When IoT met Augmented Reality: Visualizing the Source of the Wireless Signal in AR View

Yongtae Park*
ytpark@korea.ac.kr
Korea University

Sangki Yun*
sangki.yun@hpe.com
Hewlett Packard Labs

Kyu-Han Kim
kyu-han.kim@hpe.com
Hewlett Packard Labs

ABSTRACT

This paper presents VisIoT, a system that tracks the location of a wireless transmitter in IoT devices and displays it in the screen of an AR device such as smart glasses and tablet. The proposed system benefits existing IoT systems by enabling intuitive interaction between a user and IoT devices and further enhancing visualization of the data collected from IoT sensors. VisIoT achieves them through a combination of wireless sensing and camera motion tracking. By using the azimuth and elevation angles between the wireless transmitter and the camera-equipped mobile device, VisIoT can instantly identify the location of the IoT device from the camera image. This paper introduces novel azimuth and elevation estimation algorithms that leverage the phase difference of the signals from two antennas together with the tracked camera rotation. We prototype VisIoT using a tablet PC and a USRP software radio, and develop a software that tracks and visualizes the location of ZigBee nodes in real time. The evaluation results show that VisIoT can accurately track the nodes with the median position error of 6%.

KEYWORDS

Internet of Things; Augmented Reality; Wireless Device Tracking

ACM Reference Format:
Yongtae Park, Sangki Yun, and Kyu-Han Kim. 2019. When IoT met Augmented Reality: Visualizing the Source of the Wireless Signal in AR View. In *The 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '19), June 17–21, 2019, Seoul, Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3307334.3326079

1 INTRODUCTION

Motivation: Internet of Things (IoT) has emerged as a promising infrastructure framework, where a large number of inter-connected devices collect data and provide intelligent services such as automated control, process optimization, and anomaly detection. Due to a wide variety of use cases and huge potential, it is expected that IoT market size will grow up to $16 trillion by 2025 [1]. One of the useful ways of taking advantage of IoT system is real time visualization combined with Augmented Reality (AR), where the AR device recognizes the IoT devices in the screen and overlays the received data at the position of them. It not only provides straightforward data interpretation, but also allows intuitive interaction between human and IoT system. Below, we introduce three interesting applications.

Figure 1: IoT visualization in smart farming.

(1) *Industrial IoT:* Consider a smart factory supported by a number of IoT devices, where workers are using AR devices such as smart glasses and tablet. The collected data from IoT sensors are overlapped at the location of the sensors in the screen of the AR device. It can identify the exact location that needs more attention such as an out of stock inventory and a machine with high temperature. Also, by clicking the IoT device in the screen, one can easily change the configuration such as monitoring interval and data delivery frequency.
(2) *IoT in agriculture:* Smart farming is an IoT based farm management supported by inter-connected wireless sensors, controllers, and actuators. With IoT visualization, farmers with smart glasses and tablet can instantly identify the spot that needs more fertilizer or pest control while working in the field. They can also easily change the pressure level of the IoT-enabled water pump by clicking it shown in the AR device screen. Figure 1 shows the conceptual illustration of the IoT visualization in smart farming scenario.
(3) *Home IoT:* Suppose you want to remotely turn off an IoT-enabled light. Among many of them, you intend to turn off the light in the kitchen. By showing it to the smartphone screen and clicking it, you can easily control the light. It can be also useful to visualize data from home IoT devices without screen such as smart power plug and smoke sensor.

Challenge: Such concepts of IoT with AR has been introduced in prior work [2–7], where it is emphasized that it will become one of the key applications of AR. However, there is no solution to enable it yet. Two key requirements are: 1) determining the position

*Both authors contributed equally to this research.

MobiSys '19, June 17–21, 2019, Seoul, Korea

ACM ISBN 978-1-4503-6661-8/19/06…$15.00
https://doi.org/10.1145/3307334.3326079

of the IoT devices in the AR video, and 2) matching it with the wireless data frames that they are sending. With recent advances in computer vision techniques, it might not be difficult to recognize the known IoT devices in the video. However, it does not satisfy our requirements because it cannot match the recognized device with the data it sends. Also, it is difficult to know the appearance of all different kinds of IoT devices in advance.

Another possible approach is wireless signal based device tracking, which locates the source of the wireless signal and matches it with the device shown in the camera image. Although there has been significant improvement in wireless localization techniques, directly applying existing wireless localization techniques does not accomplish our goal. Recent tracking techniques achieve centimeter-level accuracy in 3D location tracking (*e.g.*, CAT [8] and μLocate [9]). However, to enable visualization, a very accurate camera orientation tracking is required in addition to the location tracking. Also, existing techniques track the wireless device based on the known location of the anchor nodes. Since there are lots of IoT devices deployed, it is not always possible to know the exact 3D location of them in advance. Some of the IoT nodes such as asset tracking beacons [10] will move over time and their locations are inherently unknown.

Our approach: In this paper, we propose *Visual-IoT* (VisIoT) that visualizes the source of the wireless signal on the screen of the AR device in real time. With a smartphone, tablet, or smart glasses, a user moves around and takes the video of surroundings. If the wireless signal is received and it is determined that the sender is within the camera angle, we estimate the position of the wireless sender in the camera image, and overlay the received data on top of it. Our novel approach here is, rather than trying to find the position of the AR device in world coordinates and matching it with the known location of the IoT device, we focus on directly finding the position of the IoT device in the 2D camera coordinates of the AR device. We show that if the azimuth and the elevation angles between the sender and the camera device are given, we can find the position of the sender in the camera image.

Based on it, we develop an idea to estimate the azimuth and elevation using the phase difference of the received signals from the two antennas, combined with the motion of the AR device tracked by Inertial Measurement Unit (IMU) sensors. We also provide solutions for filtering out wireless nodes in Non-Line-of-Sight (NLOS) and out of the viewing angle of the camera, and maintaining the device tracking while the user is moving in the area. In our approach, we refrain from applying computer vision based object recognition techniques and focus on developing solutions purely relying on wireless signal and IMU sensor data. We believe combining our approach with computer vision technique can further refine the tracking accuracy and make the system more reliable, but it is beyond the scope of this paper.

We develop a prototype of VisIoT on Microsoft Surface 2017 tablet PC connected to USRP B210 software radio. Using MicaZ ZigBee motes as IoT devices, we visualize the location of them and evaluate the tracking accuracy. Since most IoT systems dominantly use low-power communication techniques, we mainly focused on ZigBee in our prototype development. However, the proposed solution is directly applicable to any wireless communication technique that can estimate AoA, such as WiFi. With extensive experiments, we show VisIoT achieves high accuracy. It estimates the azimuth and elevation angles between AR device and IoT device with median errors of 3.2° and 4.6°, respectively. Based on it, it visualizes the position of the IoT devices with the median position error of 134 pixels from 1,920 × 1,080 pixels video. VisIoT accurately tracks the devices when they are covered by plastic, paper, and foliage. The computational cost of our tracking algorithm is very low. In our testbed, tracking all of the devices within the camera angle of view takes less than 1*ms*. Also, the device are reliably tracked while the data frames are delivered with 100*ms* interval.

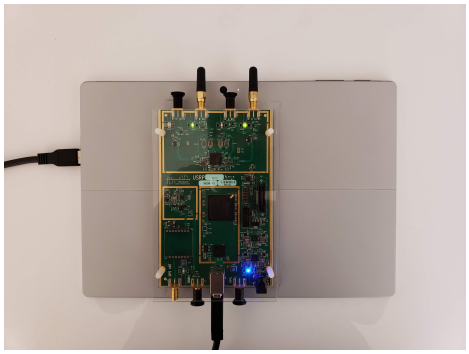


Figure 2: Prototype device for VisIoT.

Summary of contribution: To the best of our knowledge, this is the first approach to find the location of the wireless transmitter in the video without relying on computer vision techniques. The formulation in Section 2.1 clarifies how to locate the wireless sender in the camera coordinates using azimuth and elevation. In Section 2.2, we provide practical and accurate azimuth and elevation estimation schemes based on the received wireless signal and the tracked camera rotation. Section 2.3 introduces how to find the real position of the wireless sender given the ambiguity of the azimuth and elevation estimation. Section 2.5 describes NLOS filtering, and in section 2.5, we introduce how to maintain the tracked position given the movement of the user.

2 SYSTEM DESIGN

We develop a holistic system, VisIoT, which tracks the position of the wireless sender and visualizes the received data at the location of the sender in the AR device screen. It consists of three sensing modules: Radio Frequency (RF) module, IMU sensor, and camera. The RF module decodes the ZigBee frames and calculates the AoA of the signal in real time. Also, it sends ZigBee data frames to IoT devices. IMU sensors (*i.e.*, gyroscope and accelerator) are used to detect the motion transformation of the camera. Figure 2 shows the prototype device of VisIoT. It uses Microsoft Surface 2017 tablet PC as a main AR device, which is equipped with built-in IMU sensors, rear camera, Intel i7 CPU, and 8 GB memory. For RF module, we use USRP B210 software defined radio connected to the tablet through USB 3.0 interface. We attached two antennas to the RF front-end with separation of 5.6 *cm*. The antennas are tightly attached to the tablet so that the camera is exactly at the middle of the two antennas, and the two antennas are linearly aligned to the X-axis of the camera coordinates.

