it on the low-end device which has severe rendering delay, and also deploy it on the cloud to render the high-quality content. In addition, for the low-end device, we also plan to simplify the network without sacrificing prediction accuracy to further improve computational efficiency and reduce power consumption.

References

[1] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[2] Z. Oufqir, A. El Abderrahmani, and K. Satori, "ARKit and ARCore in serve to augmented reality," in *International Conference on Intelligent Systems and Computer Vision*. IEEE, 2020, pp. 1–7.

[3] P. Nowacki and M. Woda, "Capabilities of ARCore and ARKit platforms for ar/vr applications," in *International Conference on Dependability and Complex Systems*. Springer, 2019, pp. 358–370.

[4] J. Suto, S. Oniga, and P. P. Sitar, "Feature analysis to human activity recognition," *International Journal of Computers Communications & Control*, vol. 12, no. 1, pp. 116–130, 2016.

[5] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[6] J. Van Waveren, "The asynchronous time warp for virtual reality on consumer hardware," in *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, 2016, pp. 37–46.

[7] U. H. List, "Nonlinear prediction of head movements for helmet-mounted displays," AIR FORCE HUMAN RESOURCES LAB BROOKS AFB TX, Tech. Rep., 1983.

[8] R. Azuma and G. Bishop, "A frequency-domain analysis of head-motion prediction," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 401–408.

[9] R. H. So and M. J. Griffin, "Effects of time delays on head tracking performance and the benefits of lag compensation by image deflection," in *Flight Simulation Technologies Conference, New Orleans, Louisiana*, 1991.

[10] T. Mazuryk and M. Gervautz, "Two-step prediction and image deflection for exact head tracking in virtual environments," in *Computer Graphics Forum*, vol. 14, no. 3. Wiley Online Library, 1995, pp. 29–41.

[11] J. Rambach, A. Pagani, S. Lampe, R. Reiser, M. Pancholi, and D. Stricker, "[poster] fusion of unsynchronized optical tracker and inertial sensor in ekf framework for in-car augmented reality delay reduction," in *International Symposium on Mixed and Augmented Reality Adjunct*. IEEE, 2017, pp. 109–114.

[12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[13] M. Borges, A. Symington, B. Coltin, T. Smith, and R. Ventura, "HTC vive: Analysis and accuracy improvement," in *Proc. of International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 2610–2615.

[14] E. M. Foxlin, "Generalized architecture for simultaneous localization, auto-calibration, and map-building," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1. IEEE, 2002, pp. 527–533.

[15] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[16] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[18] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[19] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. of Internatoinal Conference on Computer Vision*. IEEE, 2011, pp. 2564–2571.

[20] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[21] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. of European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

[22] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[23] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, 2021.

[24] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proc. of International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[25] K. Wu, A. Ahmed, G. A. Georgiou, and S. I. Roumeliotis, "A square root inverse filter for efficient vision-aided inertial navigation on mobile devices." in *Robotics: Science and Systems*, vol. 2, 2015, p. 2.

[26] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *Journal of intelligent & robotic systems*, vol. 61, no. 1, pp. 287–299, 2011.

[27] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[28] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[29] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation." Georgia Institute of Technology, 2015.

[30] S. Agarwal and K. Mierle, "Ceres solver: Tutorial & reference," *Google Inc*, vol. 2, no. 72, p. 8, 2012.

[31] J. R. Rambach, A. Tewari, A. Pagani, and D. Stricker, "Learning to fuse: A deep learning approach to visual-inertial camera pose estimation," in *International Symposium on Mixed and Augmented Reality*. IEEE, 2016, pp. 71–76.

[32] N. Radwan, A. Valada, and W. Burgard, "VLocNet++: Deep multitask learning for semantic visual localization and odometry," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4407–4414, 2018.

[33] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[34] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "ICE-BA: Incremental, consistent and efficient bundle adjustment for visual-inertial SLAM," in *Proc. of Computer Vision and Pattern Recognition*, 2018, pp. 1974–1982.

[35] J. Weberruss, L. Kleeman, D. Boland, and T. Drummond, "FPGA acceleration of multilevel ORB feature extraction for computer vision," in *Proc. of International Conference on Field-Programmable Logic and Applications*. IEEE, 2017, pp. 1–8.

[36] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1106–1119, 2019.

[37] Z. Zhang, A. A. Suleiman, L. Carlone, V. Sze, and S. Karaman, "Visual-inertial odometry on chip: An algorithm-and-hardware co-design approach," 2017.

[38] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1281–1293, 2013.

