BatTracker: High Precision Infrastructure-free Mobile Device Tracking in Indoor Environments

Bing Zhou[1], Mohammed Elbadry[2], Ruipeng Gao[3], Fan Ye[1]
[1] Department of Electrical and Computer Engineering, Stony Brook University
[2] Department of Computer Science, Stony Brook University
[3] School of Software Engineering, Beijing Jiaotong University
{bing.zhou,mohammed.salah,fan.ye}@stonybrook.edu,rpgao@bjtu.edu.cn

ABSTRACT

Continuous tracking of the device location in 3D space is a popular form of user input, especially for virtual/augmented reality (VR/AR), video games and health rehabilitation. Conventional inertial based approaches are well known for inaccuracy caused by large error drifts. Computer vision approaches can produce accuracy tracking but have privacy concerns and are subject to lighting conditions and computation complexity. Recent work exploits accurate acoustic distance measurements for high precision tracking. However, they require additional hardware (e.g., multiple external speakers), which adds to the costs and installation efforts, thus limiting the convenience and usability. In this paper, we propose *BatTracker*, which incorporates inertial and acoustic data for robust, high precision and *infrastructure-free* tracking in indoor environments. BatTracker leverages echoes from nearby objects and uses distance measurements from them to correct error accumulation in inertial based device position prediction. It incorporates Doppler shifts and echo amplitudes to reliably identify the association between echoes and objects despite noisy signals from multi-path reflection and cluttered environment. A probabilistic algorithm creates, prunes and evolves multiple hypotheses based on measurement evidences to accommodate uncertainty in device position. Experiments in real environments show that BatTracker can track a mobile device's movements in 3D space at sub-*cm* level accuracy, comparable to the state-of-the-art infrastructure based approaches, while eliminating the needs of any additional hardware.

CCS CONCEPTS

•Human-centered computing →Ubiquitous and mobile computing systems and tools;

KEYWORDS

device tracking, mobile sensing, acoustics, infrastructure-free

SenSys '17, Delft, Netherlands

DOI: 10.1145/3131672.3131689

ACM Reference format:
Bing Zhou[1], Mohammed Elbadry[2], Ruipeng Gao[3], Fan Ye[1]. 2017. BatTracker: High Precision Infrastructure-free Mobile Device Tracking in Indoor Environments. In *Proceedings of SenSys '17, Delft, Netherlands, November 6–8, 2017,* 14 pages.
DOI: 10.1145/3131672.3131689

1 INTRODUCTION

Tracking the continuous movements thus locations of a device in an indoor space (e.g., part of a room area, or in front of a TV/game console) is a form of human-computer interaction, popular in many applications including virtual/augmented reality (VR/AR), video games and health rehabilitation. Using inertial sensors (e.g., embedded in the device) is the most straightforward approach. However, they suffer from large drift errors over time [15, 22]. To combat such errors, additional infrastructure is usually installed. The state-of-the-art VR systems (e.g., HTC vive [1], Oculus [5]) rely on multiple base stations or cameras to track users in a small area (HTC vive recommended playable area is $3.5m^2$, and Oculus recommends a maximum playable area of $2.5 \times 2.5m^2$ with three-sensor roomscale setup); they increase the cost by several hundred dollars and require installation efforts. Some computer vision based approaches may achieve high tracking accuracy [3, 8, 34]. A representative one, the Microsoft Kinect [3], leverages a depth sensor with a range of $0.8 - 4m$. However, they require a visually distinct target, and their performance is subject to lighting conditions. Besides, cameras may raise privacy concerns. RF signals such as Wi-Fi has been widely used for tracking, however they usually require customized hardware and have a limited accuracy due to the high RF signal propagation speed [30, 32, 37]. Some recent work [19, 40] has leveraged acoustics for high-precision device tracking by estimating the distances to multiple anchor points. They still require additional infrastructure (e.g., multiple external speakers), which increases the cost and adds configuration efforts. Even though these speakers may be available in users' environment (e.g., speakers on a laptop, TV), they are usually not separated in ideal locations to serve as anchor points. Users still need to play the designed sound on a laptop, or hack the TV audio system.

In this paper, we propose *BatTracker*, the first high precision, infrastructure-free mobile device tracking system in 3D space with a range comparable to existing commercial solutions [1, 3, 5]. The device continuously emits acoustic signals that bounce off surfaces of nearby objects (e.g., walls, ceilings). The echoes are received and relative distances to those reference objects are inferred. The distance estimations are used to correct drift errors in position prediction from inertial data. Unlike inertial-only approaches [15, 22], BatTracker leverages acoustic signals capable of

accurate distance measurements. It is not affected by lighting conditions and does not cause privacy concerns. Compared to recent acoustic approaches [19, 40], it uses echoes from nearby objects instead of dedicated external anchor points (e.g. speakers), thus eliminating the needs of any additional infrastructure.

Despite such benefits, accurate, robust and infrastructure-free device tracking based on echoes is far from straightforward. Due to the existence of multiple surrounding objects, many echoes, not just those bouncing off objects of large surfaces, but also those from smaller objects or over multiple surfaces (i.e, multi-path), will be received. Reliably associating echoes to objects is critical to obtain the correct distance measurements. The device movement may create occlusions (by the device, other objects or human body) of the path, thus frequent missing of echoes from reference objects; it can also produce significant noises thus inaccurate acoustic measurements. We must design robust algorithms to reliably associate distance measurements to reference objects despite noisy data and frequent missing of desired echoes.

We make the following contributions in this work:

- We achieve high-precision device tracking in 3D indoor environments by incorporating complementary inertial and acoustics measurements, while eliminating the requirement of any additional hardware.
- We design an acoustic sensing technique that can produce accurate distance measurements from the device to nearby objects using echoes, and infer the device's movement velocity from Doppler shifts.
- We follow a multi-hypothesis tracking framework, and design a probabilistic tracking algorithm that fuses distance measurements, Doppler shifts and echo amplitudes to create, prune and evolve multiple hypotheses about the device position over time, thus achieving robust and accurate tracking.
- We build a prototype and conduct extensive experiments, demonstrating that BatTracker can track mobile devices in a typical room with a maximum error of $< 1cm$ for 2D tracking, and a 90-percentile error of $\sim 1cm$ for 3D.

To the best of our knowledge, BatTracker is the first work to track mobile devices in indoor environments at sub-*cm* accuracy without the requirement of any additional infrastructure.

2 BACKGROUND

The most straightforward way for tracking is based on inertial sensors. Accelerometer and gyroscope embedded in mobile devices allow us to calculate both the direction and speed of the movement. Theoretically, we can integrate acceleration to get velocity, and integrate velocity to get the moving distance. However, this approach suffers from large drifts over time: the errors can accumulate to meter level in a few seconds [15, 40].

Acoustics is a favorable sensing modality for ranging and tracking due to the slow sound propagation speed, hence higher accuracy. Recent acoustic based device tracking work estimates distances from the device to multiple anchor points, either by integrating the device moving velocity, or directly from Frequency Modulated Continuous Waveform (FMCW) [31] or propagation delay, then triangulate the device location.

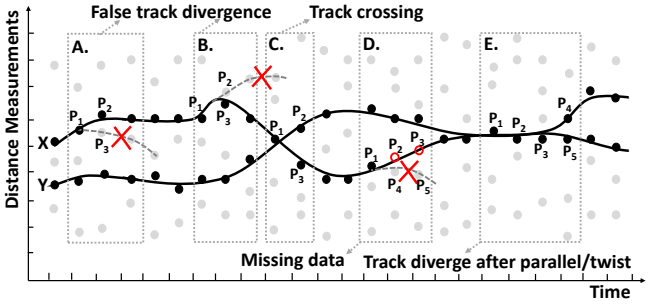


Figure 1: Solid lines represent tracks to two reference objects. Dots at each time stamp are distance measurements from acoustics, where grey ones are from clutters or noise and black ones are from reference objects. Red circles represent missing measurements.

Existing work [40] leverages velocity estimated from Doppler shift for tracking, which turns out to be more reliable than inertial tracking since it requires single integration. However, the error accumulation is still non-negligible over slightly longer time period(e.g., tens of seconds). Direct distance measurement does not require integration hence eliminating error accumulation. However, existing work (e.g., CAT [19]) requires infrastructure (e.g., multiple external speakers) as anchor points, which adds to the costs and installation/configuration efforts.

In consideration of the above, we decide to leverage echoes from nearby objects with large surfaces (e.g., walls, ceiling/floor, large furniture) commonly existing indoors. BatTracker predicts the device position using inertial data in a short recent time window, and immediately correct accumulated error using acoustic measurements.