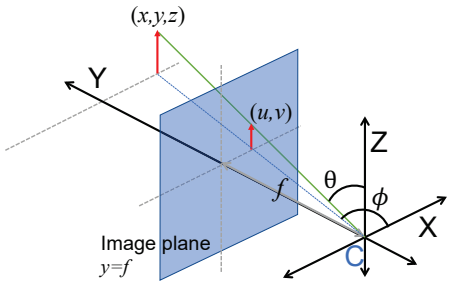


Figure 3: Pinhole camera geometry.

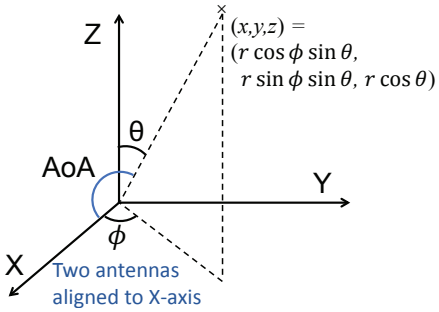


Figure 4: Spherical coordinates system.

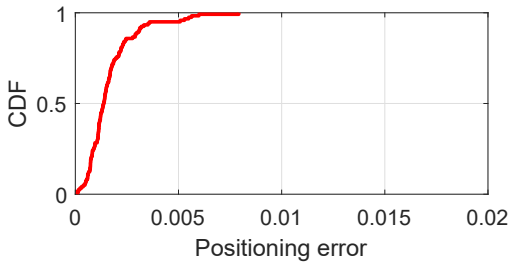


Figure 5: Distribution of the tracking error when we locate the position of the object in the camera image using azimuth and elevation.

2.1 Object Position in Camera Image

To find the location of the IoT device on the AR device screen, we need to understand the relation between the real location of the object in 3D world and its position in 2D camera image. The camera geometry shown in Figure 3 illustrates their relation, where C is the camera center and (x, y, z) is the physical location of the object in the 3D Cartesian coordinates. The pixel location of the object in the 2D camera image is denoted as (u, v). The relation between (x, y, z) and (u, v) is

$$(u, v) = (f\frac{x}{y}, f\frac{z}{y}), \tag{1}$$

where f is the focal length of the camera divided by the pixel size of the image sensor, available from the specification of the camera [11].

In addition to Cartesian coordinates, another way of representing the position in 3D space is Spherical coordinates, which is shown in Figure 4. The position of a point is specified by 1) the radial distance r from the origin to the point, 2) the azimuth angle ϕ in the xy-plane from the x-axis, and 3) the polar angle θ, also called elevation, from the z-axis. θ and ϕ are calculated by $\arccos(\frac{z}{r})$ and $\arctan(\frac{y}{x})$, respectively. The Spherical coordinates (r, θ, ϕ) are converted to the Cartesian coordinates (x, y, z) by:

$$(x, y, z) = (r\cos\phi\sin\theta, r\sin\phi\sin\theta, r\cos\theta). \tag{2}$$

By combining Equations 1 and 2, we can represent the position of the object in the camera image as

$$(u, v) = (f\frac{\cos\phi}{\sin\phi}, f\frac{\cos\theta}{\sin\phi\sin\theta}). \tag{3}$$

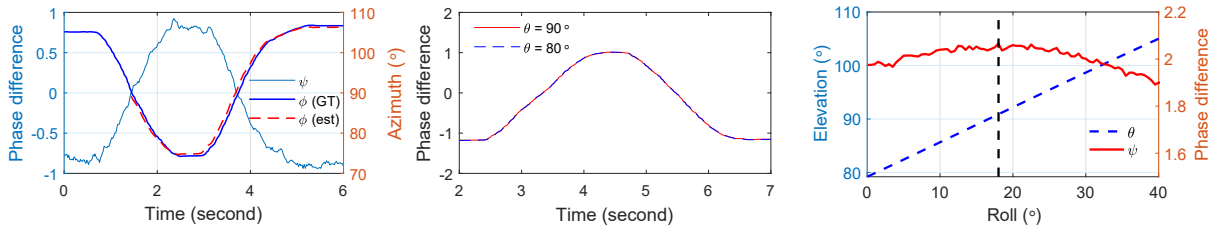
The relation between the real position of the object in Spherical coordinates and its pixel location in the 2D image shown in the above equation is promising for wireless source visualization, because it eliminates the need to estimate the distance r. While the distance estimation relying on RSSI measurement is known to be unreliable, angle estimation assisted by the antenna array is relatively more accurate [12, 13] [1].

The camera geometry model used in Equation 1 and 3 assumes pinhole camera, while most widely used digital cameras are either Charge-coupled device (CCD) or Complementary Metal-Oxide Semiconductor (CMOS) cameras. The picture from those cameras might include distortion [15], where the position of the object might be different from that calculated by Equation 3. We performed an experiment to quantify how accurately we can locate the object in the camera image only relying on the azimuth and the elevation angles. With the rear-camera of MS surface Pro 2017, we take pictures of small objects while varying the position of them. The resolution of the image is $3,264 \times 1,836$ pixels. With the known position of the objects, we calculate the ground-truth azimuth and elevation angles between the camera and the objects. Then, we estimate the position of the object in the image using Equation 3. The focal length and the pixel size of the camera sensor are 3.57 mm and 0.14 μm, respectively, where $f = 2,550$. However, we found it includes a slight approximation error, and adjusted f to 2,598 after empirical fitting. We confirmed the value is consistent with different Surface 2017 tablets with the same hardware. Figure 5 shows the CDF of the position error with 120 samples. The error is calculated as the pixel distance between ground-truth and estimated position normalized by the number of pixels in diagonal line. The result shows the error does not exceed 0.008. The median error is 0.001, which is 5 pixels. Overall, the tracking error is hard to be perceived by user. We conducted the same experiment with a Samsung Galaxy S9 plus mobile phone, and observed similar performance. The median and maximum errors with the Galaxy S9 are 0.001 and 0.007, respectively.

2.2 Estimating Azimuth and Elevation

The result in the previous section confirms that only relying on the azimuth and the elevation angles between the wireless transmitter and the camera device, we can find the exact location of the signal source in the camera image. Then the question might be how to

[1] Equation 3 is first introduced in [14], in a slightly different form (*i.e.*, Cylindrical coordinates). The main application of it is stretching the panoramic view image. To the best of our knowledge, VisIoT is the first approach to use it to find the location of the wireless sender in the image.



(a) Estimating ϕ using Z-axis rotation.

(b) Change of ψ during Z-axis rotation, when $\theta = 90°$ and $80°$.

(c) Change of ψ and θ during Y-axis rotation.

Figure 6: Illustration of the ψ, ϕ, and θ change depending on the rotation of the device .

find these angles. The answer would be observing the phase difference of the received signals from multiple antennas [16]. Let λ be the wavelength of the wireless signal. The phase of the received signal rotates from $-\pi$ to π whenever the signal travels a distance of λ. Let D be the distance between two antennas, and ψ be the phase difference of the signals from the two antennas. The relation between ψ and the azimuth and elevation angles is:

$$\psi = \frac{2D\pi}{\lambda} \cos\phi \sin\theta. \tag{4}$$

How to separate the azimuth and elevation from the given ψ estimated from the received wireless signal is the main challenge of our system design.

Regarding the azimuth and elevation estimation, there have been many related studies over the last few decades [17–22]. However, it is difficult to directly apply the existing techniques to AR devices, because they require specialized antenna arrays such as uniform circular array [17, 20], L-shaped array [18, 19], and parallel linear array [21, 22] with 8 or more antennas. This section introduces our novel solution to estimate azimuth and elevation only using the 2 antennas of the mobile device combined with the camera motion tracked by gyroscope.

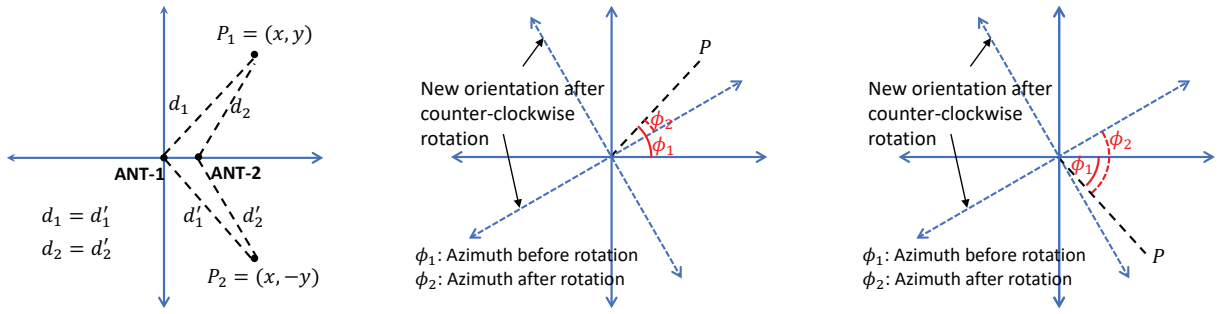
During the wireless source visualization phase, the AR user rotates the device. The amount of the rotation is calculated by integrating the angular velocity measured by gyroscope. Meanwhile, the RF receiver also collects the phase differences over time from the sequence of the received wireless frames. Given them, one possible approach of azimuth and elevation estimation would be emulating circular antenna array using the rotation of the device (*i.e.*, Synthetic Aperture Radar). However, such an approach is very sensitive to AoA and gyroscope measurement noise [23]. Instead, we propose an alternative solution to estimate the azimuth and elevation. The main idea is exploiting the phase difference only for finding reference points, and more relying on the gyroscope based rotation estimation. This is based on our novel observation on the turning points of the phase depending on the rotation of the device. The proposed angle estimation algorithm causes low computing overhead and can easily run in real time in mobile AR devices.

