FollowUpAR: Enabling Follow-up Effects in Mobile AR Applications

Jingao Xu*, Guoxuan Chi*, Zheng Yang✉, Danyang Li, Qian Zhang, Qiang Ma, Xin Miao
School of Software and BNRist, Tsinghua University, China
{xujingao13,chiguoxuan,hmilyyz,lidanyang1919,qzhangqz123,tsinghuamq,lucas.x.miao}@gmail.com

ABSTRACT

Existing smartphone-based Augmented Reality (AR) systems are able to render virtual effects on static anchors. However, today's solutions lack the ability to render follow-up effects attached to moving anchors since they fail to track the 6 degrees of freedom (6-DoF) poses of them. We find an opportunity to accomplish the task by leveraging sensors capable of generating sparse point clouds on smartphones and fusing them with vision-based technologies. However, realizing this vision is non-trivial due to challenges in modeling radar error distributions and fusing heterogeneous sensor data. This study proposes FollowUpAR, a framework that integrates vision and sparse measurements to track object 6-DoF pose on smartphones. We derive a physical-level theoretical radar error distribution model based on an in-depth understanding of its hardware-level working principles and design a novel factor graph competent in fusing heterogeneous data. By doing so, FollowUpAR enables mobile devices to track anchor's pose accurately. We implement FollowUpAR on commodity smartphones and validate its performance with 800,000 frames in a total duration of 15 hours. The results show that FollowUpAR achieves a remarkable rotation tracking accuracy of 2.3° with a translation accuracy of 2.9*mm*, outperforming most existing tracking systems and comparable to state-of-the-art learning-based solutions. FollowUpAR can be integrated into ARCore and enable smartphones to render follow-up AR effects to moving objects.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing systems and tools**.

KEYWORDS

Augmented Reality; 6-DoF Pose Tracking; mmWave Radar; Computer Vision

ACM Reference Format:
Jingao Xu, Guoxuan Chi, Zheng Yang, Danyang Li, Qian Zhang, Qiang Ma, and Xin Miao. 2021. FollowUpAR: Enabling Follow-up Effects in Mobile AR Applications. In *The 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21), June 24–July 2, 2021, Virtual, WI, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3458864.3467675

✉ Zheng Yang is the corresponding author.
*Jingao Xu and Guoxuan Chi are co-primary authors.

MobiSys '21, June 24–July 2, 2021, Virtual, WI, USA

ACM ISBN 978-1-4503-8443-8/21/06…$15.00
https://doi.org/10.1145/3458864.3467675

1 INTRODUCTION

Augmented Reality (AR), particularly Mixed Reality (MR), enhances the real world by rendering virtual overlays on users' field of view through cameras, which promises to provide unprecedented immersive experiences in entertainment, education, and healthcare. Additionally, reports forecast that 99 million AR/VR devices will be shipped in 2021 [40], and the market will reach 108 billion dollars by then [41].

The well-known mobile AR solutions such as ARKit [24] and ARCore [8] enable a smartphone to recognize surfaces (i.e. sets of keypoints) as the anchors and allow users to pin virtual objects on them. They are able to further continuously track the pose[1] of the smartphone to adjust the locations of virtual effects in the user's view when the smartphone is moving with the anchors stationary. However, once the anchors move, they will fail to render follow-up effects on these moving targets since nowadays AR frameworks *can merely track the pose of smartphones rather than anchors.* Specifically, as illustrated in Fig. 1a, when a target dragon changes its pose, the virtual flame rendered by ARCore might lose its target. The awful experience is similar for the Rubik's Cube example. Ideally, an AR framework should render the virtual effects or measurements without misalignment. Therefore, pose tracking of anchors is key to further bridge the gap between virtuality and reality in mobile AR applications.

Existing vision-based 6-DoF tracking solutions include: 1. leveraging expensive and sophisticated sensors or devices (e.g., Microsoft HoloLens2 and Magic Leap One headsets are equipped with 4 different types of cameras including color, gray, depth, and IR [44]), however, these sensors are not concurrently available on commercial smartphones; 2. only using classical visual feature matching techniques to optimize object poses, but the cumulative drift due to scale ambiguity leads to virtual effects misalignment (Fig. 1b); and 3. deep-learning-based methods, however, these approaches require complex pre-modeling of specific objects and pre-training of neural networks in advance [45, 48]. They can neither track arbitrary objects (Fig. 1c) nor run on phones in real-time. In a nutshell, none of the previous solutions can be implemented on smartphones in ubiquitous scenarios due to limited sensors and resources onboard and the uncertainty of objects and environments.

Nowadays, sensors capable of generating sparse point clouds[2] are gradually integrated into commercial smartphones, e.g., Google

[1] The pose in this article represents the position and orientation of an object, which is a variable with six degree-of-freedom (6-DoF).
[2] A point cloud is a set of data points in space, each associated with a reflection location on the surface of a 3D object typically.

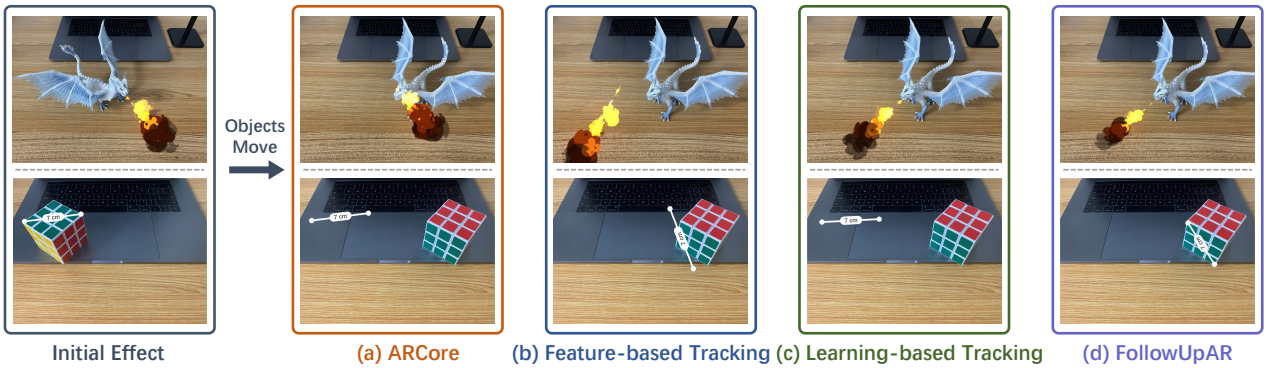


Figure 1: Running demo comparison of FollowUpAR and other solutions. FollowUpAR is capable of rendering follow-up AR effects on moving anchors. Compared with FollowUpAR, **(a) ARCore** is currently unable to track the pose of moving anchors, and classical visual **(b) Feature-matching-based** approaches have a large error due to scale ambiguity. **(c) Deep-learning-based** methods typically require complex pre-training of the target, hence cannot track arbitrary objects in real-time.

Pixel 4 and Huawei P30 pro. This inspires us to leverage the relative sparse measurements (e.g., mmWave radar) and fuse them with vision techniques to track 6-DoF pose of anchors continuously in real-time. Albeit inspiring, translating this intuition into a practical AR framework is non-trivial and faces significant challenges:

- **Radar errors are difficult to model.** Existing works generally assume that mmWave radar error follows a normal distribution and neglect the cause of errors [11, 22, 47]. In other words, there lacks a theoretical model of radar error distribution, leading to inexact optimization goals, and eventually, resulting in precision bottlenecks.
- **Heterogeneous data are difficult to fuse.** Unlike visual features in terms of precision, scale, and density, the sparse point clouds have a relatively low spatial resolution. To fuse such heterogeneous sensor data, previous fusion frameworks (e.g., extended Kalman filter [39], particle filter [51]) usually suffer from severe cumulative drift error, thus not competent in continuous high accuracy pose tracking tasks.

To tackle the above challenges, we design and implement FollowUpAR, the first framework that fuses mmWave radar and monocular vision to track the 6-DoF pose of anchors. FollowUpAR can be integrated into mobile AR frameworks and further enable them to render follow-up virtual effects in scenarios with moving anchors.

In FollowUpAR, firstly, to improve mmWave radar measurement accuracy, we dig deeper into error sources from its hardware design and working principles. On this basis, we derive *PRED* model, a **P**hysical-level theoretical model of **R**adar **E**rror **D**istribution, to benefit further optimization of the tracking result. Secondly, to perform high accuracy pose tracking, we leverage and modify a *factor graph* to fuse heterogeneous data in a tightly coupled manner. Specifically, we elaborately design the *factor nodes* and *variable nodes* in the factor graph for tracking anchor's 6-DoF pose continuously.

We have fully implemented FollowUpAR on Google Pixel 4, the running demo is shown in Fig. 1d. We also conduct extensive experiments with various object-camera distances and object moving speeds in two different scenarios (a pure laboratory and a crowded office). We further evaluate the robustness of FollowUpAR under conditions of diverse light intensity, object occlusions, and dynamic background motion. The entire experiment lasts around 15 hours, collecting 800,000 video frames as input. We compare FollowUpAR with three related works, including a classical visual feature-matching method (ICP [57]) and two state-of-the-art learning-based solutions (NOCS [46] and PoseRBPF [9]). The experiment results show that FollowUpAR achieves an average rotation accuracy of 2.3° and a translation accuracy of 2.9*mm*, outperforming ICP and NOCS by > 45%. The performance of FollowUpAR is comparable to PoseRBPF, however, only FollowUpAR can track arbitrary objects in a real-time manner on resource-limited smartphones and require no prior training.

In summary, the main contributions are as follows:

- We propose FollowUpAR, as far as we are aware of, the first framework that fuses sparse measurement and vision to track object 6-DoF pose. FollowUpAR can be integrated into mobile AR frameworks and facilitate smartphones to render follow-up AR effects on moving objects.
- We propose the *PRED* model to push the limit of mmWave radar measurement accuracy by exploring and revealing the cause of radar error from its hardware design and working principles.
- We design a novel *factor-graph*, which can fuse heterogeneous data with different spatial resolution, and achieve continuous 6-DoF pose tracking for moving objects.
- We implement and extensively evaluate FollowUpAR as well as 3 related works. The evaluation result shows the feasibility and effectiveness of this system to realize follow-up AR effects rendering for mobile AR scenarios.