3 CHALLENGES

Accurate tracking leveraging echoes faces multiple challenges: i) multi-path effects and clutters in a room make acoustic echoes thus measurements inevitably noisy; measurements to desired reference objects may be frequently lost due to phone movements or occlusion; ii) the echo-object association, i.e., which relative distance/velocity measurements correspond to which objects, must be reliably identified; iii) multiple noisy and error-prone inputs (including inertial, distance and velocity measurements) must be fused effectively and efficiently for robust, accurate tracking.

Figure 1 illustrates four problems in echo based tracking. To simplify, assume two reference objects X, Y (e.g., two walls joining at a room corner) exist and at each time point 5 echoes (some from other objects and/or multipath) thus distance measurements are obtained. As the device moves, we must reliably tell which distance measurements are the relative distances to X, Y over time (i.e., the two curves).

False track divergence. Since device movements are continuous, the distance at next time slot should be close to the previous one. Thus a straightforward way is to use the closest distance measurement in the next time slot. However, due to other objects and multi-path in cluttered environments, many other echoes thus distance measurements exist. This method can easily diverge onto a false trace if there exists a distance measurement coming from

clutter or noise but closer to the previous measurement. Case A shows a distance represented by point P_3 is closer to P_1 than the correct point P_2, causing a false divergence.

One may assume moving velocity is constant in short time periods thus traces are smooth. However, sharp acceleration or turning can still happen. Thus the past "trend" may also cause false track divergence. Case B shows that P_2 is selected based on the trend, but P_3 is the corrent one due to a sharp turn in movements.

Track crossing. For 3D tracking, we need to maintain individual distance tracks to at least three reference objects. Sometimes two (or even more) distances become close to and cross each other. We must figure out which one corresponds to which when they pass the crossing point (e.g., in case C, which of P_2, P_3 corresponds to X, Y after crossing P_1).

Missing data. Ideally, the speaker and microphone should be (at least partially) facing the reference objects all the time for robust measurements. However, facing away or opposite from reference objects, or occlusion to them by other objects or the human body, can all happen and cause echoes from desired reference objects missing. Case D shows measurements P_2 and P_3 are missed, hence P_4 and P_5 (from clutter, multi-path) are incorrectly taken to update the trace.

Track diverge after parallel/twist. This case is more difficult than crossing because two traces become almost merged into one for extended time. Using distance or velocity cannot tell which is which when they diverge. Case E shows two tracks merge through P_1, P_2, P_3, then diverge into separate tracks. Neither closest distance neighbor or smooth trend can help decide P_4 and P_5 to the correct track.

4 BATTRACKER DESIGN

BatTracker incorporates two sensing modalities for accurate, robust, and infrastructure-free 3D tracking: inertial sensors in a *Motion Model* for track prediction and acoustics in an *Observation Model* for track correction (Figure 2). Before tracking starts, a trace is initialized by locating the device in the room coordinate system using distances to reference objects (e.g., walls and ceiling in a room, or furniture such as large dressers). Based on the motion model, we predict the position of the device in the next time slot by double integration of acceleration. The short interval ensures relatively small error accumulation. Multiple hypotheses on the association between distance measurements and reference objects, thus the device location, are derived by incorporating Doppler shifts and echo amplitudes. The observation model evaluates the strengths of evidences for each of these device location hypotheses, and makes track splitting/pruning decisions to correct errors. The above prediction, association and correction steps are repeated for continuous tracking.

4.1 Acoustic Sensing

The acoustic sensing module of BatTracker consists of signal emitting, recording, and a series of signal processing steps to produce distances, amplitudes and Doppler shifts, hence velocity measurements for echo candidates from nearby objects in indoor environment (Figure 3). Unlike some existing work [10, 17] that only produces ranging measurements, we develop signal processing

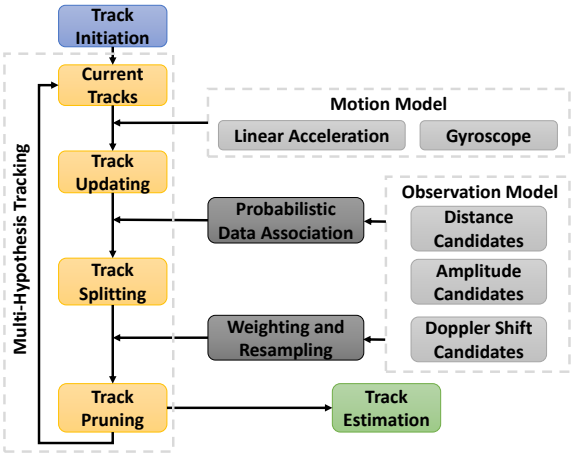


Figure 2: BatTracker uses inertial data in a motion model for track prediction, acoustics in an observation model for track correction, and fuse them into a multi-hypothesis tracking framework for robust tracking.

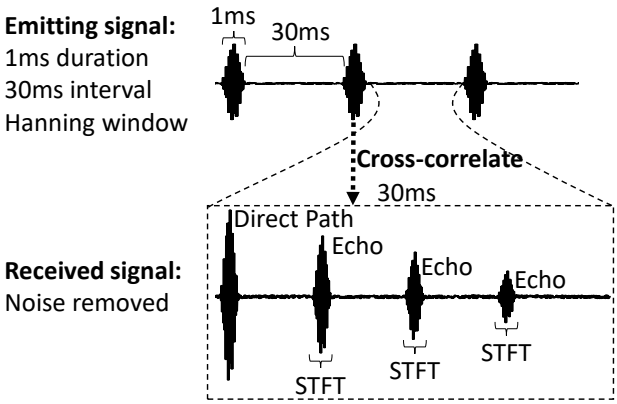


Figure 3: A particular sound signal and multiple signal processing steps produce echo distance, amplitude and velocity measurements.

techniques for accurate and reliable range measurement, and additionally Doppler frequency shift, thus both relative distance and velocity to each object.

4.1.1 Emitting Signal Design. Existing acoustic ranging work uses frequency modulated pulse signals that have a linear increasing frequency to improve the ranging resolution [10]. However, deriving frequency shifts from such frequency modulated pulses is difficult. Thus we choose signals with a constant frequency, chosen at $17KHz$, which is slightly audible to human. An even higher frequency decreases the robust tracking ranges because of faster signal power attenuation. By setting a moderate volume, our designed sound signal is nearly inaudible to most users, especially under background noise (e.g., music, video games).

We choose a pulse length of $1ms$. Such a short pulse length reduces potential overlapping between echoes traveling similar distances, thus improving the distance measurement resolution. It also provides less sampling points for each echo as inputs for

Doppler shifts extraction, thus decreasing the accuracy of velocity estimation. In BatTracker, we have a high accuracy requirement on distance measurements, while velocity from Doppler shift serves as complementary inputs, hence we prefer a shorter pulse duration.

A Hanning window [13] is applied on the pulse to reshape its envelop to increase its peak to side lobe ratio, thus producing higher signal to noise ratio (SNR) for echoes. To ensure echoes from two consecutive pulses do not overlap, there has to be enough gap in between. From experiments, objects more than $5m$ away create very weak echoes, which can be ignored. Thus the minimum delay between two pulses is $\frac{5m\times2}{343m/s} \approx 29.15ms$, where $343m/s$ is the sound propagation speed under a typical room temperature of 25°C. We give a bit buffer space and set it at $30ms$. This would allow $1000/(30+1) \approx 32Hz$ measurement rate, sufficient for tracking the device movements.

4.1.2 Acoustic Measurements Generation. We develop several steps to generate distance and velocity candidates from received signals.

Noise Removal. The received signal will go through a Butterworth bandpass filter with pass band of $17K \pm 200Hz$ to remove background noise, while preserving the frequency shift caused by the Doppler effect. Without such filtering, weak reflections can be buried in the noise. This step is critical for tracking in noisy environments.

Locate Each Pulse. Next we cross-correlate the signal with the designed pulse, a common technique [23] that produces a peak for each echo, and obtain the upper envelop for the signal. We chop the envelop into segments of small time windows of $31ms$, each containing echoes from one pulse only. To this end, we need to find the starting points of these windows. Since the first peak will always be the direct sound from the speaker to the microphone and has the highest amplitude, they are used as the starting points.

Distance Estimation. For each emitted pulse, multiple peaks corresponding to different echoes from different objects are detected. Using a threshold, we can select only the top-K strongest peaks, which are hopefully from larger, closer objects. By calculating the time delay between each echo and the starting point, we can estimate the distances between the device and surrounding objects. We also extract the amplitude for each echo, which is used for data association in tracking algorithms.