[39] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, vol. 25, no. 6, pp. 38–46, 2005.

[40] S. Rajagopal, "Personal dead reckoning system with shoe mounted inertial sensors," *Master's Degree Project, Stockholm, Sweden*, 2008.

[41] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 2013, pp. 225–234.

[42] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to cure the curse of drift in inertial odometry," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[43] S. Herath, H. Yan, and Y. Furukawa, "RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in *Proc. of International Conference on Robotics and Automation*. IEEE, 2020, pp. 3146–3152.

[44] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, "TLIO: Tight learned inertial odometry," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653–5660, 2020.

[45] A. B. Watson and A. Ahumada, *A look at motion in the frequency domain*. National Aeronautics and Space Administration, Ames Research Center, 1983, vol. 84352.

[46] S. Winograd, "On computing the discrete fourier transform," *Mathematics of computation*, vol. 32, no. 141, pp. 175–199, 1978.

[47] P. G. Weyand, R. F. Sandell, D. N. Prime, and M. W. Bundle, "The biological limits to running speed are imposed from the ground up," *Journal of applied physiology*, vol. 108, no. 4, pp. 950–961, 2010.

[48] M. E. Pittelkau, "Rotation vector in attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 6, pp. 855–860, 2003.

[49] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust imu double integration," in *Proc. of European Conference on Computer Vision*, 2018, pp. 621–636.

[50] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[51] C. Chen, X. Lu, J. Wahlstrom, A. Markham, and N. Trigoni, "Deep neural network based inertial odometry using low-cost inertial measurement units," *IEEE Transactions on Mobile Computing*, 2019.

[52] H. Liu, M. Jiang, Z. Zhang, X. Huang, L. Zhao, M. Hang, Y. Feng, H. Bao, and G. Zhang, "LSFB: A low-cost and scalable framework for building large-scale localization benchmark," in *International Symposium on Mixed and Augmented Reality Adjunct*. IEEE, 2020, pp. 219–224.

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[54] P. D. Kingma and L. J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2015.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. of Internatoinal Conference on Computer Vision*, 2015, pp. 1026–1034.

[56] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proc. of Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753.

Haoran Chen received bachelor's degree from Peking University in 2020 and is currently a graduate student at Peking University. His main research interests are in virtual reality, augmented reality and 3D vision.

Lantian Wei is a graduate student in Peking University. He is currently CEO in Holo Intelligence. His research interests include Multi-robot system, SLAM, and augmented reality.

Haomin Liu received the master and Ph.D. degrees in computer science from Zhejiang University in 2009 and 2017. He is currently research director in SenseTime Research. His research interests include Structure-from-Motion, SLAM, and augmented reality.

Boxin Shi received the BE degree from the Beijing University of Posts and Telecommunications, the ME degree from Peking University, and the PhD degree from the University of Tokyo, in 2007, 2010, and 2013. He is currently a Boya Young Fellow Assistant Professor and Research Professor at Peking University, where he leads the Camera Intelligence Lab. Before joining PKU, he did postdoctoral research with MIT Media Lab, Singapore University of Technology and Design, Nanyang Technological University from 2013 to 2016, and worked as a researcher in the National Institute of Advanced Industrial Science and Technology from 2016 to 2017. His papers were awarded as Best Paper Runner-Up at International Conference on Computational Photography 2015 and selected as Best Papers from ICCV 2015 for IJCV Special Issue. He has served as an editorial board member of IJCV and an area chair of CVPR/ICCV. He is a senior member of IEEE.

Guofeng Zhang is a Professor at State Key Lab of CAD&CG, Zhejiang University. He received his BS and Ph.D. degrees in Computer Science from Zhejiang University, in 2003 and 2009, respectively. He received the National Excellent Doctoral Dissertation Award, the Excellent Doctoral Dissertation Award of China Computer Federation and the best paper award of ISMAR 2020. His research interests include structure-from-motion, SLAM, 3D reconstruction, augmented reality, video segmentation, and editing.

Hongbin Zha received the B.E. degree from the Hefei University of Technology, China, in 1983 and the M.S. and Ph.D. degrees from Kyushu University, Japan, in 1987 and 1990, respectively. After working as a Research Associate in the Kyushu Institute of Technology, Japan, he joined Kyushu University, Japan, in 1991 as an Associate Professor. Since 2000, he has been a Professor at the Key Laboratory of Machine Perception (Ministry of Education), Peking University, China. His research interests include computer vision, digital geometry processing, and robotics. He has published more than 350 technical publications in journals, books, and international conference proceedings.