The rest of this paper is organized as follows. We first present the overview of FollowUpAR in Section 2, followed by detailed descriptions of the sensor tracking model in Section 3 and fusion framework in Section 4. The implementation and evaluation of our system is shown in Section 5. We discuss the related work in Section 6 and conclude FollowUpAR in Section 7. We also attach several Appendix sections including detailed formula derivations of essential variables.

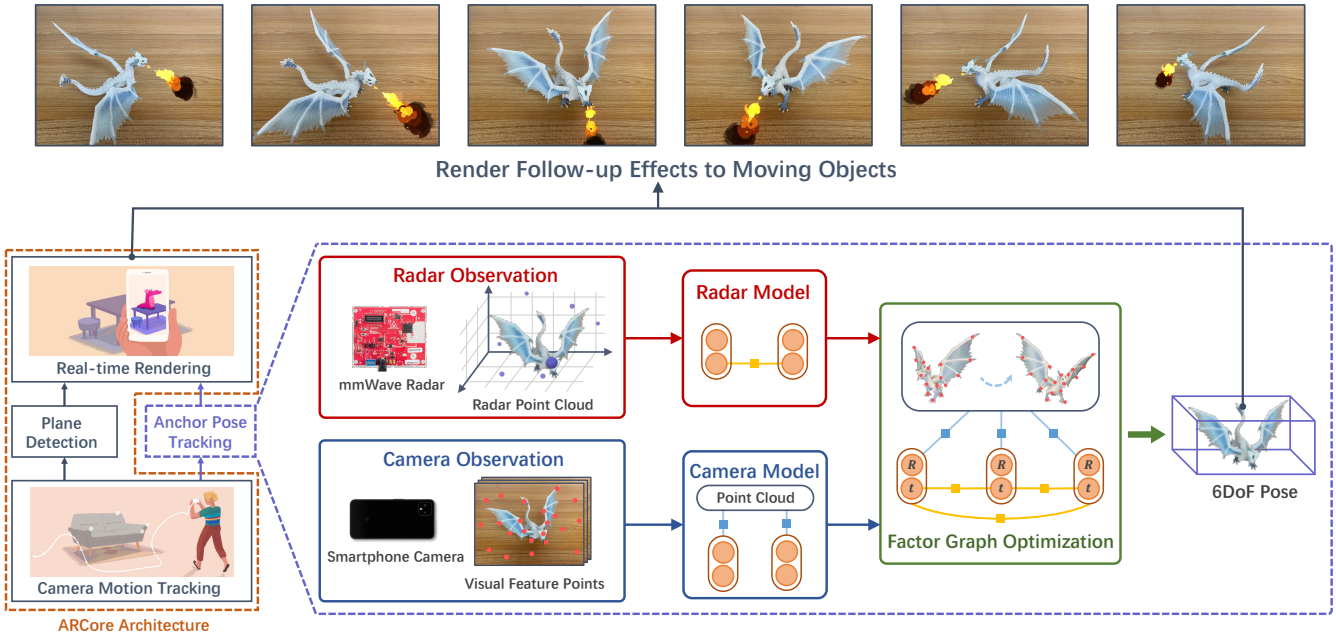


Figure 2: System architecture of FollowUpAR.

2 SYSTEM OVERVIEW

The architecture of FollowUpAR is illustrated in Fig. 2. As seen, FollowUpAR can be integrated into ARCore as an additional function module to continuously provide the essential information of the 6-DoF pose of a moving anchor. Eventually, FollowUpAR enables ARCore to render follow-up effects on it. We first briefly introduce the architecture of ARcore, followed by the description of FollowUpAR workflow.

2.1 ARCore Architecture

As shown in the left part of Fig. 2, there are three main capabilities in the current ARCore platform: camera motion tracking, plane detection, and real-time rendering. The *camera motion tracking* module allows the mobile devices to track their own rotation and translation relative to the world by leveraging VO [42] or VIO [37]. Further, the *plane detection* module allows the phone to detect the size and location of all types of surfaces (e.g., horizontal or vertical surfaces, like tables or walls) suitable for placing virtual objects. Finally, after a user clicks on the screen to place a virtual object, ARCore instantly renders the virtual effect, and adjust the rendering on the screen based on the pose reported by the camera motion tracking module.

However, today's ARCore requires the anchors to be static since ARCore only adjusts the renderings according to the camera pose rather than the anchor's pose. When the anchor moves, the rendered effects may be misaligned.

2.2 FollowUpAR Workflow

FollowUpAR aims at bridging the gap between virtuality and reality by providing ARCore with 6-DoF tracking capability. As shown in the right part of Fig. 2, FollowUpAR consists of three main components: *Radar Tracking Model*, *Camera Tracking Model*, and factor-graph-based *Pose Optimization*. Specifically, FollowUpAR first takes time-synchronized video stream and mmWave radar measurements of an anchor as inputs. The radar measurements are processed to generate a sparse 3D point cloud in *Radar Tracking Model*, while visual feature points on each frame are extracted and matched in *Camera Tracking Model*. Thereafter, based on the sensors' working principles and their measurement noise distributions, FollowUpAR derives objects' motion and pose estimations from radar measurements and visual features matching, respectively. Finally, the factor-graph-based *Pose Optimization* jointly fuses and optimizes the results from *Camera Tracking Model* as well as *Radar Tracking Model*, and then reports accurate 6-DoF pose of the anchor in a real-time manner.

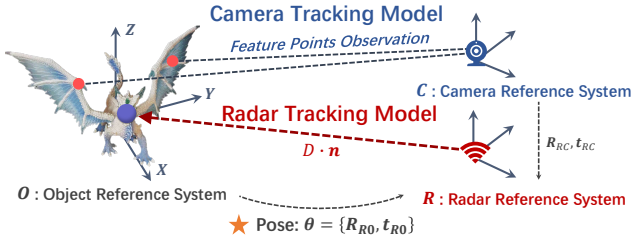
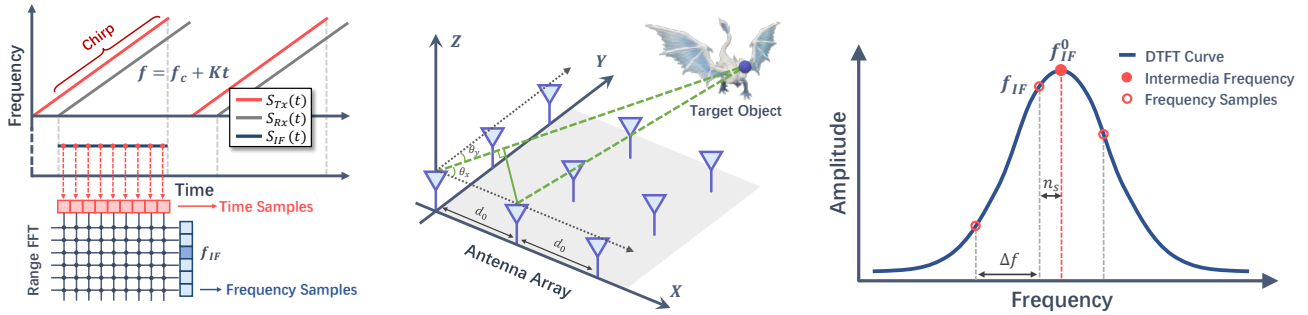


Figure 3: Illustration of reference systems and essential variables in FollowUpAR

3 SENSOR TRACKING MODEL

We first briefly describe and illustrate some essential variables in FollowUpAR. As shown in Fig. 3, there are three reference (a.k.a., coordinate) systems in FollowUpAR: the camera reference system C, the radar R and the anchor object O. Therein, C and R keep stationary,



(a) Measure Distance by Frequency Difference (b) Measure Angle by Phase Difference (c) Radar Sampling Error

Figure 4: Model and noise analysis of mmWave-radar-based object motion tracking

and they are rigidly attached on the smartphone, while O follows the motion of the object. We define the object's pose, the goal of FollowUpAR, as the transformation from coordinate system O to R. Specifically, FollowUpAR optimizes and reports object's 6-DoF pose $\boldsymbol{\theta}^i = \{\boldsymbol{R}^i_{\mathsf{RO}}, \boldsymbol{t}^i_{\mathsf{RO}}\}$ at each timestamp i, where $\boldsymbol{R}_{\mathsf{RO}}$ and $\boldsymbol{t}_{\mathsf{RO}}$ are rotation and translation from O to R[3], respectively. The transformation from C to R (i.e. $\boldsymbol{R}_{\mathsf{RC}}$ and $\boldsymbol{t}_{\mathsf{RC}}$) can be easily obtained from calibration [14].

As aforementioned, FollowUpAR contains two kinds of sensor tracking model: *Radar Tracking Model* and *Camera Tracking Model*, which operate simultaneously and estimate object's motion or pose, respectively. In this section, we first briefly describe and illustrate some essential coordinates and optimization variables in FollowUpAR, followed by a sequential presentation of these two tracking models. For each model, we will present two aspects: 1) how to derive the object's motion or pose from the input data (visual images or radar samples); and 2), equally importantly, where the tracking result's error comes from and how to accurately profile its distribution, which is the basis for further fusion and optimization of the system and influences the precision bottleneck.

3.1 Radar Tracking Model

3.1.1 Radar-based Motion Tracking. In this part, we describe how to calculate the distance D as well as its direction vector $\boldsymbol{n}$ from radar to the object leveraging mmWave frequency modulated continuous wave (FMCW) signals and, on the basis of this, resolve the translational component $\boldsymbol{t}_{\mathsf{RO}}$ of the object's pose $\boldsymbol{\theta}$.