Doppler Shift Estimation. After locating each echo, we analyze the frequency f_e of each received echo, which consists of 48 sampling points ($1ms$). As the sampling frequency $f_s = 48KHz$, taking fourier transform on 48 points will give us a frequency resolution of $\frac{48000}{48} = 1KHz$. We use a similar approach of artificial padding of zero-valued points described in [40], and apply Short Term Fourier Transform (STFT) to get $1Hz$ resolution. The relative velocity to each object where the echo comes from can be calculated by the following equation:

$$v = \frac{f_d}{2 \cdot f} \cdot c \tag{1}$$

where $f_d = f_e - f$ is the Doppler frequency shift, $f = 17KHz$ is the frequency of the designed signal, c is the sound propagation speed. The sign of f_d indicates the direction of movement, where

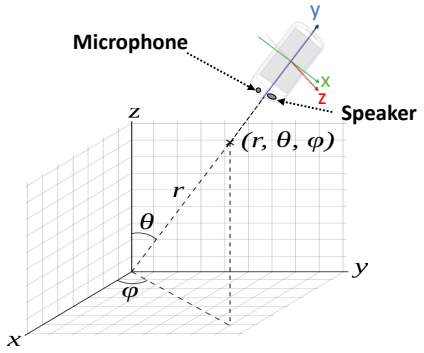


Figure 4: Microphone and speaker are at the bottom of the mobile device, which is pointed to a corner in a room for reliable echo detection.

$f_d > 0$ means movements towards the object, $f_d < 0$ those away from the object. This gives us a velocity resolution of $\sim 1cm/s$.

Note that due to the very limited amounts of sampling points, the velocity estimation from Doppler shifts may not be very accurate. Hence we do not use it directly for device moving velocity estimation, but incorporate it as a complementary input for data association improvement.

5 TRACKING ALGORITHM

We design algorithm for track initiation through simple movement gestures, and a multi-hypothesis particle filtering framework for track updating. Inertial data serves as motion model for state prediction, acoustic distance measurements are observations for state correction. Echo amplitudes and velocities from Doppler shifts are leveraged for data association problem and importance weight estimation.

5.1 Track Initiation

Track initiation is the process of finding major reference objects with stable echo reflections, and determining the distances to each of them, such that we can track such distances continuously. In practise, two adjacent side walls and the ceiling can be a very good reference object combination as they are relatively large and clean. Figure 4 shows the reference coordinate and the device local coordinate, the x-y plane is the ceiling, y-z and x-z planes are two perpendicular side walls. There are two major challenges for track initiation: i) how do we find out objects that create strong and stable echoes, hence with less chance of data missing, and ii) how to select three stable objects, and associate each of them to the x, y, z directions in reference coordinate.

We solve the above problem by simple movements using a statistic approach. Without loss of generality, we illustrate the process of finding reference object (e.g., x-z plane) in Figure 5. We perform the following two steps, track candidate generation and track association for initiation.

i) Track candidate generation. We point the phone's bottom (where speaker and microphone locate) to one direction (e.g., perpendicular to x-z plane), hold the phone still for a few seconds ($0-2.7s$ in Figure 5). In this step, we try to find out objects that create stable measurements. The horizontal line between $0-2.7s$

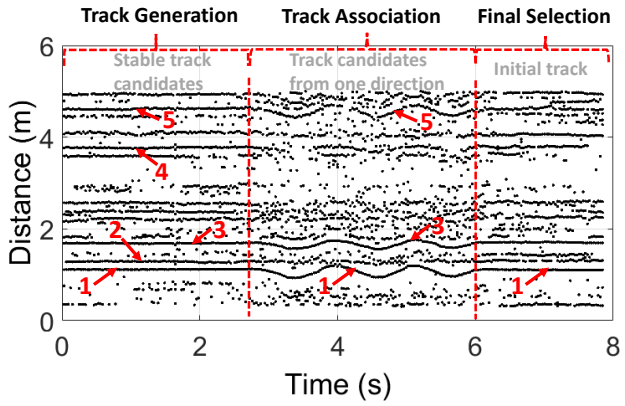


Figure 5: Track initiation process in one direction, which consists of track candidate generation, track association and final track selection.

corresponds to nearby objects from multiple directions. We use a simple density based clustering approach to locate these candidates, and keep the top-K candidates with least measurement variance as stable potential track candidates. In this example, we can easily notice that there are 5 stable candidates as marked in Figure 5. Note that these stable echoes may come from any direction, hence we still need to figure out which one is from x-z plane along y axis.

ii) Track association. We make minute movement back and forward along y axis several times ($2.7 - 6s$ in Figure 5). During this moving period, echoes coming along y axis shows a strong robustness to noise (e.g., track candidate 1, 3, 5). Since the movement is minute in a small range, we can easily track the distance measurements of such track candidates using a naive nearest neighbor method with a range constraint. We derive the moving acceleration by double deviation of the distance measurements, and cross-correlate this derived acceleration with the one measured from inertial sensor to get the similarity. Track candidates from x-z plane have the highest similarity since they have the highest match to the movement along y axis. In such way, track candidate 2, 4 are filtered out. Now we know track 1, 3, 5 are coming along y axis, and we select the one with highest cross-correlation similarity (e.g, track 1) as our final initial track.

We do the above steps for each direction to find reference objects. Typically, it takes a few seconds for moving the device along one direction, hence the whole track initiation can normally be finished in $< 30s$.

5.2 Track Updating

We update the track over time based on the sensing data from inertial sensors and acoustics, where inertial data serves as motion data for device position prediction, and acoustic measurements serve as observations for position correction. The most challenging problem is caused by the cluttered nature of indoor environments, which creates lots of undesired echoes, hence making it hard to associate distance measurements to our reference objects. We start from introducing the motion model and observation model in our design, then present how the challenges are solved using a probabilistic Multi-Hypothesis Tracking (MHT) framework.

MHT has been used in computer vision and radar applications for human or aircrafts tracking [24, 29]. MHT allows a track to be updated by more than one measurement at each update, spawning multiple possible tracks, and it calculates the probability of each track and typically only reports the most probable of all the tracks. Extended Kalman Filter (EKF) and particle filters are commonly used in MHT problems [12, 14]. A more detailed description of MHT can be found in [7]. Note that MHT is only a general framework, the critical task is the detailed design of track initialization and state update algorithms, and how to adapt MHT to our specific inertial/acoustic tracking problem.

5.2.1 Models. Following the MHT framework, we design the motion model and observation model for our tracking system.

Motion Model. Device position and velocity are defined as state vector $s(t) = [x(t), y(t), z(t), \dot{x}(t), \dot{y}(t), \dot{z}(t)]'$, where $[x(t), y(t), z(t)]$ represents the device position in reference coordinate (i.e, room), $[\dot{x}(t), \dot{y}(t), \dot{z}(t)]$ is the corresponding velocity along each axis.

The absolute acceleration without gravity along each axis in the device's local coordinate is obtained from composite sensor linear acceleration from Android API, while the absolute device orientation is obtained from composite sensor game rotation vector. We transform the linear acceleration in device coordinate into reference coordinate based on device orientation, and leverage the transformed acceleration for state prediction. The state update equation can be derived from a motion model on the state vector. Based on the acceleration, we model the target motion with velocity and acceleration along three axes. The resulting state update equation is as follows:

$$\hat{s}(t + \triangle t) = A(t)s(t) + B(t)\mu(t) + \varepsilon(t) \tag{2}$$

where $\hat{s}(t + \triangle t)$ is the predicted state at time $t + \triangle t$, $\triangle t$ is the inertial data sampling interval, $A(t)$ and $B(t)$ are motion matrix from physical model $x(t + \triangle t) = x(t) + \dot{x}(t)\triangle t + \frac{1}{2}a\triangle t^2 + \varepsilon(t)$, where a is the linear acceleration and $\varepsilon(t) \sim \mathcal{N}(0, \Sigma_u)$ is the motion noise to capture the uncertainty. Since the acoustic sampling rate is lower than inertial, we keep updating the state during time interval $[t, t + \tau]$, where τ is the time interval between adjacent acoustic measurements. This inertial based motion prediction happens every acoustic sampling cycle (e.g., $\sim 30ms$ in our implementation), then the predicted location is corrected by acoustic measurements, thus the accumulation error over this short period is sufficiently small to be negligible.