Azimuth estimation: Equation 4 implies that when the elevation is fixed, the phase difference ψ monotonically decreases as the azimuth ϕ increases from 0 to π[2]. Also, ψ becomes zero when $\phi = \frac{\pi}{2}$. Based on it, we design an azimuth estimation algorithm as follows. We first consider the case when the wireless sender is at the left side of the user (*i.e.*, $\frac{\pi}{2} < \phi < \pi$). It can be easily identified by finding the sender with ψ smaller than zero. Suppose the user rotates the device counter-clockwise. Then, ψ will gradually grow as ϕ decreases. At the timing that ψ becomes larger than zero, we notice that the current azimuth between the sender and the AR device is approximately $\frac{\pi}{2}$. From that moment, VisIoT starts to estimate the azimuth change by integrating the angular velocity on Z-axis measured by the gyroscope (*i.e.*, yaw). Until the user stops rotating, it keeps updating the estimated azimuth of the sender as $\phi = \frac{\pi}{2} + \Delta\phi$, where $\Delta\phi$ is the azimuth change from the gyroscope. Likewise, when the sender is initially at the right side of the user, it first detects the timing that ψ turns from positive to negative value while the device is rotating clockwise, and estimates the current azimuth by adding $\frac{\pi}{2}$ to the azimuth change from that time. One might wonder how much the user needs to rotate the device. It depends on the angle of view of the camera. If the sign of ψ does not change while the rotation of the device is larger than the horizontal viewing angle, it means the sender is outside of the camera angle, so it is unnecessary to track it. Note that the horizontal angle of view of MS Surface 2017 is 35°, which is not difficult for a user to rotate that amount.

Overall, the camera rotation based azimuth tracking works as follows. The user rotates the device clockwise larger than the camera viewing angle, and VisIoT detects the sender whose phase difference changes from positive to negative value. Given the detection, it starts to integrate the gyroscope output on Z-axis and continuously monitor the azimuth change. The camera rotates back to the original angle, and keeps rotating counter-clockwise, until the rotation angle becomes larger than the camera viewing angle. Using the same method, the wireless senders at the left side are identified. Finally, when the device stops rotating, VisIoT can identify the current azimuth of all of the wireless senders within the camera angle.

Figure 6 (a) shows one of our experimental results to illustrate how our azimuth estimation algorithm works. The sender is 2-meter away, and the initial azimuth is 104°. The user rotates the device counter-clockwise, and moves back to the original rotation. At 1.4 second, ψ becomes zero, and VisIoT starts to estimate the azimuth using gyroscope. When the rotation stops, it estimates the current azimuth as 108°, which is 1.1° different from the ground-truth.

[2]Here, $\psi = \angle r_1 - \angle r_2$, where r_1 and r_2 are signals from two antennas that are at the left and the right sides of the camera, respectively.



(a) Due to the symmetry of the signal, the AR device cannot distinguish whether the signal is coming from P_1 or P_2.

(b) If $\phi_1 > \phi_2$, $P = P_1$.

(c) If $\phi_2 > \phi_1$, $P = P_2$.

Figure 7: Illustration on how to find the true position P using the device rotation information.

Elevation estimation: Given the estimated azimuth and measured ψ during the rotation, one straightforward way of estimating the elevation is calculating it using Equation 4 (*i.e.*, $\theta = \arcsin(\frac{\psi\lambda}{2\pi D\cos\phi})$). However, it is very susceptible to phase noise, especially in low elevation (*i.e.*, θ close to $\frac{\pi}{2}$). Figure 6 (b) shows two phase differences while the device is rotating, where θ is 90° and 80°, respectively. The two curves are very similar so it is difficult to distinguish them using the measured ψ[3].

Therefore, we develop an elevation estimation method using another motion change of the AR device. Here, the user further rotates the device along the Y-axis, which changes the elevation. Given the rotation, our observation is that ψ is maximized when there is no elevation (*i.e.*, $\theta = \frac{\pi}{2}$). To take advantage of it, we detect the angle that maximizes ψ, and find θ from it. Initially, the device is placed parallel to the horizontal plane. The user rotates the device along the Y-axis, and rotates back to the original position. During that, we track the angle change of the device in Y-axis using gyroscope, (*i.e.*, *Roll*, denoted as ρ), as well as ψ. By detecting the time that ψ is maximized, we can find ρ that makes $\theta = \frac{\pi}{2}$. Note that, however, ρ itself is not the elevation angle that we need to estimate, but the angle between the device and the sender along the Y-axis, when the sender position is projected onto the xz-plane (*i.e.*, $\rho = \arctan(\frac{z}{x})$). Meanwhile, $\theta = \arccos(\frac{z}{r})$. Therefore, we need to calculate θ from ρ. $\frac{z}{r}$ can be written as $\frac{z}{r} = \sqrt{\frac{z^2}{x^2+y^2+z^2}} = \sqrt{(\frac{x}{z})^2 + (\frac{y}{z})^2 + 1}^{-1}$. From ρ and ϕ, we find that $\frac{z}{x} = \tan\rho$ and $\frac{y}{x} = \tan\phi$, respectively. Combining ρ and ϕ, we get $\frac{y}{z} = \frac{\tan\phi}{\tan\rho}$. Substituting them into the equation of $\frac{z}{r}$, we get:

$$\theta = \arccos(\sqrt{\tan\rho^{-2} + (\frac{\tan\phi}{\tan\rho})^2 + 1}^{-1}). \quad (5)$$

Figure 6 (c) shows how ψ and θ change according to the rotation of the device. Here, the initial θ is 79 °, and it increases as the device rotates along the Y-axis. From the phase measurement, we find that AoA is maximized when $\rho = 17.5°$. Then, we estimate the initial elevation using Equation 5, which yields 1.5° estimation error in this experiment.

Wireless sender visualization: Once the azimuth and the elevation estimation are complete, VisIoT tracks the position of the sender in the camera coordinates using Equation 3, and draws a circle at the location of the wireless sender on the AR device screen. Note that if the sender is out of the viewing angle of the camera, (u, v) values from Equation 3 becomes infeasible (*i.e.*, either smaller than zero or larger than the number of pixels of the video). Then, we realize the sender is out of the viewing angle and filter it out.

2.3 Addressing the ambiguity of angle

For practical system design, we regard that the AR device has only two antennas with linear alignment. A limitation of such a linear antenna array is it can only detect the angle in range of 0° to 180°, which implies it cannot distinguish whether the signal is coming from the front-side or the rear-side [13]. Figure 7 (a) illustrates it[4]. Azimuth is inherently estimated by observing the difference of the travel distance of the received signals from the two antennas. For the two positions $P_1 = (x, y)$ and $P_2 = (x, -y)$, the pairs of the two received signals have the same travel distances, which yield the same phase difference. As a result, given ψ, the true position can be either P_1 or P_2.

With the support of the camera motion tracking, however, finding the true position is not difficult. We observe that the true and fake angles grow in different directions while the device is rotating, which is illustrated in Figure 7 (b) and (c). When the user rotates the device counter-clockwise, if the wireless sender is in front of the user, ϕ decreases as much as the device rotates. On the other hand, if the wireless signal is coming from behind the user, the counter-clockwise device rotation increases ϕ[5]. Since we are only interested in tracking the wireless sender within the camera angle of view, we filter out the signals from behind the user. The same method is also applicable for the clockwise rotation (*i.e.*, filtering out the nodes with decreasing ϕ during the rotation).

[3]The reason is because $sine$ function has low slope near $\frac{\pi}{2}$. In such a case, a small ψ measurement error of 0.01 may cause the elevation estimation error of 10°. For example, the output of the $sine$ function when angles are 90° and 80° are 1 and 0.984, respectively.

[4]For ease of illustration, Figure 7 considers 2D space with no elevation, where $z = 0$. However, the idea here holds for any arbitrary elevation.

[5]Note that the values of ϕs in Figure 7 are all positive.

Note that in 3D tracking, both azimuth and elevation estimation cause ambiguity, where the number of possible positions given the phase measurement is 4. Like the azimuth ambiguity introduced in Figure 7, given the phase same phase difference, the height of the sender from the receiver can be either z or $-z$. VisIoT addresses it by iteratively resolving the azimuth and elevation ambiguity issues, respectively. Given the rotation of the device along Z-axis, it first finds the true azimuth angle using the method in Figure 7. Then, again, by observing the phase change during the rotation of the device on Y-axis, it determines whether the sender is above or below the receiver (*i.e.*, z or $-z$), and finds the true elevation.