Distance Calculation: As shown in Fig. 4a, the frequency difference between the *transmitted signal* (TX signal) and *received signal* (RX signal) reflects the signal propagation time, which further indicates the distance between the object and radar. Denote the time-variant distance between the radar and the object as $D(t)$, the transmitted and received signals can be expressed as:

$$\begin{aligned} S_{TX}(t) &= \exp[j(2\pi f_c t + \pi K t^2 + \phi_0)], \\ S_{RX}(t) &= \alpha S_{TX}[t - \frac{2D(t)}{c}], \end{aligned} \tag{1}$$

where α is the attenuation rate, f_c is the starting frequency, K is the chirp slope of FMCW signal, and c represents the speed of light. Both TX and RX signal will then be put into the mixer and low-pass filter LPF($\cdot$) to extract the *intermediate frequency signal* (IF signal) as:

$$S_{IF}(t) = \mathrm{LPF}[S^*_{TX}(t)S_{RX}(t)] \approx \alpha \exp[j2\pi(\frac{2KD(t)}{c})t], \tag{2}$$

whose frequency value f_{IF} contains the distance information. After applying the *Range-FFT* operation [10, 27] on the IF signal, f_{IF} will be extracted. Then the distance can be calculated as:

$$D = \frac{c f_{IF}}{2K}. \tag{3}$$

Direction Acquisition: With a well-designed antenna array, mmWave radar can also get the object's direction. As shown in Fig. 4b, the antennas are arranged as two orthogonal linear antenna arrays. Each of the linear array can acquire an AoA (i.e. the angle between the array direction and the RX signal direction) based on the phase difference between two adjacent antennas with spacing of d:

$$\cos\theta = \frac{\Delta\phi\lambda}{2\pi d}, \tag{4}$$

where θ is the AoA and λ is the wave length, and $\Delta\phi$ is the phase difference. Since the radar has two orthogonal linear antenna arrays, two different AoA θ_x and θ_y can be acquired. Then, the unit vector indicating the object's direction can be represented as:

$$\boldsymbol{n} = \left[\cos\theta_x \;\; \cos\theta_y \;\; \sqrt{1 - \cos^2\theta_x - \cos^2\theta_y}\right]^{\mathrm{T}}. \tag{5}$$

To sum up, with both the distance and angle calculations, the mmWave radar can get the object's 3D location $\boldsymbol{P}_{\mathsf{R}}$ in radar reference R as:

$$\boldsymbol{P}_{\mathsf{R}} = D\boldsymbol{n}. \tag{6}$$

FollowUpAR leverages mmWave radar to track target's 3D location and estimate the translation of the target $\boldsymbol{t}_{\mathsf{RO}}$ at each timestamp:

$$\begin{aligned} \boldsymbol{t}^j_{\mathsf{RO}} &= \boldsymbol{t}^i_{\mathsf{RO}} + \boldsymbol{U}^{ij}_{\mathsf{R}} + \boldsymbol{w}^{ij} \\ &= \boldsymbol{t}^i_{\mathsf{RO}} + \left(\boldsymbol{P}^j_{\mathsf{R}} - \boldsymbol{P}^i_{\mathsf{R}}\right) + \boldsymbol{w}^{ij}, \end{aligned} \tag{7}$$

where $\boldsymbol{U}^{ij}_{\mathsf{R}}$ is modeled as the difference between two radar calculation results at time i and j in the radar-reference. The $\boldsymbol{w}^{ij}$ represents the random error of the measurement noise. The analysis of the causes of error $\boldsymbol{w}^{ij}$, as well as modeling of its probability distribution determine the accuracy bottleneck of the system. Unlike previous works that neglected to explore it, in what follows, we will analyse the error sources of $\boldsymbol{w}^{ij}$ from mmWave radar's hardware design and working principles.

[3]We use the Lie algebraic [2] representation of object poses, where rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$ are both three-dimensional vectors

3.1.2 Radar Error Analysis from Physical-level. We systematically analyse the error distribution of the mmWave FMCW radar from a down-to-top perspective. In general, measurement errors can be roughly divided into two categories: 1) *intrinsic systematic errors*, which cause constant measurement biases; and 2) *extrinsic random errors*, which result in variances around the true value.

The *intrinsic systematic error* comes from the imperfect design of the mmWave radar device. When the TX signal is generated, it needs to pass through a certain waveguide length before being propagated into the air, and vice-versa for RX. Although the extra propagation distance costs additional time, the bias between the real distance and the calculated result is constant. Therefore, with a pre-calibration, subtracting the additional propagation distance from the calculated result is sufficient to correct it.

In contrast, the *extrinsic random errors* are difficult to address by trivial calibrations. Here we give several essential components of them:

1) Sampling Error. The mmWave radar measures the distance by extracting f_{IF} from the IF signal. However, as shown in Fig. 4c, there is a sampling error n_s due to the limited sampling rate, and the extracted discrete frequency follows a uniform distribution. Therefore, the measured distance D proportional to f_{IF} follows a uniform distribution.

2) Thermal Noise. In mmWave radar, the FMCW signals is generated by a voltage control oscillator (VCO) whose output frequency is proportional to the input voltage. The thermal noise in the circuit makes the input voltage follows a zero-mean normal distribution and further lead to an error of the frequency f_{IF}.

3) AWGN Channels. The FMCW signal is inevitably interfered with by an additive white Gaussian noise (AWGN) during propagation, which causes its phase to follow a uniform distribution. Since multiple antennas measurements average the phase difference, the resulted difference can be treated as a normal distribution.

We model each error source respectively, fuse them together, and eventually obtain *Physical-level Radar Error Distribution (PRED)*. The detailed derivation of relevant equations is presented in Appendix A. In brief, the probabilistic density function $f(\cdot)$ of *PRED* is formulated as follow:

$$\begin{aligned} f(e) &= \int_{-a}^{a} \frac{1}{2a} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(e-x)^2}{2\sigma^2}\right) dx \\ &= \frac{1}{2a}\left(\Phi\left(\frac{e+a}{\sigma}\right) - \Phi\left(\frac{e-a}{\sigma}\right)\right), \end{aligned} \tag{8}$$

where a and σ are parameters of the uniform distribution and normal distribution, respectively (can be easily obtained from pre-calibration). The $\Phi(\cdot)$ indicates the cumulative distribution function (CDF) of the standard normal distribution. Moreover, since $\boldsymbol{w}$ in Eqn. 7 follows a superposition of two independent *PRED* ($\boldsymbol{w}^{ij} = \boldsymbol{w}^j - \boldsymbol{w}^i$), the probabilistic density function of random variable $\boldsymbol{w}$ can be written as:

$$\Psi(e) = \int_{-\infty}^{\infty} f(x) f(e+x) dx. \tag{9}$$

3.2 Camera Tracking Model

In this section, we present how to estimate the target's pose $\boldsymbol{\theta}$ (both $\boldsymbol{R}_{\mathsf{RO}}$ and $\boldsymbol{t}_{\mathsf{RO}}$, i.e. rotation and translation) based on the visual feature matching results, as illustrated in Fig. 5.

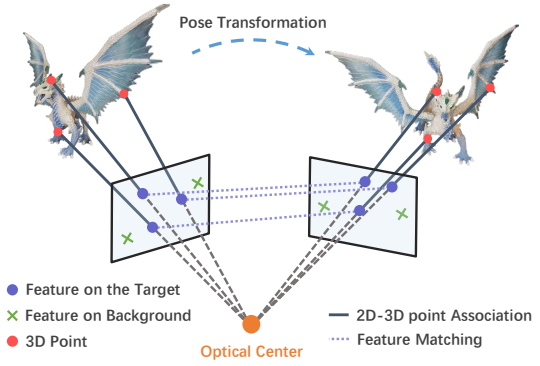


Figure 5: Illustration of vision-based object pose tracking

Vision-based Pose Tracking: We consider a conventional pinhole-camera model [12, 13, 18] with a projection function $\boldsymbol{\pi} : \mathbb{R}^3 \rightarrow \Omega$, which transforms a 3D point $\boldsymbol{X}_{\mathsf{C}}$ in camera reference C, into a 2D pixel $\boldsymbol{x}$ in image plane, where $\boldsymbol{x} \in \Omega \subset \mathbb{R}^2$. To be more specific:

$$\boldsymbol{\pi}(\boldsymbol{X}_{\mathsf{C}}) = \begin{bmatrix} f_x \frac{X_{\mathsf{C}}}{Z_{\mathsf{C}}} + c_x \\ f_y \frac{Y_{\mathsf{C}}}{Z_{\mathsf{C}}} + c_y \end{bmatrix}, \quad \boldsymbol{X}_{\mathsf{C}} = \begin{bmatrix} X_{\mathsf{C}} & Y_{\mathsf{C}} & Z_{\mathsf{C}} \end{bmatrix}^{\mathrm{T}}, \tag{10}$$

where $\begin{bmatrix} f_x f_y \end{bmatrix}^{\mathrm{T}}$ is the focal length of the camera and $\begin{bmatrix} c_x c_y \end{bmatrix}^{\mathrm{T}}$ is the principle point. Both of them are the camera intrinsic parameters. For ease of notion, this projection model doesn't consider the distortion error produced by the camera. When extracting keypoints on the image, FollowUpAR first undistort their coordinates so that they can be matched correctly.

The monocular camera provides FollowUpAR an observation model $h(\cdot)$, which reflects the target's 6D pose in the image plane. For the k-th feature point at time i, we have:

$$\begin{aligned} \boldsymbol{x}^i(k) &= h\left(\boldsymbol{X}_{\mathsf{O}}^i(k), \boldsymbol{\theta}^i\right) + \boldsymbol{v}^i(k) \\ &= \boldsymbol{\pi}\left(\boldsymbol{X}_{\mathsf{C}}^i(k)\right) + \boldsymbol{v}^i(k) \\ &= \boldsymbol{\pi}\left(\boldsymbol{R}_{\mathsf{CR}}\left(\boldsymbol{R}_{\mathsf{RO}}^i \boldsymbol{X}_{\mathsf{O}}^i(k) + \boldsymbol{t}_{\mathsf{RO}}^i\right) + \boldsymbol{t}_{\mathsf{CR}}\right) + \boldsymbol{v}^i(k), \end{aligned} \tag{11}$$

where $\boldsymbol{X}_{\mathsf{O}}^i(k)$ is the corresponding 3D point of feature point $\boldsymbol{x}^i(k)$ in object reference, and $\boldsymbol{v}^i(k)$ is the random noise of the feature point.