Observation Model. The observation consists of range estimates to nearby objects at time t, which is represented as $d(t) = \{d_1(t), d_2(t), ..., d_N(t)\}$. This observation model can be visualized as shown in Figure 5. To minimize the possibility of missing data, we generate N distance candidates at each time slot, which are represented as black dots in Figure 5 with a time interval τ. Note that a larger N can decrease the data missing probability, however it brings more clutter/noise measurements. As a balance between two cases, we set $N = 15$ in our experiments. The state $s(t)$ consists of phone location and velocity at time t, hence we still need velocity observations. There are actually two choices in our design: velocity derived from Doppler shift, and inferred by device position changes. Due to the limited sampling points for Doppler shift extraction, the velocity estimation is not robust enough as direct observations

(evaluated in Section 6.1). Hence we select the later solution, velocity can be easily estimated by dividing position changes by time interval. We estimate the velocity in a time window of multiple τs to reduce the error. In our implementation, $\tau = 31ms$ is a very small interval, a small distance error can lead to a large error in velocity estimation as τ is the denominator.

It is assumed that distance measurements at each time t are normally distributed around the true ranges to objects with variance σ_r^2 and a data missing probability P_M. These measurements may also include spurious peaks due to clutters or noises. Consider the observation $d(t) = \{d_1(t), d_2(t), ..., d_N(t)\}$, the range candidates $d_i(t)$ may correspond to one or multiple reference objects, or the clutters. Hence, we define a set $\hat{R}(t) = \{z_m(t)\}_{m=1}^{N^3}$ that consists of the all possible unordered combinations $z_m(t) = \{d_i(t), d_j(t), d_k(t)\}$, within which $d_i(t), d_j(t)$ and $d_k(t)$ are distances to the three reference objects. In our case of 3 perpendicular surfaces in a corner, the 3 distances will uniquely determine the location of the device. Since the distance to each reference object can be the same, hence we can have N^3 possible combinations, which is a non-trivial amount. We reduce the number of combinations by measurement validation, which removes measurements that are "far away" from predicted location:

$$R(t) = \{z_m(t) \subseteq \hat{R}(t) : [z_m(t) - \hat{z}(t)]' S^{-1} [z_m(t) - \hat{z}(t)] \leq \gamma\} \quad (3)$$

where $R(t)$ is the set of validated measurement at time t, $\hat{z}(t)$ is the predicted position from predicted state $\hat{s}(t)$, γ is a threshold to limit the number of validated measurements, $S = diag\{\sigma_r^2, \sigma_r^2, \sigma_r^2\}$ is the covariance matrix of the measurements, which describes the uncertainty.

5.2.2 Probabilistic Multi-Hypothesis Tracking. The tracking problem is formulated as a Sequential Monte Carlo (SMC) problem, specifically, a variation of particle filtering framework with multi-hypothesis tracking capability. We maintain a collection of $K(t)$ "particle clouds", and each cloud represents a possible state estimate of a possible track, i.e, a track hypothesis. The kth particle cloud $\{s_k^n(t)\}_{n=1}^{N_k(t)}$ consists of a collection of $N_k(t)$ "particles". Each particle has the concrete values for each dimension of a state $s(t)$, representing a possible device position and velocity at time t. The total number of particles of all the particle clouds sum up to a constant number $N = \sum_{k=1}^{K(t)} N_k(t)$. The framework operates on discrete time with an interval τ and repeats multiple steps for each time slot: track update, track splitting, track pruning and track estimation.

We introduce the high-level design of these steps, and elaborate each step afterwards. *Track update* predicts the state $s(t)$ of each particle in each existing particle clouds according to the motion model. *Track splitting* splits each particle cloud into "particle sub-clouds" when multiple validated measurements are available, and computes the weight for each sub-cloud. *Track pruning* removes sub-clouds with a weight less than a certain threshold, normalizes the weights of remaining sub-clouds, then repopulating the number of particles within each sub-cloud to form a new set of particle clouds with a total number of particles of N. *Track estimation* returns the weighted mean of all the particles as the estimate of current state. Figure 6 shows the above steps, illustrating how particle clouds evolve, and life cycles. In such a way, we can maintain multiple

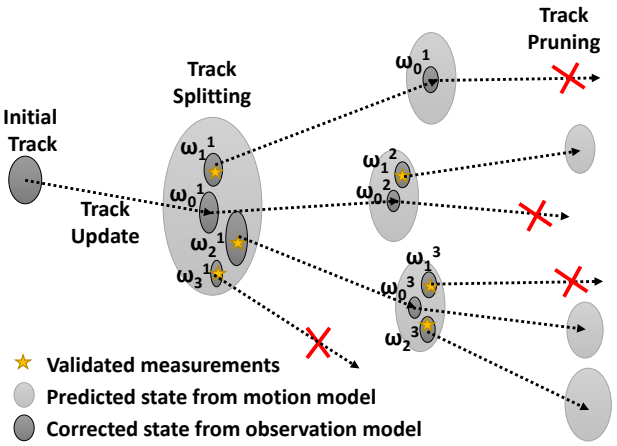


Figure 6: Tracks are represented by particle clouds, which are updated by motion model and split into multiple sub-clouds according to validated measurements. Those clouds with a weight lower than a threshold are pruned.

potential "correct" track hypotheses by multiple particle clouds, and leverage future measurements to kill the "incorrect" ones.

Track Update. The state $\vec{s}(t) = \{s_1(t), s_2(t), ..., s_{K(t)}(t)\}$ consists of all the particle clouds $s_k(t)$ indexed by k, where $K(t)$ is the total number of particle clouds at time t. The kth particle cloud at time t is defined as:

$$s_k(t) = \{[x_k^n(t), y_k^n(t), z_k^n(t), \dot{x}_k^n(t), \dot{y}_k^n(t), \dot{z}_k^n(t)]\}_{n=1}^{N_k(t)} \quad (4)$$

where $N_k(t)$ is the total number of particles in this cloud. For each iteration, the state represented by each particle is updated according to the state update equation (2) in motion model.

Track Splitting. Given the state $\vec{s}(t) = \{s_1(t), s_2(t), ..., s_{K(t)}(t)\}$ at time t, each particle cloud $s_k(t)$ is updated according to the validated measurements from observation model $R_k(t+\tau) = \{\{d_x^{m,k}(t+\tau), d_y^{m,k}(t+\tau), d_z^{m,k}(t+\tau)\}\}_{m=1}^{M_k(t+\tau)}$ at time $t+T$, where $M_k(t+\tau)$ is the number of validated measurements for the kth cloud. $M_k(t+\tau) = 0$ means no validated measurement is available, i.e, data missing. We leverage all the $M_k(t+\tau)$ validated measurements, and update a fraction of particles according to each measurement, hence splitting the particle cloud. A fraction of particles without update is always preserved to capture the case of data missing. When multiple validated measurements are available, we compute the likelihood for each of them that it is the "correct" measurement from our reference object using a probabilistic data association approach.

Probabilistic data association. For each validated measurements in $R_k(t+\tau)$, we may have two situations: the "correct" measurement is captured, or missed. We denote the data missing probability as $P_M(t)$, hence the weight for each measurement to be "correct" can be defined by a normalized weights:

$$\omega_m^k(t+\tau) = \begin{cases} \frac{\mathcal{L}_m^{*k}(t+\tau)}{P_M(t+\tau) + \sum_{j=1}^{M_k(t+\tau)} \mathcal{L}_j^{*k}(t+\tau)}, m = 1, 2, ..., M_k(t+\tau), \\ \frac{P_M(t+\tau)}{P_M(t+\tau) + \sum_{j=1}^{M_k(t+\tau)} \mathcal{L}_j^{*k}(t+\tau)}, m = 0, \end{cases} \quad (5)$$

where $P_M(t+\tau)$ is the data missing probability at time $t+\tau$, which means the "correct" measurement is missed, and

$$\begin{aligned}\mathcal{L}_m^{*k}(t+\tau) &= p(z_m^k(t+\tau)|\hat{z}^k(t+\tau)) \\ &= \mathcal{N}[z_m^k(t); \hat{z}^k(t+\tau), S] \\ &= \frac{1}{\sqrt{2\pi S}} exp\{-\frac{1}{2}[z_m^k(t+\tau) - \hat{z}^k(t+\tau)]' S^{-1} \\ &[z_m^k(t+\tau) - \hat{z}^k(t+\tau)]\}\end{aligned} \tag{6}$$

is the likelihood of the measurement $z_m^k(t+\tau)$ originating from the desired reference object rather than from clutter, where $\hat{z}^k(t+\tau)$ is the predicted device position at time $t+\tau$ from motion model, S is the covariance matrix of measurements.