2.4 Filtering NLOS signal

IoT system consists of a large number of wireless devices, where not all of them are within the viewing angle of the camera. Trying to track all of the received wireless signals incurs not only unnecessary processing overhead but also false visualization. This section explains how VisIoT filters out the sender in NLOS using the received signal and the camera motion information. Note that wireless senders that are in Line-of-Sight (LOS) but outside of the camera angle are filtered out as explained in § 2.2 and § 2.3.

The main idea of distinguishing LOS and NLOS is monitoring how consistently Θ changes. During the azimuth estimation phase in § 2.2, we compare the difference between Θ from the phase difference and the azimuth change $\Delta\phi$ from the gyroscope while the user is rotating the device along the Z-axis. As shown in Equation 4, when the elevation is fixed, the rotation of the device yields similar curves for Θ and ϕ. If the sender is in LOS, the impact of the direct-path signal will be dominant, where the difference between Θ and ϕ will be bounded. On the other hand, if the direct-path is blocked (*i.e.*, NLOS), the signals from multiple reflected paths will determine the phase, which yields noisy Θ estimation. As a result, the estimated AoAs during the device rotation will be highly varying as well as inconsistent with $\Delta\phi$. Considering them, given the sequence of N AoAs and azimuth samples collected during the rotation, Θ_k and $\Delta\phi_k$, we define the following criterion to filter out NLOS signals:

$$\frac{1}{N}\sum_{k=1}^{N} \|\Theta_k - \Delta\phi_k - (\mu_\Theta - \mu_{\Delta\phi})\|^2 > \sigma, \tag{6}$$

where μ_Θ and $\mu_{\Delta\phi}$ are the means of Θ and $\Delta\phi$, respectively, and σ is NLOS detection threshold. Since $\Delta\phi$ does not include the initial azimuth, $\mu_\Theta - \mu_{\Delta\phi}$ is subtracted to compensate the difference between them. In our system design, we use $\sigma = 0.1$.

2.5 Keep tracking the device

After the device position is identified as introduced in Section 2.2, the user might change the orientation of the camera or move in the area, which will alter the position of the wireless source in the camera image. This section explains how to maintain the tracking and continuously visualize the wireless source given the motion change of the camera.

2.5.1 Tracking rotation. The camera rotates when the user tries to watch different angle of the view in the AR device screen. When the position of the wireless device is already tracked, finding the adjusted position of it upon the rotation of the camera is not difficult. It is because gyroscope provides accurate estimation on the rotation angle. Given the tracking of the wireless source in advance, suppose the user further rotates the device along the arbitrary axis. Let α, β, and γ be the amount of the angle rotation on X, Y, and Z axis, respectively. Each of them can be estimated by integrating the angular velocity from the gyroscope on the corresponding axis. Given the estimation of ϕ and θ before the rotation, we can find the position of the wireless sender after the rotation using 3D rotation matrix [11]:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

where (x, y, z) and (x', y', z') are the sender location before and after the rotation, respectively (*cf.*, $(x, y, z) = (r\cos\phi\sin\theta, r\sin\phi\sin\theta, r\cos\theta)$. Then, its position in the 2D image (u', v') is calculated by $(u', v') = (f\frac{x'}{y'}, f\frac{z'}{y'})$. Here, estimating r is unnecessary because it is canceled out when x' and z' are divided by y'.

2.5.2 Tracking linear transformation. When the AR user changes the position, the camera coordinates are linearly transformed [11]. Compared to the rotation tracking, linear transformation tracking is more challenging. It is because, unlike gyroscope, accelerometer is very noisy and it is difficult to accurately estimate the position change using it [24]. In VisIoT, using the accelerometer, we first detect the rough direction of the movement whether the user is moving forward, backward, left, right, up, or down. Then, depending on the direction, we use different approaches of tracking.

Tracking forward and backward movement: If the user moves backward or forward, it causes linear transformation of the camera position along the Y-axis. To track the device position, we need to estimate the azimuth and elevation after the movement. Let ϕ' and θ' be the new azimuth and elevation angles after the user moves backward or forward. In Y-axis movement, one important condition is that the x and z values of the wireless sender position do not change in 3D world coordinates. This relates the previous angles to the new angles as follows (*c.f.*, Equation 2):

$$r\cos\phi\sin\theta = r'\cos\phi'\sin\theta' \tag{7}$$

$$r\cos\theta = r'\cos\theta', \tag{8}$$

where r' is the new distance from the camera to the wireless sender after the user has moved. Equation 8 implies that if we get $\frac{r'}{r}$, we can calculate θ'. However, the problem is neither r nor r' is known. We found that the phase difference is useful to estimate $\frac{r'}{r}$. Equation 7 gives us $\frac{r'}{r} = \frac{\cos\phi\sin\theta}{\cos\phi'\sin\theta'}$. From the ratio of the phase differences, we also get the same relation: $\frac{\psi}{\psi'} = \frac{\cos\phi\sin\theta}{\cos\phi'\sin\theta'} = \frac{r'}{r}$, where ψ' is the new phase difference after the user moves (*c.f.*, Equation 4). Substituting it into Equation 8, we can get $\theta' = \arccos(\frac{\psi}{\psi'}\cos\theta)$.

Tracking up and down movement: Likewise, given the up and down movement of the user, we can track the position of the sender exploiting that x and y remain the same. In fact, it is even easier than the previous case, because the movement along the Z-axis does

not alter the azimuth angle. Since the azimuth is the angle in the xy-plane to the X-axis (*i.e.*, $\phi = \arctan(\frac{y}{x})$), having different position in z does not affect ϕ. Then, the movement gives us a new elevation θ' and a new phase difference ψ', which have the following relation: $\frac{\psi'}{\pi} = \cos\phi \sin\theta'$. Since ϕ is given from the previous estimate, we can calculate θ' using the measured ψ', and eventually track the new position of the sender.

Tracking left and right movement: With the movement along the X-axis, y and z of the wireless sender position remain the same. In this case, the horizontal position of the sender in 2D camera coordinates, v, does not change regardless of the user's movement (*i.e.*, $v = f\frac{z}{y}$). This gives us an important condition on ϕ' and θ': $\frac{f}{v} = \sin\phi' \tan\theta'$, where v is given from the previous tracking. It can be re-written as $\sin\theta' = \frac{\frac{f}{v\sin\phi'}}{\sqrt{1+(\frac{f}{v\sin\phi'})^2}} = \Phi(\phi')$. By substituting it into $\psi' = \frac{2D\pi}{\lambda}\cos\phi'\sin\theta'$, we get $\psi' = \frac{2D\pi}{\lambda}\cos\phi'\Phi(\phi')$. Since it is difficult to further derive closed form solution, we estimate it by finding ϕ' that minimizes the difference between ψ' and $\frac{2D\pi}{\lambda}\cos\phi'\Phi(\phi')$.

Limitation: The tracking methods introduced in this subsection require linear transformation of the camera motion aligned to a single axis, which limits the movement of the user. Also, if the rotation and linear transformation of the camera are mixed (*i.e.*, rigid body transformation), the solution in this section is not applicable. In such cases, we need to re-track the wireless sender position using the method in Section 2.2. Another possible solution is using computer vision techniques. Given the identification of the wireless senders with our technique, updating the position of them with object tracking techniques such as SIFT [25] and SURF [26] is not difficult. However, it is beyond the scope of this paper and we leave it for future work.

2.6 Putting it all together

This section provides a brief summary on how VisIoT works. Suppose a user with an AR device is moving in the area with IoT infrastructure. Given the need of visualizing the IoT devices, the AR device first broadcasts an initiation message. Upon receiving it, nearby wireless devices increase the wireless frame delivery rate enough to track them. The impact of the frame interval on the tracking accuracy is evaluated in §4.2. Then, the user rotates the device along the Z-axis. Using the collected wireless signals and the tracked camera rotation, VisIoT estimates the azimuth angles between the camera and the IoT devices (§2.2). The IoT nodes in NLOS and behind the user are filtered out using the methods introduced in §2.4 and §2.3, respectively. The user further rotates the device along the Y-axis, where the elevation angle is estimated (§2.2). Using the estimated azimuth and elevation, VisIoT tracks the location of IoT devices (§2.1), and visualizes the position of them in the AR device screen along with the data they sent. If the user clicks the tracked device on the screen, the AR device can send a message to it. If the user moves the position, VisIoT continuously updates the position using the algorithms introduced in §2.5.

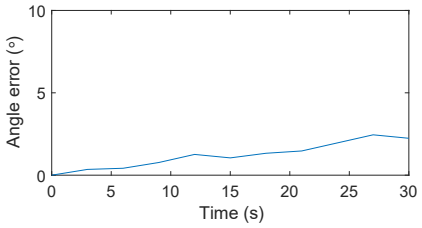


Figure 8: Gyroscope based rotation tracking error over time.