Tracking Error Distribution: Most existing vision-based systems [30, 31, 52] treat $\boldsymbol{v}^i(k)$ as Gaussian distribution (i.e. normal distribution). The assumption has been proved to be effective in many tracking systems [12, 18]. Similarly, FollowUpAR also performs the least square optimization to minimize the error:

$$\boldsymbol{\theta}^i = \arg\min_{\boldsymbol{\theta}^i} \sum_k \|\boldsymbol{x}^i(k) - h\left(\boldsymbol{X}_{\mathsf{O}}^i(k), \boldsymbol{\theta}^i\right)\|_2^2. \tag{12}$$

3.3 Target Object Separation

As a reminder, it is our superiority that FollowUpAR does not require any prior knowledge of the environment or the target object (e.g., size, shape, color). Previous learning-based solutions require either a pre-trined neural network to recognize the target from the whole image [23, 48], or a 3D rigid model of the target [9, 25], which is labor-intensive and time-consuming. FollowUpAR, in contrast,

leverages the user's tapping on the screen to obtain the rough location of the target object[4]. In the following tracking process, FollowUpAR will rapidly and precisely separate the target object from the background by a filter-like method.

Specifically, once the user clicks on the screen to give an rough target location $\bar{\boldsymbol{x}}$ in the image plane, FollowUpAR selects 8 nearest matched feature correspondences around $\bar{\boldsymbol{x}}$ and estimate an approximate fundamental matrix $\bar{\boldsymbol{F}}$, which represents the inter-frame relative pose of the selected features based on epipolar constraint [18]. Since all selected feature points are concentrated in a small region on the target, we assume that no background features are included. Further, FollowUpAR checks all matched feature points in two frames:

$$\boldsymbol{x}_1^{\mathrm{T}} \bar{\boldsymbol{F}} \boldsymbol{x}_2 < \epsilon, \tag{13}$$

where ϵ is a predefined threshold. Based on Eqn. 13, inliers (i.e. features on the anchor object) are reserved, and outliers (i.e. features on the background) are removed. Finally, based on Eqn. 10, we know that the anchor object must approach the ray starting from the camera optical center and passing through $\bar{\boldsymbol{x}}$ in the image plane. By searching the radar point closest to the ray, FollowUpAR finds out the radar point of the target and filters other measurements.

4 FACTOR-GRAPH-BASED POSE OPTIMIZATION

In this part, we focus on the fusion (a.k.a., joint optimization) of the above two tracking models. We design a factor graph based *Pose Optimization* framework, which takes visual and radar estimations as input and outputs object's pose in real-time. As illustrated in Fig. 6, the whole optimization process consists of two parallel tightly coupled modules: *Tracking Optimization Module* (TOM) and *Modeling Optimization Module* (MOM). Specifically, the TOM ensures the continuous tracking of the object's pose with high precision through a combination of short-term (Inter-frame Tracking, Fig. 6a) and long-term (Local Pose Tracking, Fig. 6b) optimization. The MOM constructs and optimizes the object's 3D point cloud during the tracking process, which in turn enhances the quality of visual feature-point matching and thus improves the tracking performance. A detailed theory of the proposed factor graph is described in Appendix B.

4.1 Tracking Optimization Module

In this section, we introduce the tracking optimization module (TOM), which continuously tracks the 6-DoF pose of the anchor object by both short-term and long-term optimization.

Inter-frame Tracking. As shown in Fig. 6a, once a camera frame and the corresponding radar measurements are available, the inter-frame optimization is performed to give an instant pose tracking result. Similar to modern SLAM systems [31], FollowUpAR first utilizes a constant velocity motion model to predict the current object pose based on the previous one:

$$\begin{aligned} \bar{\boldsymbol{R}}_{\mathsf{R0}}^{i} (\boldsymbol{R}_{\mathsf{R0}}^{i-1})^{\mathrm{T}} &= \boldsymbol{R}_{\mathsf{R0}}^{i-1} (\boldsymbol{R}_{\mathsf{R0}}^{i-2})^{\mathrm{T}}, \\ \bar{\boldsymbol{t}}_{\mathsf{R0}}^{i} - \boldsymbol{t}_{\mathsf{R0}}^{i-1} &= \boldsymbol{t}_{\mathsf{R0}}^{i-1} - \boldsymbol{t}_{\mathsf{R0}}^{i-2}. \end{aligned} \tag{14}$$

[4] A user needs to interact with the screen to determine the initial location of virtual effects in most AR applications.

Then FollowUpAR performs a guided search of the 3D points observed in the last frame. As shown in Fig. 6a, current object pose $\boldsymbol{\theta}^i$ is constrained by two factor nodes. The optimization problem can be formulated as follows:

$$\begin{aligned} \boldsymbol{\chi}^* &= \arg\min_{\boldsymbol{\chi}} \left(\boldsymbol{E}_{radar}^{i} + \boldsymbol{E}_{proj}^{i} \right), \\ \boldsymbol{\chi} &= \{ \boldsymbol{R}_{\mathsf{R0}}^{i}, \boldsymbol{t}_{\mathsf{R0}}^{i} \}, \end{aligned} \tag{15}$$

where the camera projection error term can be given based on Eqn. 27 and Eqn. 28:

$$\begin{aligned} \boldsymbol{E}_{proj}^{i} &= -\log \left(\prod_{k=1}^{K} p(\boldsymbol{x}^i(k) | \boldsymbol{\theta}^i) \right) \\ &\approx \sum_{k=1}^{K} \rho \left(\left\| \boldsymbol{x}^i(k) - \boldsymbol{\pi} \left(\boldsymbol{X}_{\mathsf{C}}^{i}(k) \right) \right\|_{\Omega_C}^2 \right), \\ \boldsymbol{X}_{\mathsf{C}}^{i}(k) &= \boldsymbol{R}_{\mathsf{CR}} \left(\boldsymbol{R}_{\mathsf{R0}}^{i} \boldsymbol{X}_{\mathsf{0}}^{i}(k) + \boldsymbol{t}_{\mathsf{R0}}^{i} \right) + \boldsymbol{t}_{\mathsf{CR}}. \end{aligned} \tag{16}$$

Note that the $\rho(\cdot)$ indicates the Huber loss function [21] adopted in FollowUpAR to increase its resilience to outliers. Similarly, the mmWave radar error term is:

$$\begin{aligned} \boldsymbol{E}_{radar}^{i} &= \rho \left(-\log \left(p(\boldsymbol{U}_{\mathsf{R}}^{i} | \boldsymbol{\theta}^i) \right) \right) \\ &\approx \rho \left(-\log \left(\Psi \left((\boldsymbol{t}_{\mathsf{R0}}^{i} - \boldsymbol{t}_{\mathsf{R0}}^{i-1}) - \boldsymbol{U}_{\mathsf{R}}^{i} \right) \right) \right). \end{aligned} \tag{17}$$

Note that the initial value of the optimization is provided by constant velocity model in Eqn. 14:

$$\bar{\boldsymbol{\chi}}^i = \{ \bar{\boldsymbol{R}}_{\mathsf{R0}}^{i}, \bar{\boldsymbol{t}}_{\mathsf{R0}}^{i} \}. \tag{18}$$

Local Pose Tracking. As shown in Fig. 6b, for every few seconds (in general, a keyframe[5] is selected), the local pose tracking is triggered to correct the accumulated error. As shown in Fig. 6b, Local Pose Tracking takes all frames after the most previous KF as input and jointly optimizes the pose. Denote the set of frames as $\mathcal{F}$, the optimization problem can be formulated as follows:

$$\begin{aligned} \boldsymbol{\chi}^* &= \arg\min_{\boldsymbol{\chi}} \sum_{i \in \mathcal{F}} \left(\boldsymbol{E}_{radar}^{i} + \boldsymbol{E}_{proj}^{i} \right), \\ \boldsymbol{\chi} &= \bigcup_{i \in \mathcal{F}} \{ \boldsymbol{R}_{\mathsf{R0}}^{i}, \boldsymbol{t}_{\mathsf{R0}}^{i} \}. \end{aligned} \tag{19}$$

Global Pose Tracking is a particular case of Local Pose Tracking, where all KFs are included in the set $\mathcal{F}$. It is triggered only if there is a significant error occurs during the whole optimization procedure. The users can also manually trigger it when they feel the tracking error is intolerable.

Generally speaking, the performance of TOM, especially the visual feature-matching-based optimization, highly depends on the quality of the generated point clouds associated with the object. In other words, if the point clouds are not updated and optimized in parallel with TOM, FollowUpAR will suffer from degraded tracking performance. According to our evaluation, leveraging TOM alone for tracking will accumulate more than 15mm translation and 7° rotation bias in merely 3 seconds, which hardly meet the needs of

[5] Keyframes (KFs) represent the frames with high quality of feature points over a period of time. In FollowUpAR, KFs are automatically selected by the system, and the selection strategy is similar to that in ORB-SLAM [30].

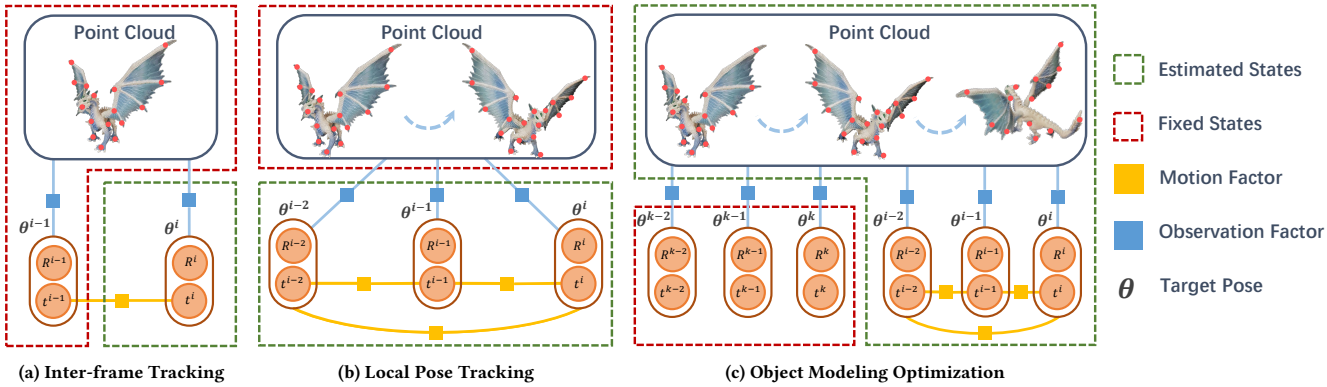


Figure 6: Three Types of Optimization with Factor Graph. (a) and (b) are correspondingly short-term and long-term parts of TOM, while (c) is MOM. The red dotted parts are fixed variables during each optimization process, while the green parts are the estimation goals.