Data association enhancement. Now that we get an estimation of how likely the distance measurement combination $z_m^k(t+\tau)$ is the "correct" one based on the similarity between predicted position and distance measurement. However, due to the noisy inertial data, the predicted position is not accurate enough for robust data association. Besides, two close distance measurements can be easily confused. Hence we further optimize the data association by incorporating additional information from data: the velocity from Doppler shift and echo amplitude.

We define velocity $v_m^k(t+\tau)$ as the velocity measured from Doppler shift at time $t+\tau$, and $\alpha_m^k(t+\tau)$ is the vector of echo amplitudes. The device movement is always continuous, hence amplitudes are supposed to be continuous in very short time interval. Since they are independent observations, we incorporate the likelihood probability $p(v_m^k(t+\tau)|\hat{v}^k(t+\tau))$ and $p(\alpha_m^k(t+\tau)|\alpha^k(t))$ to enhance the data association, where $\hat{v}^k(t+\tau)$ is the predicted velocity from motion model, $\alpha^k(t)$ is the vector of amplitudes at time t. Thus the integrated data likelihood can be formulated in a product form as:

$$\mathcal{L}_m^k(t+\tau) = \mathcal{L}_m^{*k}(t+\tau) \cdot p(v_m^k(t+\tau)|\hat{v}^k(t+\tau)) \cdot p(\alpha_m^k(t+\tau)|\alpha^k(t)) \tag{7}$$

We further estimate the data missing probability P_M at each time slot according to the pose (location and orientation) of the device. For simplicity, the data missing probability P_M can be set as a constant value, hence we always split a fixed portion of particles, and evolve this sub-cloud without measurement data. Due to the physical layout of mobile phone and hardware constrains, we find that the data missing probability is highly dependent on the relative orientation of the phone to reference object, whereas the distance to the object has less impact within a certain range. Figure 4 shows a typical way of how we hold the phone for best tracking performance: the phone points to a corner to get strong echoes from three reference surfaces. In practise, a ceiling corner in a room is a preferred choice. For clarity, Figure 4 shows the case when the phone points to a bottom corner. From experiments, we found that θ has a strong impact of data missing from x-y plane, while φ has a strong impact on both x-z and y-z plane. As data missing from each reference object is independent, this data missing probability $P_M(t)$ can be formulated as a product form as follows:

$$P_M(t) = P_{MX}(\theta, t) P_{MY}(\varphi, t) P_{MZ}(\varphi, t) \tag{8}$$

where $P_{MX}(\theta, t), P_{MY}(\varphi, t), P_{MZ}(\varphi, t)$ are data missing probabilities from each direction, which are approximated using polynomial functions from experiment data in Section 6. We omit the distance as it has negligible impact compared to orientation within our tracking range.

Particle Cloud Splitting. Each particle cloud may have multiple validated measurements, and their associated weights. We consider one validated measurement $z_m^k(t+\tau)$ at each time, hence our problem becomes a traditional particle filter problem, which requires weight calculation and resampling [43]. Based on this measurement, we compute the weight for each particle within the kth cloud using the same form in Equation 7, and resample a number of $N_m^k(t+\tau)$ particles to form a sub-cloud, where $N_m^k(t+\tau)$ is proportional to $\omega_m^k(t+\tau)$ and $\sum_{m=0}^{M_k(t+\tau)} N_m^k(t+\tau) = N_k(t)$.

Track Pruning. Each particle cloud is split into multiple sub-clouds at each time slot. Without careful pruning, the number of particle clouds will increase exponentially. Hence we need to terminate some clouds and repopulate the remaining ones with new particles. As shown in Figure 6, there's only one particle cloud when the track is initialized. Then it's split into 4 sub-clouds, each with a weight of ω_i, which is the data likelihood. We compare ω_i to a pruning threshold λ, and terminate clouds with weights lower than λ. In this example, the cloud with ω_3 is terminated. Then we normalize the weight of remaining particle clouds, and repopulate each cloud by drawing new particles randomly from existing cloud to maintain the total number of particles. As the tracking evolves, the repopulated particle clouds are further split into multiple sub-clouds, each with a weight proportional to the product of current cloud weight and the data likelihood. Then we repeat the pruning, weight normalization, and repopulation steps.

A particle cloud is split into multiple sub-clouds when multiple validated measurements are available. By setting a proper threshold, we keep the number of validated measurements within 10, typically $3-5$. The size of a cloud can increase or decrease during splitting and pruning. A cloud with larger weights, might evolve with a larger number of particles after each pruning step. On the contrary, a cloud with low weights may evolve with a lower number of particles, and disappear eventually.

Track Estimation. The estimated state of tracking is the mean of particle distributions and can be obtained as a weighted average of all the particle clouds:

$$s_E(t) = \Sigma_{i=0}^{K(t)} \omega_i s_i(t) \tag{9}$$

this estimated track is further smoothed using a moving average filter to reduce jitters.

Our multi-hypothesis tracking algorithm repeatedly performs track update, track splitting, and track pruning over time, and generates track estimation for device tracking.

6 EVALUATION

We use Huawei P9, a representative mobile phone of mainstream design with speaker and one microphone located at the bottom, as our mobile device to evaluate BatTracker from two aspects: acoustic measurements and tracking performance. We conduct tracking experiments in a highly cluttered laboratory with area of

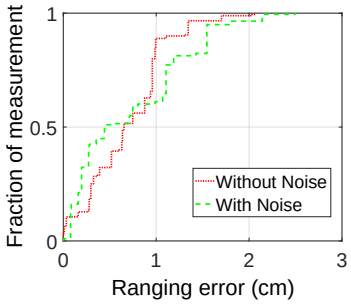
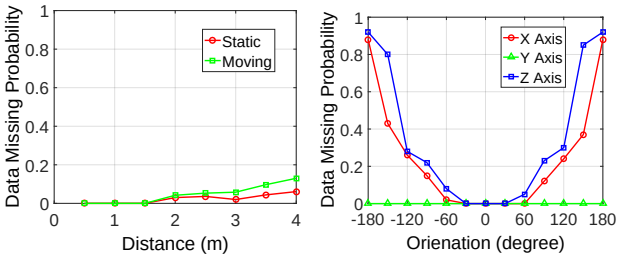


Figure 7: Distance candidate accuracy with and without background noise.



(a) Distance impact on data missing. (b) Orientation impact on data missing.

Figure 8: Data missing probability under different distances and relative orientations.

$\sim 15m^2$. Inertial data are sampled at $50Hz$, and acoustic signal is sampled at $48KHz$.

6.1 Acoustic Measurements

Acoustic measurements consist of distance measurements to nearby objects and relative moving velocities from Doppler shift. We evaluate acoustic measurements using the following metrics: ranging accuracy, data missing probability, and velocity from Doppler shift.

Ranging Accuracy. The ranging accuracy to reference objects is critical for accurate tracking. To evaluate the ranging performance, we select a plain wall in an empty space, and measure the distance to the wall at different locations using the mobile phone. We vary the location thus the ground truth distance changes from $0.5 - 3m$ with steps of $0.5m$, and we repeat 30 times at each location [1]. We also conduct the same measurement experiments in a noisy environment by playing a mixed sound (different kinds of music and talk shows) from a nearby laptop at a normal volume. Figure 7 shows the CDF for all the measurement errors of both with and without background noise. In quiet environment, the maximum error is within $2cm$ while the 90-percentile error is less than $1cm$. Errors under background noise have slightly larger maximum error of $\sim 2.5cm$, however, there's no significant performance deterioration. Both cases have a sub-cm level median error, which lays the foundation for high precision tracking.

Data Missing Probability. Data missing is one of the most challenging problems for robust tracking. Too much missing data in a short period can easily cause a tracking failure. From experiments, we find both the distance and relative orientation to an object have an impact on data missing probability, and phone movement can aggravate the problem. We evaluate the data missing probability in two individual experiments: impact of distance and movement, and impact of relative orientation.

Impact of distance and movement. A larger distance creates weaker echo reflections, which have lower signal to noise ratio (SNR), hence such echoes may not be detected and missed. To evaluate the impact of distance, we point the phone bottom to a plain wall, and vary the distance from $0.5 - 4m$ with steps of $0.5m$. We collect data for $10s$ at each location, which consists of ~ 300 measurements. To simulate the movement, we repeat the above steps while wobbling the phone within a small range while collecting data. For each measurement period, the measurement distances are supposed to be continuous within a certain threshold. If no measurement is present in this range, we count it as data missing. We set this threshold as $3cm$, and count the number of such outliers, which are regarded as missing data. Figure 8(a) shows the data missing probability of both static and wobbling situations. As we can observe, measurement are very reliable with almost no missing data within the range of $1.5m$. When the phone is static, the data missing probability remains at a low level ($< 5\%$) up to $3m$, and less than 10% up to $4m$, which turns out to be quite reliable. While the phone is wobbling, the probability gets larger, however, it's still within 6% up to $3m$.