3 IMPLEMENTATION

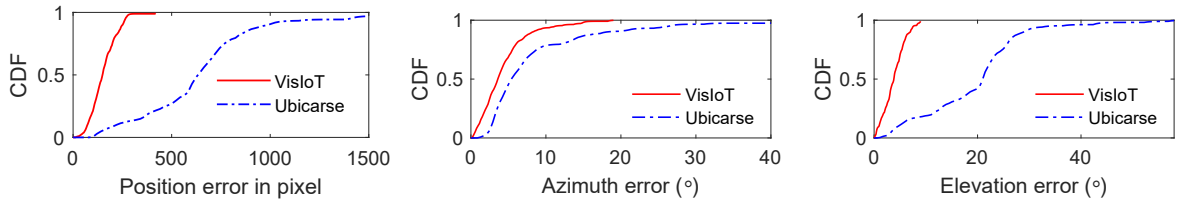
We build VisIoT as Windows application program developed in C++. It gathers data from camera, IMU sensor, and USRP software radio, implements the tracking algorithms in Section 2, and visualizes the location of the ZigBee sender in real time. In addition, it stores all of the collected data so that we can run microscopic benchmark in offline.

Camera: Using OpenCV 3.4.2 library, VisIoT accesses the camera of the tablet and renders the video to the screen. Also, it collects the ground-truth position of the ZigBee node in the camera image. We attached a color marker to the node and developed a simple color tracking program, which provides the ground-truth position of the ZigBee node in camera 2D coordinates (*i.e.*, (u, v)). In addition, using Equation 3 and f, we calculate the ground-truth azimuth and elevation angles between the Zigbee node and the camera: $\phi = \tan^{-1}(\frac{f}{u})$ and $\theta = \tan^{-1}(\frac{f\sin\phi}{v})$.

IMU sensor: To access sensor values from gyroscope and accelerometer, we use HID Sensor Collection V2 driver and Microsoft Sensor APIs [27]. The driver periodically updates the sensor values through the APIs when the values exceed the threshold. For the rotation tracking, we integrate the angular velocity of the gyroscope in each axis. We observed that the gyroscope of MS surface 2017 is reliable, where VisIoT does not severely suffer from the drift problem. Figure 8 shows the average rotation tracking error over time while the device is repeatedly rotating left and right within 90°. We measure the angle error by comparing the azimuth change calculated from gyroscope with that from the video [6]. The result shows the drift is not significant, which slowly increases the error. Considering that the maximum device rotation time for visualization is 18 seconds (*cf.*, §4.2), the error growth due to the drift does not severely degrade the tracking performance. Note that there have been approaches to compensate the gyroscope drift by adaptive filtering [28, 29]. While we believe adopting these techniques will improve the reliability of VisIoT, we leave it as future work. For the linear transformation tracking in §2.5.2, we use accelerometer. We integrate the output of the linear accelerometer, and detect the movement if the velocity is higher than $0.5m/s$. Then, we determine whether the direction is left-right, forward-backward, or up-down, by finding the axis with the maximum speed.

RF module: To collect raw RF signals from the software radio, we use USRP Hardware Driver (UHD) 3.11.1.0. From the two antenna ports, IQ samples are delivered with 4 MHz sampling rate for each antenna. Using UCLA ZigBee PHY [30], VisIoT decodes the ZigBee frames in real time. Upon the successful frame decoding, it estimates

[6] Here, the tablet is attached to a tripod so that it can rotate in a fixed position.



(a) Position error in pixel. (b) Azimuth estimation error. (c) Elevation estimation error.

Figure 9: CDFs for tracking accuracy comparison between VisIoT and Ubicarse.

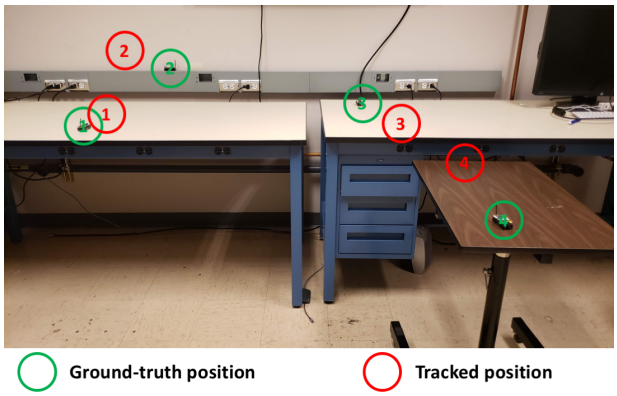


Figure 10: Example of the position error.

the phase difference of the signals from the two antennas. Once the ZigBee node position is identified and visualized, the user can send a ZigBee data frame to the node by clicking the circle that visualizes the location. Receiving the frame, the node blinks the LED light so that the user can easily find where the IoT device is in real world. The data frame is also encoded with UCLA ZigBee PHY library, and fed into the software radio RF chain through UHD. In our testbed system, the average ZigBee frame encoding and decoding times are 0.015ms and 0.26ms, respectively. The computational cost of our tracking algorithm is very low, which takes less than 1ms to find the location of the sender.

4 PERFORMANCE EVALUATION

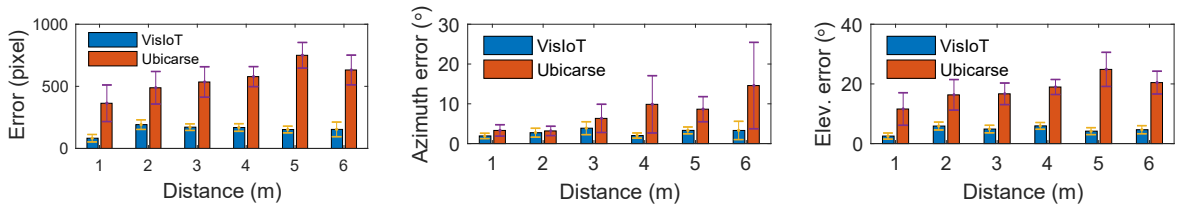
We conducted our experiments in an office area with surrounding desks, chairs, cubicles, PCs, and many other small electronic devices. As mentioned, Microsoft Surface 2017 with USRP B210 and MicaZ ZigBee motes are used as an AR device and IoT wireless sender devices, respectively. 2.48 GHz channel is used for ZigBee communication. By default, it sends data frame with 20ms interval. This is considering a scenario where at least 10 IoT devices are sending frames without severe interference and collision. We evaluated the impact of the traffic intensity on the tracking accuracy while varying the frame interval from 20ms to 100ms. For data collection, we recruit 5 users, where four are men and one is woman. For 10 minutes, we trained them how to rotate the device to visualize the data. We also conducted user study to understand how the users perceive the visualization quality. The resolution of the video is 1,920 × 1,080 pixels.

Compared scheme: For the performance comparison, we also implemented the 3D tracking algorithm of Ubicarse [23]. Using the rotation of the device, it emulates Synthetic Aperture Radar (SAR), and estimates the azimuth and elevation. Note that the main application of Ubicarse is indoor localization, which finds the position of the mobile device in world coordinates. By itself, it is difficult to localize the wireless source in the video. Even if Ubicarse can accurately locate the AR device in 3D coordinates with infrastructure support (*i.e.*, known position of the 3 - 5 WiFI APs), if the 3D positions of the all of the IoT devices are not given in addition, it is not possible to identify the location of them in the video. Therefore, we only implement the 3D tracking algorithm of Ubicarse (§4.3 in [23]), and visualize the wireless source with our technique.

4.1 Localization Accuracy

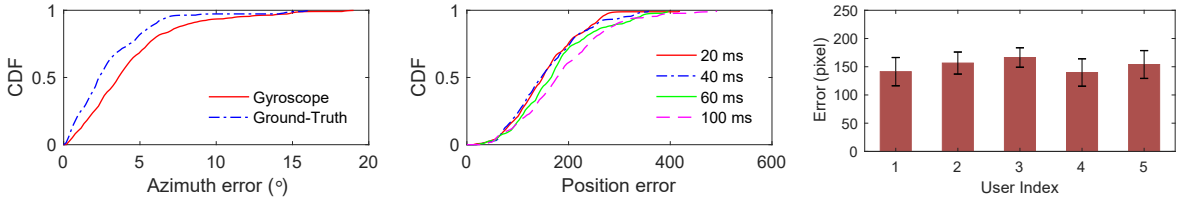
We first evaluate how accurately VisIoT finds the position of the wireless sender. For the performance evaluation, we performed experiments while varying the position of the MicaZ motes. The distance of the mote from the AR device is between 1m to 6m. In each experiment, we tracked the position of 4 devices in parallel. Figure 9 (a) shows the CDF of the position error with 300 experiments. VisIoT achieves accurate tracking with the median and the 95-percentile pixel errors of 134 and and 239 pixels, respectively. When normalized by number of pixels in diagonal line (*i.e.*, 2,203 pixels in 1,920 × 1,080 pixels video), they correspond to 6% and 11% of errors. On the other hand, Ubicarse yields much higher position error, with the median and the 95-percentile errors of 594 and 1,387 pixels, respectively. Figure 10 shows the example cases of the position error so that the reader can get a feel on the impact of the tracking error on the user experience, where the positions errors of device 1, 2, 3, and 4 are 90, 156, 139, and 224 pixels, respectively.