AR applications. Therefore, in what follows, we will present how to optimize the point clouds by MOM.

4.2 Modeling Optimization Module

Modeling Optimization Module (MOM) constructs and optimizes the point clouds attached to the object. As shown in Fig. 6c, the process is triggered when a new KF is selected during system operation. MOM leverages the set of recent frames after the previous KF (denoted as $\mathcal{F}$), and all 3D points observed by those frames as a point set $\mathcal{P}$. Other frames (denoted as $\mathcal{F}'$), which share observations of these points in $\mathcal{P}$, also contribute to the total cost but are fixed during the optimization process. Similar to *Local Pose Tracking*, in MOM, the cost function also includes the radar error term $\boldsymbol{E}_{radar}$ and reprojection error term $\boldsymbol{E}_{proj}$:

$$\begin{aligned} \boldsymbol{\chi}^* &= \arg\min_{\boldsymbol{\chi}} \sum_{i\in\mathcal{F}} \left(\boldsymbol{E}^i_{radar} + \boldsymbol{E}^i_{proj}\right) + \sum_{j\in\mathcal{F}'} \boldsymbol{E}^j_{proj}, \\ \boldsymbol{\chi} &= \bigcup_{i\in\mathcal{F}} \{\boldsymbol{R}^i_{\mathrm{RO}}, \boldsymbol{t}^i_{\mathrm{RO}}\} \bigcup_{k\in\mathcal{P}} \boldsymbol{X}^k_0, \end{aligned} \tag{20}$$

where $\boldsymbol{\chi}$ is the optimization objective of MOM, which consists of the 3D location of each point in $\mathcal{P}$ and the object's pose in each frame in $\mathcal{F}$. In Eqn. 20, minimizing $\boldsymbol{E}_{radar}$ aims to make the optimized target motion fit the mmWave radar measurements, while minimizing $\boldsymbol{E}_{proj}$ makes the association between the optimized object 3D model and captured 2D frames conform to the projection characteristics [31]. The determination of optimization window length N depends on the computation capability of the mobile device.

Moreover, MOM also maintains the model of the target object, creating new 3D points and culling points with large bias. Specifically, as the target object rotates and translates, the camera observes several new visual features on the object from different perspectives. Then, MOM creates new 3D points via the triangulation algorithm [18]. It is worth noting that we assume the anchor object is rigid (i.e. the keypoint pairwise distances remain constant). Once the object's shape has changed, MOM needs to remodel it, and the tracking performance will deteriorate.

5 IMPLEMENTATION AND EVALUATION

5.1 Implementation and Methodology

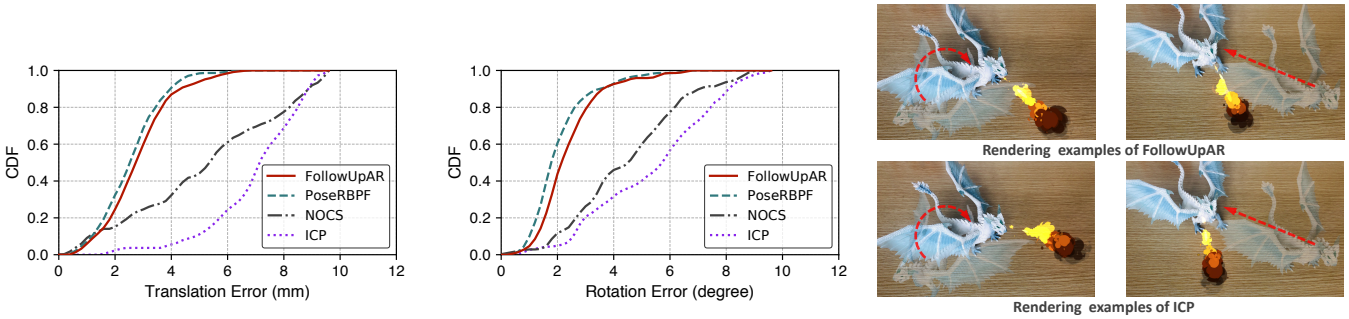
Implementation: We implement FollowUpAR on a commercial smartphone, Google Pixel 4, with a Snapdragon 855 processor and 6GB memory. Nowadays, Google Pixel 4 is equipped with a mmWave radar, however, the underlying signal acquisition API has not been released yet[6], thus we use an additional mmWave radar board, Texas Instruments (TI) IWR1443, to transmit and receive mmWave signals (77 GHz ~ 81 GHz). The TI IWR1443 integrates 3 transmitting antennas (denoted as TX1~TX3) and 4 receiving antennas (denoted as RX1~RX4), which form two linear antennas on the horizontal plane. The raw samples are recorded by a TI DCA1000EVM data acquisition board and further streamed to the smartphone. The camera of the smartphone captures 1080p video at 30fps, and the diagonal field of view (dFoV) is 78°. It is worth mentioning that except the raw data sampling and streaming, entire calculation tasks are processed on the smartphone in real-time, and we aim at providing extra functional modules for smartphone-based AR frameworks.

Experiment setting: Fig. 7 shows the experimental scenarios in a laboratory and a crowded office. The experimental setup is shown in Fig. 7a. As seen, the mmWave radar and camera are fixed on a bracket in the front of the experimental area. We evaluate FollowUpAR on a variety of target objects, including a plush doll, a toy dragon, and some books. All these objects are of different shapes, sizes, and colors. We also deploy four *OptiTrack* [33] cameras with 180fps to cover an area of $3m \times 8m$ for recording ground truth. We conduct extensive experiments for around 15 hours and collecte more than 800,000 video frames and 15GB mmWave raw samples. We additionally use a Kinect-V2 for RGB-D data collection to compare FollowUpAR with related works.

Comparative Methods: We compare our FollowUpAR with three related systems including: 1)**PoseRBPF** [9], the state-of-the-art deep learning based pose tracking system with monocular images as input, however, it can merely track pre-modeled object; 2) **NOCS** [46], a CNN-based system for category-level object pose

[6]Google Pixel 4 aims at enabling intelligent gesture recognition leveraging mmWave radar [1], and Android will gradually open data acquisition APIs.

Figure 7: Experimental setup and scenarios of FollowUpAR. (a) A laboratory scenario. (b)-(e) A crowded office with different illumination intensity (normal and weak), as well as diverse background dynamics (static, slight, and severe), respectively.



(a) Translation Accuracy Comparison **(b) Rotation Accuracy Comparison** **(c) Illustration of Tracking Performance**

Figure 8: Overall performance comparison of FollowUpAR and three related works.(a) and (b) show the translation and rotation accuracy comparison, respectively. (c) visually demonstrates the impact of an average 2-vs.-6° rotation errors (left row) and 3-vs.-7mm translation errors (right row) on virtual effects rendering.

tracking, which takes RGB-D images as input; and 3) **ICP** [57], a classical vision-based tracking algorithm by 3D point-cloud matching, which has been integrated in many SLAM systems [31, 50]. To compare FollowUpAR with PoseRBPF and NOCS, we first model the three target objects and train neural networks in advance. Note that none of these three solutions can run on smartphones in real-time, hence we additionally use a server with multiple CPUs and GPUs to run them offline.

Evaluation Metrics: FollowUpAR reports the real-time 6-DoF pose of the object in each frame. Similar to related works, we calculate the mean rotation error in degrees, and the mean translation error in millimeters.

Robustness Experiments: To demonstrate the ubiquitousness of FollowUpAR, we conduct experiments under diverse conditions, including different environments, illumination intensities, and background motions, as shown in Fig. 7b-e. We also evaluate the impact of different object-phone distances, occlusions, and velocities on system performance to demonstrate the robustness of FollowUpAR.

5.2 Overall Performance

Fig. 8a and Fig. 8b depict the performance of the proposed FollowUpAR as well as three other comparative systems. Since the two learning-based methods, NOCS and PoseRBPF, require complex pre-training in the specific environment, we only compare their performance in the laboratory scenario (Fig. 7a). As shown, FollowUpAR outperforms NOCS and ICP and achieves competitive performance compared with PoseRBPF. The average translation accuracy of FollowUpAR is 3.0*mm*, which outperforms NOCS by 69.9%, and exceeds ICP by more than 74%. As for the rotation tracking performance, the average accuracy of FollowUpAR is 2.6°, which outperforms NOCS and ICP by 47.9%, and 56.3%, respectively. The 95th percentile accuracy outperforms these systems by 47.3%, and 57.6%, correspondingly. As for the performance in different scenarios, the average translation and rotation accuracy of FollowUpAR in the crowded office (Fig. 7c-e) is 3.4*mm* and 2.9°, respectively. Compared with the results in the pure laboratory, the degradation of system performance is within 10%, demonstrating the robustness of FollowUpAR to different environments. Fig. 8c further illustrates a couple of rendering example of FollowUpAR (i.e. 3*mm* translation error and 2° rotation error) and ICP (i.e. 7*mm* tran. error and 6° rota. error). As seen, with increased tracking errors, the orientation and position of the virtual flame are gradually misaligned (bottom row in Fig. 8c), which leaves users the impression of distortion. On the contrary, the tracking accuracy gap between FollowUpAR and PoseRBPF is within 10%, and the rendering performance is almost identical (compared in columns (c) and (d) of Fig. 8). Above theoretical and empirical results demonstrate FollowUpAR achieves remarkable performance gains based

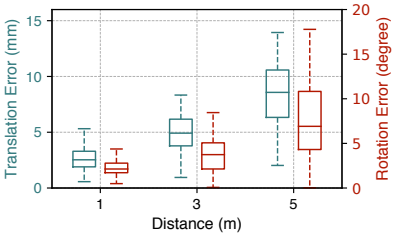


Figure 9: Impact of Distance

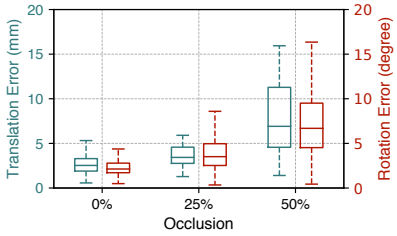


Figure 10: Impact of Occlusion

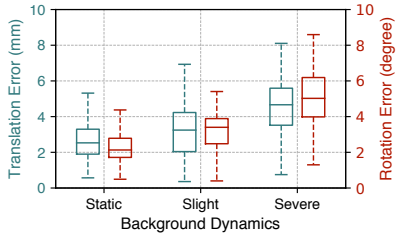


Figure 11: Impact of Background

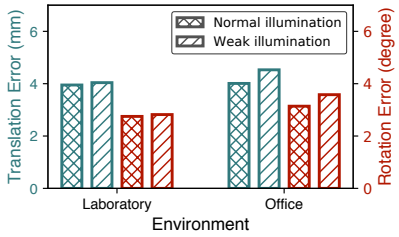


Figure 12: Impact of Env. & Illu.