Impact of relative orientation. The relative orientation of the mobile device to a reference object has a large impact on data missing problem for two reasons: i) it determines the facing direction of the speaker; ii) it determines the opening direction of the microphone. Since the microphone on mobile devices is not designed omnidirectional for application like BatTracker, the user needs to hold the device in a particular way (explained later) to minimize data missing probability.

We first evaluate the microphone sensitivity to sounds from different directions by analyzing the recording pulse amplitudes. One phone emits the designed signal continuously as sound source, simulating a mirrored speaker from a reference object. We use another phone to record the sound with a distance of $1.5m$ away from the sound source. Figure 9 shows the placement of two devices, and the rotation axes. The default orientation are defined as zero when the microphone faces the speaker of the source, which has the highest amplitudes. Then we rotate the receiving device along x, y, z axis sequentially with a step of 30° for recording. We use a bandpass filter to remove low frequency components, and analyze the amplitude of the filtered signal, which is shown in Figure 9. As we can observe, amplitude decreases slightly when the device rotates along x axis away from 0°, which is due to the slightly blocking from the phone frame with a blocking distance of the phone height. After that, it decreases a lot as the microphone faces the opposite direction. Rotation along y axis does not change the amplitude too much; the amplitude remains at a high level since the microphone always faces the speaker. Similar to x axis, amplitude also decreases when the device rotates along z axis away from 0°. However, it decreases much faster as the signal is blocked by

[1]Further experiments show that distance beyond $3m$ has a high probability of data missing, and causing tracking errors, thus not used.

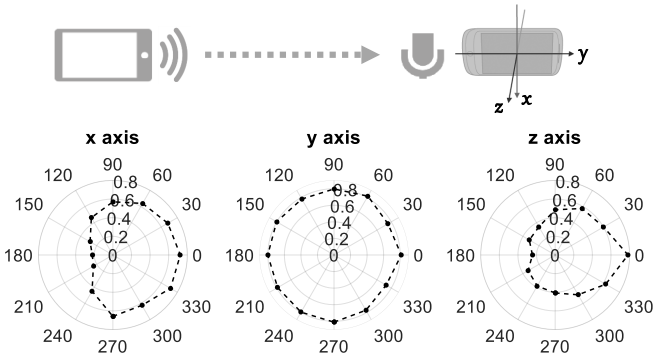


Figure 9: Normalized received signal amplitude under different rotations.

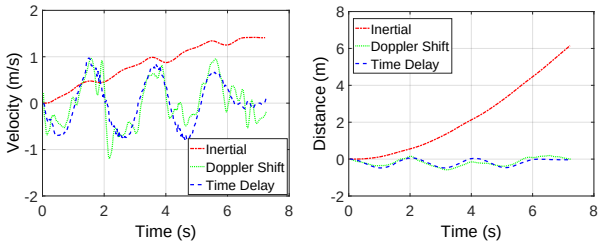
a distance of phone width. Figure 8(b) shows the data missing probability with various rotation. As expected, we almost have no missing data when the phone rotates along y axis. The data missing probability remains < 10% when rotating along x, y axis within the range $[-60°, 60°]$. Rotation along z axis incurs higher data missing probability than that along x axis. The evaluation results are used for parameter estimation in Equation 8 to approximate the data missing probability. In most cases, we select three perpendicular walls in a room corner as reference objects. To make a balance among the three walls, we recommend the user points the bottom of the device to a corner with the phone screen facing sideways. In this way, we can have a balance between two side walls, get a reliable refection from the ceiling, and suppress the unwanted clutter echoes from ground.

Velocity from Doppler Shift. We evaluate the accuracy of velocity derived from Doppler shift, and compare the distance estimations with inertial sensor and sound propagation time delay. We choose a clean wall as reference object to simplify the data association with a distance of $1.5m$ between the wall and device. Then we move the phone back and forth, and compare the velocity and ranging from different schemes. Figure 10 shows the velocity and ranging results. Velocity from Doppler shift and time delay (calculated from distance measurements) has high correspondence, however, they differs a lot at some time periods (e.g., 5-6s in Figure 10(a)). This is easy to explain, when echoes from different directions overlap, we can not extract accurate frequency shift from the mixed signal. Velocity from inertial sensor has a constant drift, which keeps increasing. Figure 10(b) shows integrated ranging from Doppler shift shows a high match to the time delay scheme, while the inertial based result has a large accumulated error, which goes up to $6m$ in a few seconds.

6.2 Tracking Performance

We evaluate the tracking performance from several aspects: tracking accuracy, various impact factors, comparison of different algorithms, and comparison with other work [19, 40].

Experiment Setup. We evaluate BatTracker in a highly cluttered laboratory with large tables and cabinets (Figure 11(a)). We select the ceiling corner as potential reference objects. To quantify the tracking error, we use a tilted box with drawn traces, and move



(a) Device moving velocity. (b) Distance to the wall.

Figure 10: Comparison of velocity and ranging from different schemes: inertial sensor, Doppler shift, and sound propagation time delay.

the mobile device along the traces. In such a way, we are able to get accurate ground truth for comparison.

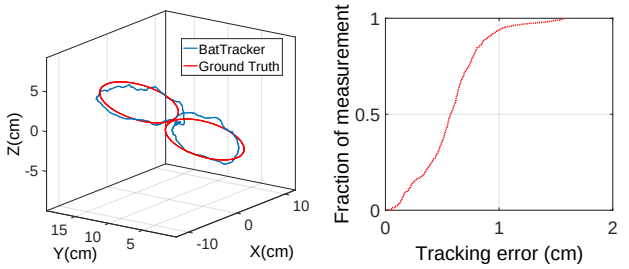
(a) Overview of experiment environment. (b) Tilted box with drawn traces.

Figure 11: We evaluate BatTracker in a highly cluttered laboratory, and a tilted box with drawn traces used as ground truth.

Tracking Accuracy. First, we evaluate the 3D tracking accuracy of BatTracker (2D tracking accuracy is evaluated later in the comparison with other work). We move the phone along a double circle "8" shape trace with a diameter of $10cm$ drawn on a tilted box, as shown in Figure 11(b). Figure 12(a) shows the generated traces and the ground truth. To quantify the error, we calculate the nearest distance for each point in the generated trace to the points in ground truth. Although this method does not perfectly capture the real tracking error, it can provide a reasonable benchmark for error quantifying. Figure 12(b) shows the CDF of 3D tracking error, which is less than $1cm$ at 90-percentile, and the maximum error is $\sim 1.5cm$.

Impact of the Number of Particles. We evaluate the impact of number of particles on tracking accuracy. Figure 13 shows that the 90-percentile error keeps decreasing from $\sim 5cm$ to $\sim 1cm$ when the number of particles increases from 500 to 1500, then it stays relatively stable. We also evaluate the number less than 500. However, in such cases, too few particles can easily lead to tracking failure, hence only present results with ≥ 500 particles.

Impact of Background Noise. To evaluate the robustness of our scheme to background noise, we repeat the 3D tracking accuracy evaluation experiment under mixed background noise. We



(a) Tracking result and ground truth. (b) Tracking error.

Figure 12: 3D tracking result and CDF.

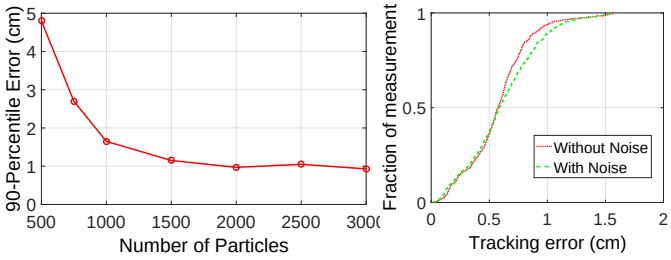


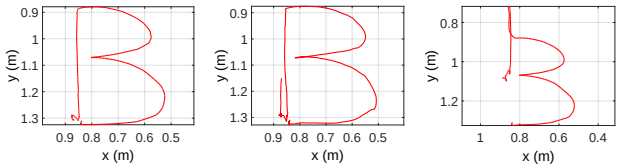
Figure 13: Tracking error decreases as the number of particles increases, then becomes stable.