Figure 9 (b) and (c) show the CDFs of the azimuth and elevation estimation errors, respectively. While VisIoT and Ubicarse are comparable in terms of azimuth estimation, there is huge difference in elevation estimation. Basically, it is difficult to estimate elevation based on Z-axis rotation of the device, because what changes during the rotation is not the elevation but the azimuth. Although it is possible to estimate the elevation using the relation among AoA, azimuth, and elevation, (*i.e.*, $\Theta = \arccos(\cos\phi \sin\theta)$), it is very susceptible to noise. On the other hand, VisIoT exploits Y-axis rotation to estimate elevation, where the elevation change is proportional to the device rotation. The reason that the elevation estimation performance of Ubicarse is not as good as shown in [23] is because of the small elevation in our experimental environment (*i.e.*, the



(a) Position error. **(b) Azimuth estimation error.** **(c) Elevation estimation error.**

Figure 11: Tracking evaluation in various distances.



(a) Azimuth estimation error with ground-truth rotation. **(b) Position error while varying the frame interval.** **(c) Position errors of 5 users.**

Figure 12: Microbenchmark results.

sender and the receiver are in similar heights). As illustrated in §2.2 and Figure 6 (b), *sine* function has low slope near 90°, where non-negligible angle difference (*e.g.*, 10°) causes very slight difference in *sine* output (*e.g.*, 0.01). Since we consider the scenario where the wireless senders are within the camera angle, all of the senders in this experiment are within 17° from 90° in terms of elevation. In such cases, a slight error in ψ can yield significant error in θ, which results in low elevation tracking accuracy of Ubicarse. Meanwhile, in the experiments of [23], APs are presumably attached to the ceiling, where Ubicarse tracks the elevation more reliably, according to [23]. Also, note that the tracking here is based on single sender and receiver, while Ubicarse uses 3 - 5 anchor points in [23].

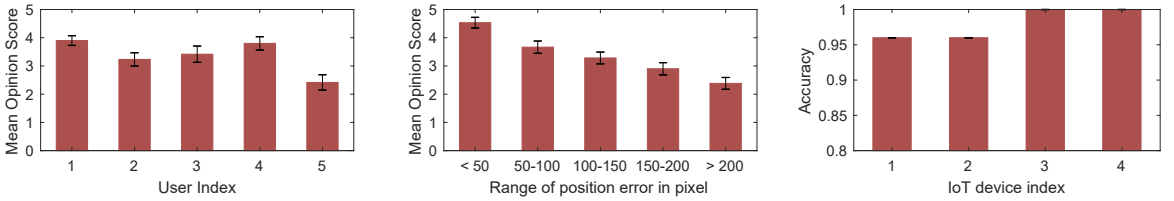
Figure 11 shows the mean position errors and angle estimation errors with 95% confidence interval in various distances. We observe that the error of VisIoT marginally grows as the distance increases from 1m to 6m. Note that due to the low-power principle of ZigBee, the transmission power of the MicaZ mote is fixed to 0 dBm. With our USRP receiver, the SNR of the received signal in 6m distance is only around 1 dB, where UCLA Zigbee decoder [30] drops more than 50% of the frames either due to the preamble detection failure or CRC checksum error. Even in such an environment, VisIoT can reliably track the sender and visualize it. On the other hand, Ubicarse is more sensitive to SNR. Further increasing the distance makes most of the frames undecodable. We believe using high power wireless techniques such as WiFi can extend the range of VisIoT.

4.2 Micro benchmark

Using the data traces collected from the experiments in the previous subsection, we further analyze the performance in microscopic viewpoint.

Impact of gyroscope error: In our azimuth and elevation estimation algorithms, gyroscope based camera rotation tracking is as important as phase based AoA tracking. Here, we analyze how much estimation error is caused by the imperfect gyroscope. From the collected traces in §4.1, we replace the gyroscope based rotation tracking with the ground-truth azimuth change from the video. Figure 12 (a) shows the performance comparison. It demonstrates that the phase measurement error affects more on the tracking accuracy than the gyroscope error. When the ground-truth rotation is used, the median azimuth estimation error reduces from 3.2° to 2.1°. Note that the azimuth estimation error is not simple summation of gyroscope error and phase error. We find that in a fraction of data, gyroscope based tracking even yields smaller error than ground-truth based tracking. It is because gyroscope error and phase error are opposite, which reduces the error. Meanwhile, gyroscope based tracking yields higher errors in the worse cases. The 95th percentile position errors of ground-truth and gyroscope rotations are 6.6° to 11.1°, respectively. From the recorded video, we find these are the cases where users make improper movement during the rotation and cause unnecessary linear transformation of the device. In elevation estimation, we also observed similar result regarding the gyroscope error, but we skip presenting it due to the limit of space.

Impact of frame interval: While low-power IoT devices send data frames infrequently, VisIoT requires a certain amount of data traffic to track the phase change during the rotation. If the traffic requirement is too high, it not only increases the energy consumption of the IoT devices, but also incurs collision and interference when the device density is high. Figure 12 (b) shows the CDFs of the position error while varying the frame interval. From the collected traces, we evaluated the tracking accuracy while adjusting the frame interval. The result in Figure 12 (b) demonstrates VisIoT does not require heavy data traffic. Increasing the frame interval from 20ms to 40ms hardly affects the tracking error. When the



(a) MOS of 5 users.

(b) MOS across different position errors.

(c) Probability that the user identifies the correct device given the visualized device position.

Figure 13: User study results.

frame interval reaches 100*ms*, the median position error increases from 134 pixels to 161 pixels.

Time to rotate the device: In our experiments, users spend 9.04 seconds in average to rotate the device in Z-axis, with the standard deviation of 1.18 seconds. Also, they spent 9.41 seconds for Y-axis rotation in average with the standard deviation of 1.37 seconds. The total amount of time to rotate the device for wireless sender visualization is 18.5 seconds in average. Note that this is the case of visualizing all of the senders within the camera angle. When the user already knows the location of the sender in real world and only focuses on visualizing it, the tracking can be done with smaller rotation in 3 - 5 seconds. Also, once the sender location is tracked, VisIoT can continuously track the position of the senders using the method in §2.5. We classified the position error with respect to the rotation time, but we have not observed high correlation between the rotation time and the tracking performance.

Result from different users: Figure 12 (c) individually shows the mean position errors of 5 users. The user 2 is a female user while the other users are male. The result shows that users achieve similar accuracy.

4.3 User Study

To understand how users perceive the quality of the IoT visualization, we conducted user study.

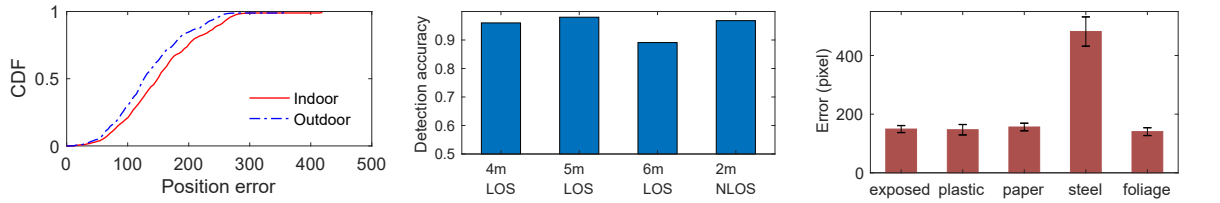
Mean Opinion Score: Since it is difficult to quantify the impact of the position error on the visual perception quality, we collected the Mean Opinion Score (MOS) from the 5 users who participated in the tracking experiment. MOS has been widely used as an indicator of the media quality such as VoIP and compressed video [31]. Given the visualized position of the IoT device shown in the AR device screen as well as the ground-truth position, we asked the user to evaluate the quality of the visualization from the following ratings: 5 - *excellent*, 4 - *good*, 3 - *fair*, 2 - *poor*, 1 - *bad*. Figure 13(a) shows the MOS for each user. The average score is 3.4, which is between *good* and *fair*. However, the individual score for each user varies from 3.9 to 2.4. One interesting observation from this user study is that even though the tracking performance across the users is consistent (*cf.*, Figure 12 (c)), the visualization quality that each user perceives highly varies. Figure 13(b) shows the MOSs across different amount of position errors. When the error is less than 50 pixels, the MOS is 4.5, and it gradually decreases as the error increases. In case of the median tracking error, the MOS is 3.3.

Matching the tracked and the real device: In AR device screen, if multiple devices are closely located and the data from the devices are concurrently visualized, the position error might induce the user to incorrectly match the visualized data with the IoT device in real world. To understand how the position error affects finding the IoT device, we conducted another experiment that asks the users to select the correct IoT device in real world given the visualization of the 4 devices. Figure 10 shows the experimental environment, where the distances between device 1 and 2, and 3 and 4 are 0.8*m* and 1.3*m*, respectively. In this experiment, the users correctly identifies the device with 98% accuracy. From the experiment, we found that the distance among IoT devices makes critical impact on the accurate device identification. Figure 13(c) shows the identification accuracy for each device. For the device 3 and 4, due to the sufficient spacing between them, the users did not make any wrong identification regardless of the position error. On the other hand, depending on the position error, the users occasionally mis-determines the visualized output of device 1 to device 2, and vice versa. In this user study, we do not observe particular performance variation across users.