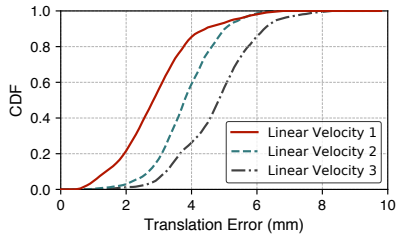


Figure 13: Impact of Linear Velocity

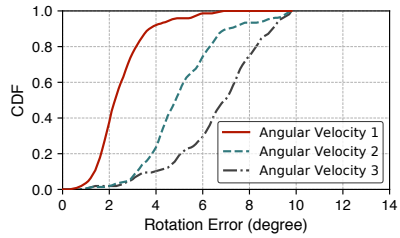


Figure 14: Impact of Angular Velocity

on fusing vision and mmWave sensors. Moreover, as a reminder, FollowUpAR can track an object's pose in real-time and does not require the complex pre- modeling or training procedure, ensuring the ubiquitous of AR-based applications.

5.3 System Robustness Evaluation

5.3.1 Impact of Distance. We examine the impact of the distance between the object and smartphone. The results are shown in Fig. 9. As the distance grows from $1m$ to $5m$, the translation error increases from an average $2.7mm$ to $8.3mm$ while the rotation error increases from 2.4° to 7.5°. In practical usage of mobile AR applications, the distance between the smartphone and the anchor object is generally no more than 3 meters. According to the experimental results, the translation error at this distance is within $3.9mm$ and the rotation error $< 3.1^\circ$, meeting the needs of most AR applications.

5.3.2 Impact of Occlusion. We further verify the robustness of the system when the target objects are partially occluded. As shown in Fig. 10, when 25% of the object is occluded, FollowUpAR is still able to maintain high performance with an average $3.5mm$ translation tracking accuracy and 4.1° rotation accuracy, and the accuracy reduction is within 30%. When the rate rises to 50%, the average translation error and rotation error increase to $8.0mm$ and 7.2°, respectively. Compared with the other model-based and learning-based methods, it is our superiority that FollowUpAR can track the pose of target normally with partial occlusion.

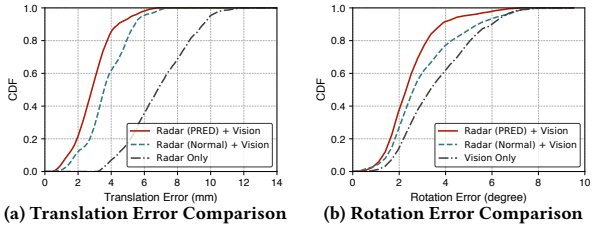
5.3.3 Impact of Background Motion. We conduct experiments with dynamic background motion (as shown in Fig. 7). The evaluation result is presented in Fig. 11. As seen, although the tracking performance decreases with the increasing of background dynamics, FollowUpAR still maintains an average translation tracking accuracy of $4.5mm$ as well as rotation accuracy of 4.3° in most challenging scene. Referring to the rendering examples, we believe the accuracy meets the needs of mobile AR scenarios. Moreover, as mentioned above, a user can manually trigger the *global pose optimization* function to avoid significant errors.

5.3.4 Impact of Environment & Illumination. The performance of FollowUpAR in different environments with different lighting conditions is shown in Fig. 12. Comparing the left and right clusters, the performance of FollowUpAR is almost the same in different environments. As for the system robustness under weak illumination intensity, FollowUpAR maintains the translation tracking bias and rotation bias within 4mm and 3.8°, respectively. Compared with other learning-based solutions, FollowUpAR does not require pre-training in different environments, making FollowUpAR more ubiquitous.

5.3.5 Impact of Velocity. We further evaluate the robustness of FollowUpAR on a different object moving velocities. The results of rotation tracking bias and translation tracking bias are illustrated in Fig. 14 and Fig. 13. We divide the velocities (linear velocity, v and rotation velocity ω) into three categories, corresponding to the demands of different AR applications. In Fig. 14, the angular velocity 1 indicates a slow rotation speed with $\omega < 5^\circ/s$, while the velocity 2 and velocity 3 represent medium and fast speed with $5^\circ/s \leq \omega < 10^\circ/s$ and $10^\circ/s \leq \omega \leq 20^\circ/s$, respectively. As seen, the average rotation tracking error is 2.6°, 5.2°, and 8.3° under slow, medium, and fast speed, respectively. With the rapid rotation, the same feature points on the target object will fail to be observed by consecutive frames, resulting in decreased orientation tracking accuracy. Similary, in Fig. 13, the linear velocity 1, 2, and 3 represent slow, medium, and rapid translation speed with $v < 30mm/s$, $30mm/s \leq v < 60mm/s$, and $60mm/s \leq v < 120mm/s$, respectively. And the corresponding position tracking error is $3.0mm$, $3.9mm$, and $4.9mm$. Although the accuracy of pose tracking will decrease slightly as the moving and rotating speed of the object increases, the tracking performance of FollowUpAR with medium velocity (5.2° orientation with $3.9mm$ translation tracking accuracy) meets the needs of the most smartphone-based AR applications.

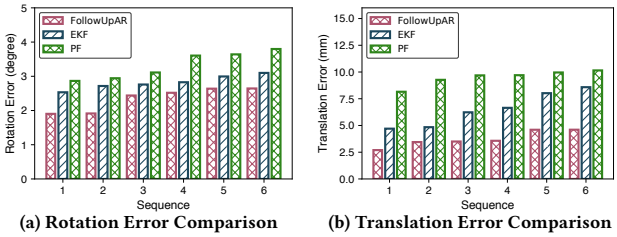
5.4 System Micro-benchmark

We further experimentally analyze some core components of FollowUpAR, and in particular, the performance gains that each of them brings into the overall system.



(a) Translation Error Comparison (b) Rotation Error Comparison

Figure 15: Effectiveness of Sensor Fusion



(a) Rotation Error Comparison (b) Translation Error Comparison

Figure 16: Comparison of Different Fusion Framework

5.4.1 Effectiveness of Multimodule Fusion. In this part, we demonstrate that 1) fusing radar and vision is superior to individual of them, and 2) our proposed *PRED* model performs better in the pose tracking task. As illustrated in Fig. 15a and Fig. 15b, the performance of fusion-based approach far exceeds that of radar-only and vision-only in terms of translation tracking accuracy and rotation accuracy, respectively. Specifically, the average translation error under our proposed *PRED* noise distribution, traditional normal noise distribution, and radar-only-based ranging method is 3.0mm, 3.8mm, and 7.0mm, respectively. As for the rotation tracking performance, the average accuracy of *PRED* outperforms the normal distribution by 18.8%, and the fusion-based tracking exceeds vision-only based solution by 30%. The delightful results lies in our in-depth study of the error generation mechanism of mmWave based ranging, instead of simply treating it as a normal distribution as in previous works.

5.4.2 Anchor Separation. To evaluate the capability of FollowUpAR to separate the anchor object from a complex environment, we conduct several experiments at a different object moving speed and distance. The anchor separation process can be summarized as a binary classification problem, which determines whether a feature point is on the anchor object or not. For each experiment, we evaluate the classification result and calculate the corresponding F1-score as shown in Table 1. The result shows that when the distance is less than 3m, and the anchor object moves with a relatively low speed, the F1-score consistently achieves over 0.9, which is a satisfactory result. With the increase of the distance and the moving speed of the object, the performance of the anchor separation module decreases. This is mainly due to reducing the effective feature points on the object, resulting in an inaccurate pose reconstruction process.

5.4.3 Comparison of Different Fusion Framework. We also compare the pose tracking performance with different multi-modal data fusion strategies. In this part of the experiment, we compare our novel factor graph based fusion framework with another two fusion approaches, particle filter (PF) and extended Kalman filter (EKF),

Table 1: F1-score of Anchor Separation

Distance \ Speed	Velocity 1	Velocity 2	Velocity 3
1 m	0.985	0.964	0.936
3 m	0.957	0.913	0.845
5 m	0.779	0.736	0.627

both widely used in previous works. Fig. 16a and Fig. 16 respectively. As seen, for all experimental traces, FollowUpAR achieves enhanced rotation tracking performance for more than 15% and 30% compared to PF and EKF, respectively, and the rates increase to 40% and 60% when tracking the translation. These outstanding performance, on the one hand, is due to the leverage of the factor graph as a tightly coupled manner to fuse multi-modal data; on the other hand, lies in our theoretical derivation of the error distribution for the two different observations, which have been demonstrated to be more efficient.

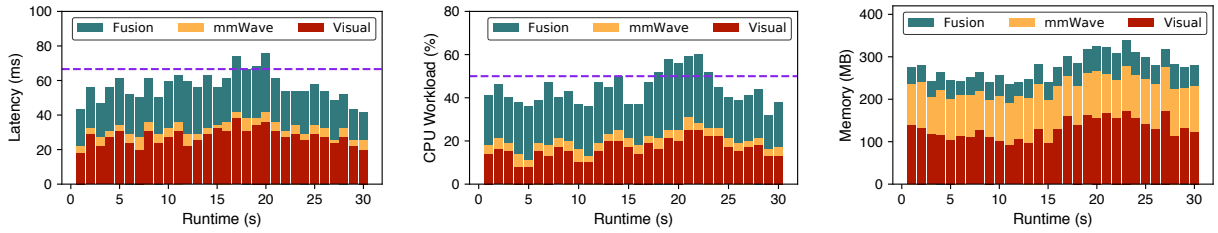
5.4.4 System Efficiency. As a reminder, compared to existing model-based and learning-based methods, it is our superiority that FollowUpAR can run on commercial smartphones in real-time and does not require considerable resource overhead. Fig. 17a illustrates the end-to-end latency (including mmWave-based tracking delay, vision-based tracking delay, and fusion delay) through the whole target pose tracking process. As seen, the average end-to-end latency of FollowUpAR is around 60ms, which indicates FollowUpAR can run on mobile devices at 16fps, meeting the needs of most AR applications. Fig. 17b and Fig. 17c further depict that the CPU usage on the smartphone does not exceed 50% with an average memory usage under 300MB in the process. Compared with other deep learning-based solutions, FollowUpAR leaves the smartphone with sufficient computational resources to further render elegant special effects, as tracking the anchor object's pose is only the fundamental part of the whole AR applications.