Figure 14: Impact of background noise.

generate background noise using the same way as we evaluate ranging accuracy. Figure 14 compares the tracking error with and without background noise. Similar to the ranging evaluation under background noise, we observe no significant difference between two cases, which indicates that BatTracker is very robust to background noise. Our designed signal is at $17KHz$, which is far from common sound noises (usually from $1-8KHz$), hence we can easily filter out low frequency noises.

Impact of Different Environments. Besides the laboratory environment which has concrete walls, we also evaluate BatTracker in two other typical environments, one bedroom with wooden walls and a conference room with glass walls. No obvious performance difference is observed, thus we omit those figures.

Resistance to Track Initiation Error. To initialize a track, the user needs to move the phone in three directions back and forth sequentially to get the distances to three reference objects. These minute movements may introduce initiation error since the device can not be guaranteed to return back to the same location. To evaluate the impact of initiation error on the tracking accuracy, we manually add initiation error on the initial track. Figure 15(a) shows a normal tracking result by initiation without additional artificial error. Figure 15(b) shows the result by manually adding an error of $15cm$ in y axis. We observe that the initiation error is corrected near the beginning, and the overall tracking result is not obviously impacted. We further increase the initiation error up to $25cm$, and the result is shown in Figure 15(c). The resulting track has large error at the beginning, however, the track is automatically recovered in a few time slots. This is because our algorithm maintains multiple hypotheses and those evolve along the accurate measurements are assigned higher weights, hence preserved; while those using the inaccurate initiation measurements are pruned. This demonstrates that our algorithm is robust to initiation error, however, extreme initiation error may result in initiation failure. From user experiments, users can easily move the device back to the original position within $10cm$ for successful track initiation.



(a) Accurate initiation. (b) Medium initiation error. (c) Large initiation error.

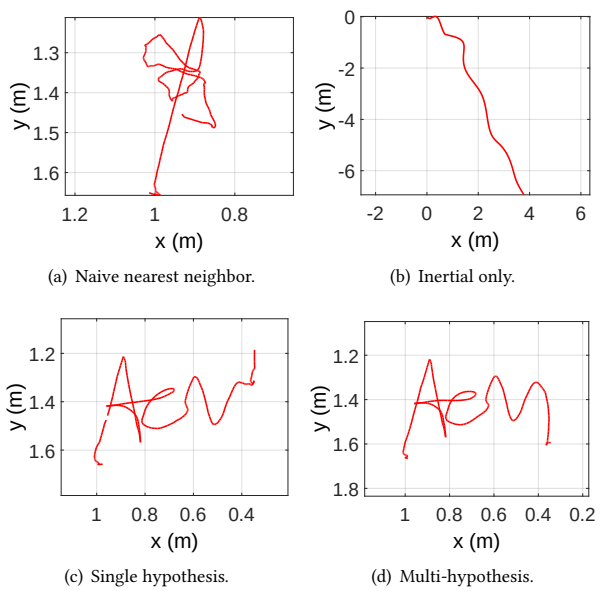
Figure 15: Tracking results with different level initiation error.

Comparison with Different Algorithms. We compare the performance of different algorithms, which includes *naive nearest neighbor range-only tracking*, *inertial only tracking*, *single hypothesis conventional particle filter tracking*, and our *multi-hypothesis tracking*. We use the same data set ("ACM" written in the air), and apply the above algorithms for comparison. Figure 16(a) shows the result from nearest neighbor algorithm. This algorithm finds the nearest measurement at next time stamp to maintain tracking based on the simple intuition that the track is continuous. However, the track is lost at the peak of letter "A". This demonstrates naive algorithm can fail easily due to challenges we summarize in Section 3, such as wrong data associate to a clutter noise. Figure 16(b) shows the result from inertial data only. We can notice a constant drift along y axis, which makes the track totally deviate from ground truth. Figure 16(c) shows the result of single hypothesis particle filter tracking. The result is much more robust compared to naive algorithm, however, the track is lost at the last turning point of letter "M" due to track deviation caused by wrong data association. Figure 16(d) shows the tracking result of BatTracker, which avoids the track deviation that happens in Figure 16(c) by maintaining multiple hypotheses, which contain the correct one and later it prunes incorrect ones due to the lack of supporting measurement evidences.

6.3 Drawing Evaluation

We evaluate the drawing capability of BatTracker, and compare the performance with CAT [19] and AAMouse [40]. In this comparison, BatTracker requires clean walls as reference objects, while CAT/AAMouse require external speakers. We also show more drawing examples to demonstrate the usability.

Comparison with CAT and AAMouse. We compare the performance of BatTracker to the most recent acoustic based tracking work CAT and AAMouse. CAT develops a FMCW based approach which can accurately measure the distance from mobile device to external speakers. Thus the device position can be triangulated from distances to multiple speakers. AAMouse shares the similar hardware architecture as CAT. It estimates the relative moving speed between mobile device and multiple speakers, hence tracks mobile device movements.



(a) Naive nearest neighbor. (b) Inertial only.

(c) Single hypothesis. (d) Multi-hypothesis.

Figure 16: Tracking results of different approaches. Naive nearest neighbor tracking is easily lost, while the inertial only tracking drifts rapidly. Single hypothesis particle tracking is more robust but still lost due to track deviation. BatTracker's multi-hypothesis tracking outperforms others.

To make the comparison straightforward, we adapt the same ground truth drawing shapes as CAT: a double-circle, a triangle, and a loop back. We draw such shapes on a paper box and move the device along the drawings as shown in Figure 11(b). To quantify the error, we calculate the nearest distance for each point in the generated trace to those in the ground truth. Figure 17 shows the traces produced by the three schemes, we can observe that both BatTracker and CAT show a high match to ground truth without obvious drift error, while AAMouse shows obvious deviation from the ground truth. To quantify the error, Figure 18 shows the tracking error CDF of all schemes. [2] In Figure 18(a), CAT has a maximum error of $\sim 2cm$, and a 90-percentile error of $\sim 1cm$, which is an obvious improvement compared to AAMouse. For the loop back, AAMouse shows a large drift error up to $6cm$. Figure 18(b) shows the error of BatTracker with a maximum error less than $1cm$, while the 90-percentile error is $\sim 0.5cm$, which turns out to be even higher than CAT. Although the ranging accuracies of BatTracker and CAT are comparable, the mechanism for tracking is different. CAT measures the distances from device to anchor speakers, and triangulate the location of the mobile device. Such triangulation can enlarge the tracking error, and its accuracy is impacted by the distance between anchor speakers and the initial position error. In contrast, BatTracker measures distances to perpendicular reference planes (e.g., walls) directly, thus avoids triangulation and eliminates such error in CAT. Besides, our scheme is robust to track initiation error, the overall tracking is not impacted by such errors. Additionally, we smooth the tracking result with a moving average filter, which further reduces errors for human drawing.

[2]Results of CAT and AAMouse in Figure 17 and Figure 18 are from the CAT paper [19].

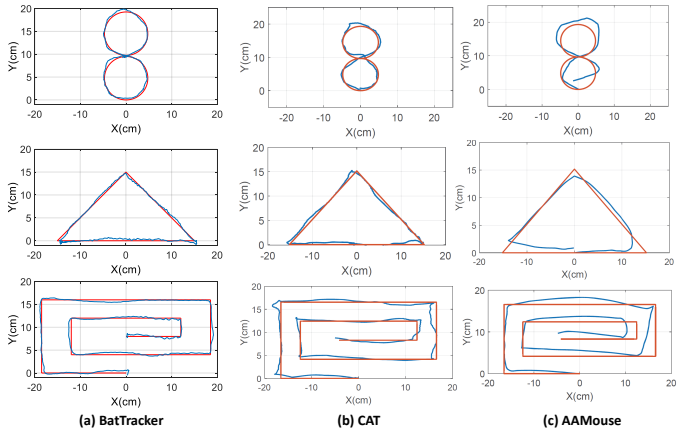


Figure 17: Shapes created by BatTracker, CAT, and AAMouse.