4.4 Performance in various environments

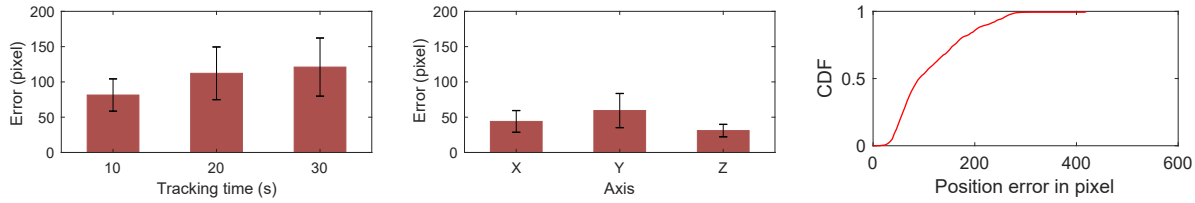
Outdoor experiment: While most of our experiments were done in an office area that easily constructs multi-path fading channel, we performed the same experiment of §4.1 in outdoor environment, where no obstacle exists near the transmitters and receiver. Figure 14(a) shows the CDF of the position error in the outdoor experiment. Compared to the indoor experiment, the normalized median position error reduces from 6.4% to 5.4%. This result demonstrates the multi-path fading of indoor office environment does not severely damage the performance of VisIoT. It can be because VisIoT minimally relies on the wireless signal to locate the wireless sender. Also, visualizing wireless sender inherently requires LOS, where the strong direct-path signal weakens the impact of the multi-path fading.

Identifying NLOS devices: We conduct an experiment with and without obstacle that blocks the direct path between the AR device and the ZigBee node, and evaluate the performance of the NLOS filtering algorithm introduced in §2.4. First, we vary the distance between the AR device and ZigBee node in LOS environment. We evaluate the probability that the path is determined to LOS. The result in Figure 14 (b) shows that when the distance is less than 5*m*, VisIoT can correctly determine LOS with probability higher than 96%. When the distance is 6*m*, 10% of the tests are mis-determined as NLOS, due to the phase estimation noise in the received frame. Again, the range issue is caused by the low power of the ZigBee node



(a) Indoor vs. Outdoor. (b) LOS identification accuracy. (c) Position error when the IoT device is covered.

Figure 14: Performance in various environments.



(a) Tracking performance when the user rotates the AR device. (b) Tracking performance when the user moves the position. (c) Tracking 10 devices.

Figure 15: Evaluation of VisIoT in various aspects.

together with the low signal quality of the USRP ZigBee decoder, and it can be overcome by using WiFi. In NLOS experiment, a 20cm concrete wall blocks the direct path. The distance between the AR device and the ZigBee node is 2m. Further increasing the distance makes most of the ZigBee frames undecodable. Figure 14 (b) shows the probability that the path is correctly determined as NLOS. VisIoT yields accurate NLOS detection with probability higher than 97%. We also conducted an experiment on determining whether the signal is coming from the front side or backward, as introduced in §2.3. During the experiment, we hardly observed the detection error, so we do not show the result here.

Tracking hidden IoT devices: Here, we evaluate the tracking performance of VisIoT when the IoT devices are invisible to user. In this experiment, the ZigBee mote is covered by 1) plastic, 2) paper, 3) steel, and 4) foliage. For plastic, we used the plastic cover of an Aruba WiFi AP-335 that is used to hide the antennas. For the paper and steel experiments, we place the mote in a small paper box and a steel cabinet, respectively. For the foliage experiment, we put the mote behind the leaves of a small plant. The plastic, paper, and steel experiments are done in indoor office environment, while the foliage experiment is conducted in outdoor garden environment. Note that the experimental scenario here is different from NLOS experiment introduced in the previous subsection. The former one shows the feasibility of the visualization when the direct-path is lightly blocked, while the latter one evaluates the filtering performance when it is presumably too difficult to accurately track the device. The performance evaluation result in Figure 14(c) shows that compared to the case when the IoT device is not covered (*i.e.*, *exposed* in the figure), covering it with plastic, paper, and foliage does not degrade the tracking accuracy. These materials are known to not severely degrade the signal strength when they block the signal path [32]. On the other hand, when the IoT device is covered by steel, it yields much higher tracking error.

4.5 Keep tracking the device

Here, we evaluate how VisIoT maintains the tracking after the wireless sender location is identified. Assuming that the sender position is estimated without any error, we evaluate how accurately VisIoT can update the position of the sender after the movement of the camera.

Rotation tracking: In this experiment, the user freely rotates the device in X, Y, and Z axes in a fixed position. The rotation angle is estimated using gyroscope, and the device position is tracked using the algorithm in §2.5. Figure 15 (a) shows the mean position error in various tracking time. We find that the position error gradually increases over time. The error grows because of 1) the drift error of the gyroscope, and 2) the accumulation of the linear transformation of the camera position that is difficult to track with IMU sensor. One possible solution on the drift error is compensating the drift using the wireless signals as reference [23]. We leave it for future work.

Linear transformation tracking: Here, we evaluate the tracking accuracy when the user moves the device along X, Y, or Z-axis with fixed rotation of the camera. Initially, the user is 3-meter away from the sender. For X and Y-axis experiments, the user moves 1m. In Z-axis experiment, the height of the device changes for 50cm, due to the limit of human body. For each axis, we repeated the experiment 50 times. Figure 15 (b) shows the mean position error in each axis. It demonstrates that VisIoT can accurately track the position given the movement of the user. We find that during the movement, ψ slowly changes over the distance change. It makes the tracking algorithm less sensitive to the noise.

4.6 Putting it all together

Finally, we perform an experiment with all things considered. In a $6m \times 10m$ office area, we place 10 ZigBee devices, and let the user

visualize them. The user moves towards the devices up to $1m$ to $2m$ away, and identify the location of them by rotating the device. Then, it moves to another position, where the device position is continuously tracked until it is out of the camera viewing angle. The experiment is completed when all of the devices are visualized. Devices in NLOS and behind the user are filtered out. Figure 15 (c) shows the CDF of the position errors when 5 users conduct the experiment. The median and 95th percentile errors are 83 and 242 pixels, respectively, which is slightly lower than the evaluation in §4.1. The reason is because the user was a bit closer to the device in this experiment, and the device is accurately tracked while the user is moving. Overall, this experiment shows that VisIoT can accurately visualize the IoT devices in a realistic environment with many devices surrounding the user.

5 RELATED WORK

IoT with augmented reality: Making interaction between human and IoT infrastructure using AR has been introduced in previous work [2–7]. They all emphasize it is very promising approach, but most of them just introduce the concept and do not provide a feasible solution to enable it. In [2], Garcia *et al.* propose to use QR code to identify IoT device, but the QR code is readable only in limited range. Also, many IoT devices have forms that are hard to attach QR code. Likewise, Subakti *et al.* [3] use color marker to identify IoT devices, which limits scalability. They also propose to use Bluetooth as invisible marker, where RSSI is used to identify the nearby device. It cannot distinguish multiple devices in similar range. ARIoT [7] uses SURF algorithm to detect the object in the video, and matches it with the IoT devices in the database. It does not specify how to distinguish multiple IoT devices with the same appearance.

Tracking wireless device in 3D space: In order to locate the wireless device within the camera image, we need i) location, and ii) 3D direction (*i.e.*, azimuth and elevation angles of arrival) of the AR device relative to the position of the wireless device. Recently, a few tracking techniques such as CAT [8] and μLocate [9] achieve centimeter-level 3D localization accuracy. They both track the device in 3D space by estimating the distances from 3 or more anchor points, and finding the intersections of the circles whose radii are the estimated distances. Although they provide very reliable location estimation, this alone cannot achieve the goal of IoT visualization, because they cannot identify the direction of arrival (DoA). On the other hand, Ubicarse [23] tracks the device by estimating azimuth and elevation angles, where the wireless source visualization idea of VisIoT is directly applicable. In §4.1, we extensively compared the performance of Ubicarse with VisIoT. Recently, Ioannis *et al.* propose mWaveLoc [33] that enables 3D localization using 60 GHz mmWave communication. Similar to Ubicarse, it estimates the azimuth and elevation angles using the channel impulse responses from the directional transmissions. The median angle estimation error of mWaveLoc is 11°, which is more accurate than Ubicarse. However, considering the directional propagation and the small coverage of the mmWave channel, it is difficult to imagine that mmWave communication will be widely used for IoT devices.

Besides Ubicarse and mWaveLoc, there have been many studies on the the azimuth and elevation estimation for 3D tracking, so called 2D-DoA estimation [17–22]. To separate azimuth and elevation from the AoA measurement, they use specialized antennas such as uniform circular array [17, 20], L-shaped array [18, 19], and parallel linear array [21, 22]. Those are all theoretical approach, which neither considered any system design issue nor performed experiment in real wireless channel.