6 RELATED WORK

We briefly review the related works in the following.

Vision-based Object 6-DoF Tracking. Most existing 6-DoF tracking systems are based on a given 3D object model. Some previous works try to extract the edges [4, 6] or the color histogram [7] of the target object from the image, and then match them with the template model. Recent data-driven approches [15, 43] instead learn to directly match input images with renderings [26] or silhouette [5, 19] of the object model. PoseRBPF [9] compares the latent code of the input images and that of the model rendering to recover the rotation part of the object pose. Many deep learning based methods [23, 48] utilize CNN-based networks to extract robust features in images for pose estimation. However, the requirements of knowing the object models confine these methods to a specific object instance.

The category-level pose estimation techniques has drawn significant interests in research areas such as autonomous driving [36, 49, 58] due to the availability of large-scale datasets. These works combine the classical keypoint matching ideas and modern deep learning techniques by directly predicting either category-level semantic keypoints [34] or 3D bounding box corners [38, 46]. However, supervised keypoints learning requires a large amount of



(a) System Latency (b) CPU Workload (c) Memory Usage

Figure 17: System Efficiency on the Smartphone

labeled data, and the manually annotated keypoints or the bounding box corners may not be the optimal landmarks to track. It's worth mentioning that the category-level object pose estimation is only applicable to objects of specific shapes and sizes, and the generalization problem still exists. Applying these methods to arbitrary objects may result in severe performance degradation or even unavailability.

Compared with previous works, our FollowUpAR achieves delightful tracking accuracy, without any prior knowledge (e.g., training data and 3D model). Besides, FollowUpAR is also a light-weight mobile framework, which means the entire tracking process can be executed on the smartphone in real-time, shedding lights on future mobile AR/MR applications.

mmWave for Localization and Tracking. Millimeter-wave is highly sensitive and more accurate due to its mm-level wavelength. Previous works adopt mmWave for object tracking [47], human localization [35, 53] and indoor map construction [17, 56]. However, existing mmWave-based works fail to give the target object's rotation and can only achieve centimeter-level location accuracy due to the limited sample rate of the hardware. Though some previous works [28, 47] attempt to filter out the noise and adopt advanced optimization methods to improve the location accuracy, it is still difficult to achieve the millimeter-level result.

To break the accuracy bottleneck, our FollowUpAR explores the actual error distribution of mmWave radar and fuses radar measurement with monocular vision. Therefore, FollowUpAR can not only track the rotation of the object but also achieves millimeter-level location accuracy.

Sensor Fusion Techniques. The Lidar-vision fusion techniques are commonly used in autonomous driving systems for ego-motion estimation and mapping. Most of these techniques [16, 20] are mainly based on lidar's dense point cloud, which provides the depth information for each pixel [55] or feature points [54] in each camera frame. This kind of fusion techniques, which directly associate two types of measurements, are simple and efficient, however, can only be applied for fusing two types of dense measurements (e.g., 3D point cloud with high spatial resolution). Recent DNN-based fusion systems [3, 29] adopt convolutional neural networks (CNN) and recurrent neural networks (RNN) to fuse mmWave radar and IMU signals for end-to-end ego-motion estimation. However, these systems need large amounts of labeled training data and suffer from performance degradation when the environment changes. Inspired by existing visual-inertial odometry (VIO) systems [32, 37], a tightly coupled fusion framework based on the factor graph is adopted in FollowUpAR to jointly optimize the radar tracking model and the camera tracking model. The tracking results given by FollowUpAR represents the most likely 6D pose of the target object, which has a clear interpretation in probability. The most important thing is, compared with DNN-based methods, our fusion framework is faster and more efficient, which guarantees its real-time performance on resource-limited mobile devices.

7 CONCLUSIONS

In this paper, we propose the design and implementation of FollowUpAR, the first framework that fuses sparse measurement and vision to track 6-DoF pose of object. The innovation of FollowUpAR lies in two aspects: 1) we derive a theoretical model of radar error distribution based on in-depth understanding of its hardware-level working principles; and 2) we design a novel factor graph competent in fusing heterogeneous data. Extensive evaluation results demonstrate its superior performance over previous multi-agent solutions. We integrate FollowUpAR in ARCore, and believe FollowUpAR takes a promising step towards enabling smartphones to render follow-up effects in mobile AR applications.

8 ACKNOWLEDGMENTS

We sincerely thank our shepherd and the anonymous reviewers for their valuable feedback in improving this work. This work is supported in part by the National Key Research Plan under grant No. 2018AAA0101200, the NSFC under grant 61832010.

REFERENCES

[1] Google Pixel 4. 2020. https://en.wikipedia.org/wiki/Pixel_4
[2] Lie algebra. 2006. https://en.wikipedia.org/wiki/Lie_algebra
[3] Y. Almalioglu, M. Turan, C. X. Lu, N. Trigoni, and A. Markham. 2020. Milli-RIO: Ego-Motion Estimation with Low-Cost Millimetre-Wave Radar. *IEEE Sensors Journal* (2020).
[4] P. Azad, D. Münch, T. Asfour, and R. Dillmann. 2011. 6-DoF model-based tracking of arbitrarily shaped 3D objects. In *2011 IEEE International Conference on Robotics and Automation*.
[5] G. Billings and M. Johnson-Roberson. 2019. SilhoNet: An RGB Method for 6D Object Pose Estimation. *IEEE Robotics and Automation Letters* (2019).
[6] C. Choi and H. I. Christensen. 2010. Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *2010 IEEE International Conference on Robotics and Automation*.
[7] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. 2011. The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research* (2011).
[8] Google AR Core. 2021. https://developers.google.com/ar
[9] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. 2019. Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking. *arXiv preprint arXiv:1905.09304* (2019).
[10] L. Ding, M. Ali, S. Patole, and A. Dabak. 2016. Vibration parameter estimation using FMCW radar. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
[11] Sedat Dogru and Lino Marques. 2020. Pursuing drones with drones using millimeter wave radar. *IEEE Robotics and Automation Letters* (2020).

[12] Erqun Dong, Jingao Xu, Chenshu Wu, Yunhao Liu, and Zheng Yang. 2019. PairNavi: Peer-to-Peer Indoor Navigation with Mobile Visual SLAM. In *Proceedings of the IEEE INFOCOM*.
[13] Liang Dong, Jingao Xu, Guoxuan Chi, Danyang Li, Xinglin Zhang, Jianbo Li, Qiang Ma, and Zheng Yang. 2020. Enabling Surveillance Cameras to Navigate. In *Proceedings of the IEEE ICCCN*.
[14] Paul Furgale, Joern Rehder, and Roland Siegwart. 2013. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*.
[15] M. Garon and J. Lalonde. 2017. Deep 6-DOF Tracking. *IEEE Transactions on Visualization and Computer Graphics* (2017).
[16] J. Graeter, A. Wilczynski, and M. Lauer. 2018. LIMO: Lidar-Monocular Visual Odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
[17] F. Guidi, A. Guerra, and D. Dardari. 2016. Personal Mobile Radars with Millimeter-Wave Massive Arrays for Indoor Mapping. *IEEE Transactions on Mobile Computing* (2016).
[18] Richard Hartley and Andrew Zisserman. 2003. *Multiple view geometry in computer vision*. Cambridge university press.
[19] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. 2019. Segmentation-Driven 6D Object Pose Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[20] S. S. Huang, Z. Y. Ma, T. J. Mu, H. Fu, and S. M. Hu. 2020. Lidar-Monocular Visual Odometry using Point and Line Features. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*.
[21] Peter J Huber. 2004. *Robust statistics*. John Wiley & Sons.
[22] Chengkun Jiang, Junchen Guo, Yuan He, Meng Jin, Shuai Li, and Yunhao Liu. 2020. mmVib: micrometer-level vibration measurement with mmwave radar. In *Proceedings of the ACM Mobicom*.
[23] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. 2017. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *2017 IEEE International Conference on Computer Vision (ICCV)*.
[24] Apple AR Kit. 2021. https://developer.apple.com/augmented-reality/
[25] Alexander Krull, Frank Michel, Eric Brachmann, Stefan Gumhold, Stephan Ihrke, and Carsten Rother. 2014. 6-DOF Model Based Tracking via Object Coordinate Regression. In *Computer Vision – ACCV 2014*. Springer International Publishing.
[26] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. 2018. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.
[27] Jaime Lien, Nicholas Gillian, M Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)* (2016).
[28] Chris Xiaoxuan Lu, Stefano Rosa, Peijun Zhao, Bing Wang, Changhao Chen, John A Stankovic, Niki Trigoni, and Andrew Markham. 2020. See through smoke: robust indoor mapping with low-cost mmWave radar.. In *MobiSys*.
[29] Chris Xiaoxuan Lu, Muhamad Risqi U Saputra, Peijun Zhao, Yasin Almalioglu, Pedro PB de Gusmao, Changhao Chen, Ke Sun, Niki Trigoni, and Andrew Markham. 2020. milliEgo: single-chip mmWave radar aided egomotion estimation via deep sensor fusion. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*.
[30] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* 31, 5 (2015), 1147–1163.
[31] Raul Mur-Artal and Juan D Tardós. 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics* 33, 5 (2017), 1255–1262.
[32] R. Mur-Artal and J. D. Tardós. 2017. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robotics and Automation Letters* (2017).
[33] Opti-Track. 2016. https://optitrack.com/
[34] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 2017. 6-DoF object pose from semantic keypoints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*.
[35] Ioannis Pefkianakis and Kyu-Han Kim. 2018. Accurate 3D Localization for 60 GHz Networks. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*.
[36] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. 2018. Frustum PointNets for 3D Object Detection From RGB-D Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[37] T. Qin, P. Li, and S. Shen. 2018. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics* (2018).
[38] M. Rad and V. Lepetit. 2017. BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth. In *2017 IEEE International Conference on Computer Vision (ICCV)*.
[39] Anshul Rai, Krishna Kant Chintalapudi, Venkata N. Padmanabhan, and Rijurekha Sen. 2012. Zee: Zero-effort Crowdsourcing for Indoor Localization. In *Proceedings of the ACM MobiCom*.
[40] Virtual Reality and Augmented Reality Device Sales to Hit 99 Million Devices in 2021. 2017. http://www.capacitymedia.com/Article/3755961/VR-and-AR-device-shipments-to-hit-99m-by-2021
[41] The reality of VR/AR growth. 2017. https://techcrunch.com/2017/01/11/the-reality-of-vrar-growth/
[42] D. Scaramuzza and F. Fraundorfer. 2011. Visual Odometry [Tutorial]. *IEEE Robotics Automation Magazine* (2011).
[43] D. J. Tan, F. Tombari, S. Ilic, and N. Navab. 2015. A Versatile Learning-Based 3D Temporal Tracker: Scalable, Robust, Online. In *2015 IEEE International Conference on Computer Vision (ICCV)*.
[44] Dorin Ungureanu, Federica Bogo, Silvano Galliani, Pooja Sama, Xin Duan, Casey Meekhof, Jan Stühmer, Thomas J. Cashman, Bugra Tekin, Johannes L. Schönberger, Pawel Olszta, and Marc Pollefeys. 2020. HoloLens 2 Research Mode as a Tool for Computer Vision Research. arXiv:2008.11239 [cs.CV]
[45] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu. 2020. 6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*.
[46] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. 2019. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[47] C. Wu, F. Zhang, B. Wang, and K. J. Ray Liu. 2020. mmTrack: Passive Multi-Person Localization Using Commodity Millimeter Wave Radio. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*.
[48] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. 2018. *PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes*.
[49] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. 2018. PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[50] Jingao Xu, Hao Cao, Danyang Li, Kehong Huang, Chen Qian, Longfei Shangguan, and Zheng Yang. 2020. Edge Assisted Mobile Semantic Visual SLAM. In *Proceedings of the IEEE INFOCOM*.
[51] Jingao Xu, Hengjie Chen, Kun Qian, Erqun Dong, Min Sun, Chenshu Wu, Li Zhang, and Zheng Yang. 2019. iVR: Integrated Vision and Radio Localization with Zero Human Effort. In *PACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*.
[52] Jingao Xu, Erqun Dong, Qiang Ma, Chenshu Wu, and Zheng Yang. 2021. Smartphone-Based Indoor Visual Navigation with Leader-Follower Mode. *ACM Transactions on Sensor Networks (TOSN)* (2021).
[53] Zheng Yang, Zimu Zhou, and Yunhao Liu. 2013. From RSSI to CSI: Indoor localization via channel response. *ACM Computing Surveys (CSUR)* (2013).
[54] Ji Zhang, Michael Kaess, and Sanjiv Singh. 2017. A Real-Time Method for Depth Enhanced Visual Odometry. (2017).
[55] J. Zhang and S. Singh. 2015. Visual-lidar odometry and mapping: low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*.
[56] A. Zhou, S. Yang, Y. Yang, Y. Fan, and H. Ma. 2019. Autonomous Environment Mapping Using Commodity Millimeter-wave Network Device. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*.
[57] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *CoRR* (2018).
[58] Yin Zhou and Oncel Tuzel. 2018. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

A PHYSICAL-LEVEL RADAR ERROR DISTRIBUTION

According to the sources of physical-level noise mentioned in Section 3.1, the *Physical-level Radar Error Distribution (PRED)* can be derived as follows.

We first analyze the radar distance error caused by the limited radar sampling rate Δf. The extracted discrete frequency follows a uniform distribution from $f_{IF}^0 - \frac{\Delta f}{2}$ to $f_{IF}^0 + \frac{\Delta f}{2}$. Based on the radar ranging principle in Eqn. 3, the distance error also follows a uniform distribution:

$$n_D^s \sim \mathcal{U}(-\frac{c\Delta f}{4K}, \frac{c\Delta f}{4K}). \tag{21}$$

Secondly, the thermal noise could also cause a distance error, which can be formulated as follows:

$$n_D^t \sim \mathcal{N}(0, \frac{c\sigma_t}{2K}), \tag{22}$$

where the σ_t depends on the frequency stability of VCO, which can be easily get from the hardware manual. We should also consider the angle error caused by the AWGN channel:

$$\boldsymbol{n}_a \sim \mathcal{N}(0, \frac{\lambda\sigma_a}{2\pi d}), \tag{23}$$

where the σ_a is determined by the number of multipath in the working environment. To sum up, we give the following equations:

$$\begin{aligned} D &= D_0 + n_D = D_0 + n_D^s + n_D^t, \\ \boldsymbol{a} &= \boldsymbol{a_0} + \boldsymbol{n}_a, \end{aligned} \tag{24}$$

where n_D follows a superposition of a normal distribution and a uniform distribution, and the $\boldsymbol{n}_a$ follows a zero-mean normal distribution. Based on the radar working principles, the analysis of radar measurement error is shown as follows:

$$\begin{aligned} D\boldsymbol{a} &= D_0\boldsymbol{a}_0 + D_0\boldsymbol{n}_a + n_D\boldsymbol{a}_0 + n_D\boldsymbol{n}_a \\ &\approx D_0\boldsymbol{a}_0 + (D_0\boldsymbol{n}_a + n_D\boldsymbol{a}_0) \\ &= D_0\boldsymbol{a}_0 + \boldsymbol{n}_P. \end{aligned} \tag{25}$$

Denote $\boldsymbol{n}_P \sim \mathcal{D}(0, a, \sigma)$ as the *PRED*, which can be treated as a superposition of a normal distribution $\mathcal{N}(0, \sigma)$ and a uniform distribution $\mathcal{U}(-a, a)$. Therefore, the probabilistic density function $f(\cdot)$ of radar error distribution can be obtained based on the convolution formula as follows:

$$\begin{aligned} f(e) &= \int_{-a}^{a} \frac{1}{2a} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(e-x)^2}{2\sigma^2}\right) dx \\ &= \frac{1}{2a}\left(\Phi\left(\frac{e+a}{\sigma}\right) - \Phi\left(\frac{e-a}{\sigma}\right)\right), \end{aligned} \tag{26}$$

where $\Phi(\cdot)$ represents the cumulative distribution function (CDF) of the standard normal distribution.

B FACTOR GRAPH IN OUR SYSTEM

Generally speaking, a typical factor graph like Fig. 6 consists of two types of nodes. The variable nodes indicate the values to be optimized (e.g. $\boldsymbol{\theta}^i$), while the factor nodes represent the probability relationship between two variable nodes (e.g. $p(\boldsymbol{x}^i(k)|\boldsymbol{\theta}^i)$). To be more specific, the factor nodes in our FollowUpAR can be further divided into two types: the factor nodes based on the radar tracking model denoted as $p(\boldsymbol{U}_{\mathsf{R}}^i|\boldsymbol{\theta}^i)$, and the factor nodes based on the camera tracking model $p(\boldsymbol{x}^i(k)|\boldsymbol{\theta}^i)$.

In order to estimate the values of a certain set of variable nodes $\boldsymbol{\chi} = \{\boldsymbol{\theta}^i | i \in \mathcal{P}\}$, FollowUpAR optimizes all the factor nodes connected with them based on maximum a posteriori (MAP) estimation:

$$\begin{aligned} \boldsymbol{\chi}^* &= \arg\max_{\boldsymbol{\chi}} \prod_{i\in\mathcal{P}} p(\boldsymbol{\theta}^i|\boldsymbol{U}_{\mathsf{R}}^i) p(\boldsymbol{\theta}^i|\boldsymbol{x}^i) \\ &= \arg\max_{\boldsymbol{\chi}} \prod_{i\in\mathcal{P}} (p(\boldsymbol{\theta}^i))^2 p(\boldsymbol{U}_{\mathsf{R}}^i|\boldsymbol{\theta}^i) \prod_{k=1}^{K} p(\boldsymbol{x}^i(k)|\boldsymbol{\theta}^i), \end{aligned} \tag{27}$$

which follows the Bayes theorem, and supposing all measurements are independent. The $p(\boldsymbol{U}_{\mathsf{R}}^i|\boldsymbol{\theta}^i)$ and $\prod_{k=1}^{K} p(\boldsymbol{x}^i(k)|\boldsymbol{\theta}^i)$ are the likelihood of the radar and camera measurements respectively, and $p(\boldsymbol{\theta}^i)$ is a prior probability over $\boldsymbol{\theta}^i$. In most cases no prior knowledge of $\boldsymbol{\theta}^i$ is available, so $p(\boldsymbol{\theta}^i)$ can be treated as a uniform distribution (or a constant), and thus, the MAP estimation reduces to maximum likelihood (ML) estimation.

Considering that we have analyzed the random noise distribution of $\boldsymbol{v}$ in Eqn. 11 following a zero-mean normal distribution, and the probabilistic density function of $\boldsymbol{w}$ derived from Eqn. 9 as $\Psi(\cdot)$. Therefore, the measurement likelihood can be presented as follows:

$$\begin{aligned} p(\boldsymbol{U}_{\mathsf{R}}^i|\boldsymbol{\theta}^i) &\propto \Psi\left((\boldsymbol{t}_{\mathsf{R0}}^i - \boldsymbol{t}_{\mathsf{R0}}^{i-1}) - \boldsymbol{U}_{\mathsf{R}}^i\right), \\ p(\boldsymbol{x}^i|\boldsymbol{\theta}^i) &\propto \exp\left(-\frac{1}{2}\left\|p(\boldsymbol{x}^i - \boldsymbol{\pi}\left(\boldsymbol{X}_{\mathsf{C}}^i(k)\right)\right\|_{\Omega_C}^2\right), \end{aligned} \tag{28}$$

where we use the notation $\|\boldsymbol{e}\|_{\Omega}^2 = \boldsymbol{e}^{\mathrm{T}}\Omega\boldsymbol{e}$. The Ω_C is the information matrix (i.e. inverse of the covariance matrix) of the camera measurements.