Combining wireless and IMU sensing: To improve the accuracy of the localization, recent works propose to use IMU sensing in addition to wireless sensing [8, 34–36]. SpinLoc [34] uses gyroscope to determine the true AoA among multiple candidate AoAs from MUSIC algorithm. The idea of SpinLoc is later adopted to CUPID [35] to enable large-scale indoor WiFi localization. CAT [8] enhances the accuracy of the acoustic signal based device tracking by leveraging accelerometer and gyroscope sensing data. Likewise, Epsilon [36] achieves high tracking accuracy by fusing the measurements of light and IMU sensors. These techniques are closely related to VisIoT in that they combine wireless and IMU sensing. However, their main application is indoor device localization while VisIoT focuses on IoT device visualization.

Combining wireless and vision sensing: There have been approaches that combine wireless sensing with computer vision technique to match the received data with the object shown in the video [37–41]. ID-match [37] uses RFID to match the ID with the human recognized in the video. Likewise, TAR [38] exploits BLE signal to analyze the trajectory and match it with the movement of the person in the video. Both techniques rely heavily on computer vision techniques. ID-match uses Kinect RGB-D sensor for human recognition. TAR implements deep learning based human body detection with powerful GPU support. Due to the high computational requirement, the number of simultaneous tracking is limited to one. Also, they consider the scenario that the camera is in a fixed position, which are not applicable to AR where the camera position and orientation changes freely.

6 CONCLUSION

This paper presents VisIoT, a novel system design that accurately tracks the location of the wireless sender in AR user viewpoint. We formulate how to find the location of the device in the camera image using azimuth and elevation angles. Then, we propose lightweight azimuth and elevation estimation algorithms by combining the wireless signal based AoA estimation with the gyroscope based device rotation estimation. We implement VisIoT in commodity tablet and software radio, and visualize ZigBee nodes in real time with high tracking accuracy. We believe the wireless source visualization introduced in this paper can bring many interesting applications and open up new research opportunity on how to combine computer vision and wireless sensing data. A few example applications are: 1) recognizing vehicles that sends V2X message in autonomous driving, 2) multi-user AR games with wireless toy devices, and 3) assisting 3D reconstruction using wireless devices as anchor points. Moving forward, we plan to design a new wireless source visualization system that does not require user motion.

REFERENCES

[1] Roundup Of Internet Of Things Forecasts And Market Estimates, 2016. https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#2b069df0292d.

[2] J Antonio García-Macías, Jorge Alvarez-Lozano, Paul Estrada-Martinez, and Edgardo Avilés-López. Browsing the internet of things with sentient visors. *Computer*, (5):46–52, 2011.

[3] Hanas Subakti and Jehn-Ruey Jiang. A marker-based cyber-physical augmented-reality indoor guidance system for smart campuses. In *2016 IEEE 18th International Conference on High-Performance Computing and Communications, IEEE 14th International Conference on Smart City, and IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1373–1379. IEEE, 2016.

[4] Volker Paelke. Augmented reality in the smart factory: Supporting workers in an industry 4.0. environment. In *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, pages 1–4. IEEE, 2014.

[5] Konstantinos Michalakis, John Aliprantis, and George Caridakis. Visualizing the internet of things: Naturalizing human-computer interaction by incorporating ar features. *IEEE Consumer Electronics Magazine*, 7(3):64–72, 2018.

[6] Paula Fraga-Lamas, Tiago M Fernández-Caramés, Óscar Blanco-Novoa, and Miguel A Vilar-Montesinos. A review on industrial augmented reality systems for the industry 4.0 shipyard. *IEEE Access*, 6:13358–13375, 2018.

[7] Dongsik Jo and Gerard Jounghyun Kim. Ariot: scalable augmented reality framework for interacting with internet of things appliances everywhere. *IEEE Transactions on Consumer Electronics*, 62(3):334–340, 2016.

[8] Wenguang Mao, Jian He, and Lili Qiu. Cat: high-precision acoustic motion tracking. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 69–81. ACM, 2016.

[9] Rajalakshmi Nandakumar, Vikram Iyer, and Shyamnath Gollakota. 3d localization for sub-centimeter sized devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 108–119. ACM, 2018.

[10] Aruba Asset Tracking. https://www.arubanetworks.com/assets/so/SO_AssetTracking.pdf.

[11] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. Spotfi: Decimeter level localization using wifi. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 269–282. ACM, 2015.

[13] Jie Xiong and Kyle Jamieson. Arraytrack: a fine-grained indoor location system. Usenix, 2013.

[14] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 251–258. ACM Press/Addison-Wesley Publishing Co., 1997.

[15] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[16] Jue Wang, Deepak Vasisht, and Dina Katabi. Rf-idraw: virtual touch screen in the air using rf signals. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 235–246. ACM, 2014.

[17] Cherian P Mathews and Michael D Zoltowski. Eigenstructure techniques for 2-d angle estimation with uniform circular arrays. *IEEE Transactions on signal processing*, 42(9):2395–2407, 1994.

[18] Zhang Xiaofei, Li Jianfeng, and Xu Lingyun. Novel two-dimensional doa estimation with l-shaped array. *EURASIP Journal on Advances in Signal Processing*, 2011(1):50, 2011.

[19] Nie Xi and Li Liping. A computationally efficient subspace algorithm for 2-d doa estimation with l-shaped array. *IEEE signal processing letters*, 21(8):971–974, 2014.

[20] Yuntao Wu and Hing Cheung So. Simple and accurate two-dimensional angle estimation for a single source with uniform circular array. *IEEE Antennas and Wireless Propagation Letters*, 7:78–80, 2008.

[21] Tieqi Xia, Yi Zheng, Qun Wan, and Xuegang Wang. Decoupled estimation of 2-d angles of arrival using two parallel uniform linear arrays. *IEEE Transactions on Antennas and Propagation*, 55(9):2627–2632, 2007.

[22] Lu Gan, Jian-Feng Gu, and Ping Wei. Estimation of 2-d doa for noncircular sources using simultaneous svd technique. *IEEE Antennas and Wireless Propagation Letters*, 7:385–388, 2008.

[23] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. Accurate indoor localization with zero start-up cost. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 483–494. ACM, 2014.

[24] Sangki Yun, Yi-Chao Chen, and Lili Qiu. Turning a mobile device into a mouse in the air. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 15–29. ACM, 2015.

[25] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[26] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[27] Microsoft Sensor API. https://docs.microsoft.com/en-us/windows/desktop/sensorsapi/portal.

[28] Hyung-Jik Lee and Seul Jung. Gyro sensor drift compensation by kalman filter to control a mobile inverted pendulum robot system. In *2009 IEEE International Conference on Industrial Technology*, pages 1–6. IEEE, 2009.

[29] Sheng Shen, Mahanth Gowda, and Romit Roy Choudhury. Closing the gaps in inertial motion tracking. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 429–444. ACM, 2018.

[30] Thomas Schmid, L Choong, and S Leidelöf. Ucla zigbee phy, 2009.

[31] Robert C Streijl, Stefan Winkler, and David S Hands. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227, 2016.

[32] Robert Wilson. Propagation losses through common building materials 2.4 ghz vs 5 ghz. *Magis Networks Inc.: San Diego, CA, USA*, 2002.

[33] Ioannis Pefkianakis and Kyu-Han Kim. Accurate 3d localization for 60 ghz networks. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 120–131. ACM, 2018.

[34] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. Spinloc: Spin once to know your location. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, page 12. ACM, 2012.

[35] Souvik Sen, Jeongkeun Lee, Kyu-Han Kim, and Paul Congdon. Avoiding multipath to revive inbuilding wifi localization. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 249–262. ACM, 2013.

[36] Liqun Li, Pan Hu, Chunyi Peng, Guobin Shen, and Feng Zhao. Epsilon: A visible light based positioning system. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 331–343, 2014.

[37] Hanchuan Li, Peijin Zhang, Samer Al Moubayed, Shwetak N Patel, and Alanson P Sample. Id-match: A hybrid computer vision and rfid system for recognizing individuals in groups. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4933–4944. ACM, 2016.

[38] Xiaochen Liu, Yurong Jiang, Puneet Jain, and Kyu-Han Kim. Tar: Enabling fine-grained targeted advertising in retail stores. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, pages 323–336. ACM, 2018.

[39] Francesco Cafaro, Alessandro Panella, Leilah Lyons, Jessica Roberts, and Josh Radinsky. I see you there!: developing identity-preserving embodied interaction for museum exhibits. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1911–1920. ACM, 2013.

[40] Michael Goller, Christoph Feichtenhofer, and Axel Pinz. Fusing rfid and computer vision for probabilistic tag localization. In *RFID (IEEE RFID), 2014 IEEE International Conference on*, pages 89–96. IEEE, 2014.

[41] Zhongqin Wang, Min Xu, Ning Ye, Ruchuan Wang, and Haiping Huang. Rf-mvo: Simultaneous 3d object localization and camera trajectory recovery using rfid devices and a 2d monocular camera. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 534–544. IEEE, 2018.