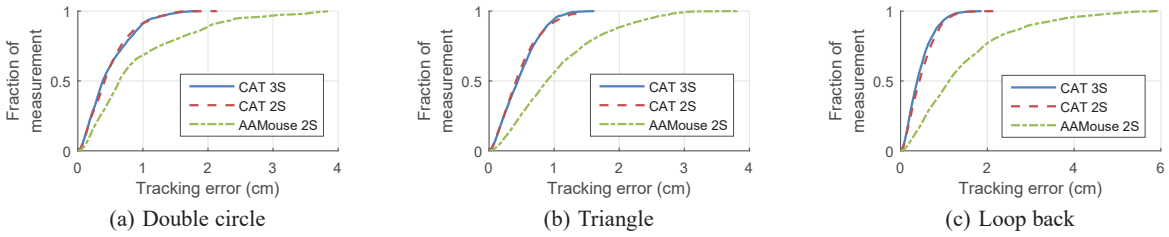
More Drawing Examples. We show more drawing examples produced by BatTracker. We write " I ♡ ACM Sensys" and draw a spiral freely in a typical bedroom. For better readability, "I ♡", "ACM" and "Sensys" are written separately. Figure 19 shows the drawing shapes, which are easy to be recognized. Note that there is a sharp jump at the beginning at the first letter "S" in the word "Sensys". This is caused by the initiation error, and corrected automatically in a very short period. "Sensys" is written with a span over $1.5m$ in space, which shows our algorithm is able to keep high accuracy over large distances. The spiral shows BatTracker is able to produce 3D tracking with free movements.

6.4 Computation Complexity

We implement the core tracking algorithms on Android devices, and develop a real-time tracking application. All the computation are done in real-time on the smart phone, while the tracking results are streamed to laptop through Wi-Fi for display only. On the smart phone screen, we also display all the raw distance measurements. We vary the number of particles from 600 to 1800, and monitor the allocated memory and CPU usage, which are listed in Table 1. From our experiments, the allocated memory for BatTracker is less than $15MB$, while the CPU usage is from $16.5\% - 19.65\%$ when the screen is on. Turning the screen off will disable the graph rendering, which cuts down the CPU usage by $\sim 6\%$. No obvious lag is observed with particles within this range, which is sufficient for our requirement. Further increasing particles to 3000 causes serious lag, while the CPU usage is still under 30%, which indicates the computation potential is not fully exploited. There's still a large room for optimization, such as a better multi-threading design. As our test device HuaWei P9 is a middle-end smart phone released more than one year ago, we believe most current and future smart phones have sufficient resources to run our algorithms.

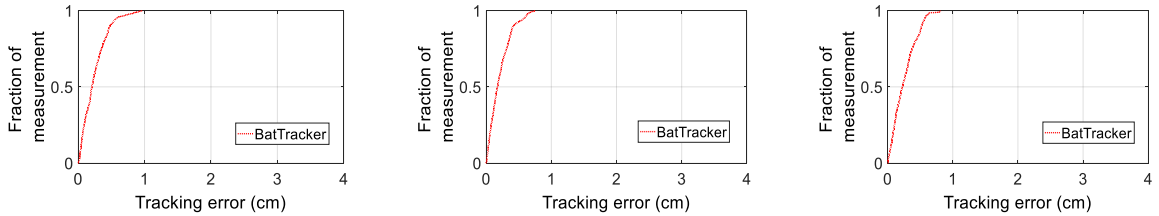
7 DISCUSSION

Potential Use. High precision, infrastructure-free device tracking in indoor environments has broad application scenarios. The latest video games and VR devices rely on additional anchor devices for high precision tracking, and gesture tracking requires inertial sensors on wearable devices which are always inaccurate. To the best



(a) Double circle (b) Triangle (c) Loop back

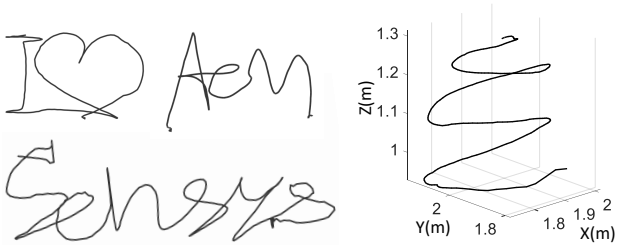
(a) Drawing error of CAT and AAMouse.



(a) Double circle (b) Triangle (c) Loop back

(b) Drawing error of BatTracker.

Figure 18: The drawing error comparison between different schemes. The results in figure (a) are from the CAT paper [19], and two figures (a, b) are drawn separately because we do not have the raw data in figure (a).



(a) " I ♡ ACM Sensys". (b) Spiral.

Figure 19: " I ♡ ACM Sensys" and a spiral drawn in the air.

Table 1: Allocated memory and CPU usage with different numbers of particles.

# particles	600	900	1200	1500	1800
Memory (*MB*)	9.56	11.52	12	12.33	13.02
CPU (screen on)	16.5%	17.74%	18%	18.36%	19.65%
CPU (screen off)	9.6%	11%	11.7%	12.01%	13.74%

of our knowledge, BatTracker is the first to achieve high precision tracking without any additional hardware, which makes it very convenient and attractive for indoor device tracking.

Limitations. Despite the high accuracy and infrastructure-free convenience we have, there exist multiple limitations which need to be addressed in future work.

i) Tracking range. We limit the range of reliable tracking in our current design within a space of $3 \times 3 \times 3m^3$, which is the size of a typical bedroom and sufficient for many popular scenarios such as video gaming, health rehabilitation training. However, the current design is not designed for use in a larger space. Simply increasing the gap between each acoustic measurement, thus increasing the range, will decrease the update rate, thus the accuracy of tracking trajectory.

ii) Device holding gesture. Due to the hardware limitations, users have limited range of device rotation to ensure reliable acoustic measurements. Rotating the device to an extreme orientation can cause serious data miss, thus a tracking failure.

iii) Reference Objects. For optimal performance, BatTracker uses three perpendicular planes as reference objects, which could be the ceiling and side walls in a room, or large furniture such as closets, cabinets, and tables. However, such objects are not always available. Besides, the ambient environment may change with the movement of the target, thus possibly the reference objects. Our current design does not consider adaptive reference object selection.

iv) Track loss problem. Despite the sophisticated algorithms that we propose for robust tracking, BatTracker still has a small chance of track loss, especially when the device is not held properly or is too far away from reference objects. Our current design can handle moderate data missing cases, such as human body blocking when people passing by. However the tracking can be lost if there are significant data loss during this period.

Future Work. Our future work for BatTracker focuses on improving tracking robustness, especially dealing with track loss problem. We will continue working on infrastructure-free tracking from the following aspects:

i) Fast track recovery. We plan to design a mechanism for automatical track loss detection and recovery. One intuition is from the match between inertial data and acoustic measurements. A lost track may show a strong inconsistency to the prediction from inertial sensor, hence the data likelihoods tend to be very small. By detecting such cases, we have the opportunity to detect track loss. Then we try to find stable track candidates in a short period, and associate these candidates to reference objects according to the match of inertial data.

ii) Utilize all the available objects. Our current design only utilizes three large objects as reference, hence a large portion of measurement information from other objects are not used. We will try to leverage all stable reflections, and build a more sophisticated algorithm to improve robustness. By selecting reference objects dynamically, the tracking range can be expanded.

iii) Customized hardware. BatTracker presents a novel approach for mobile device tracking in indoor environments. However, due to the hardware limitations, the device needs to be held in a particular way to minimize data missing problem. Adding customized omnidirectional, high-sensitivity microphones or multiple orthogonal microphones into existing devices can enhance the tracking robustness, while easing or eliminating the constraints on holding gestures.

iv) Variations among different smart phones. The audio pipelines and hardware performance may vary among different make/models of smart phones, which may impact the tracking performance. We only tested a few phones including HuaWei P9 and Samsung Note3, which prove to have similar performance. To make BatTracker ubiquitous, more comprehensive tests on different smart phones are needed as our future work.

8 RELATED WORK

Device tracking has been widely deployed for virtual reality and augmented reality, with different types of approaches in both academia and industry. We classify the related work according to their respective tracking techniques.

IMU Based Tracking. There are plenty of work using IMU sensors to track walking users in indoor environments. Some [9, 22] adopt dead-reckoning method using accelerometer and gyroscope readings from dedicated customized hardware, which are not available in mobile devices, and they are still prone to a large margin of errors from the double integration for distance computation. Zee [28] further leverages the indoor map as constraints to improve the tracking accuracy. In comparison, we focus on device tracking, which requires tracking in 3D with much higher accuracy. Our method leverages IMU sensors and microphone on commodity mobile devices, thus is more flexible with no external hardware requirements, and we demonstrate that it achieves better accuracy than previous IMU based gesture recognition [25] and device tracking [6] approaches.

Vision Based Tracking. Kinect [3], Wii [4] and LeapMotion [2] are all successful business products in the market for movement tracking with customized hardware including special cameras and depth sensors, whereas they all leverage dedicated devices and require line-of-sight, thus limiting their generality. Tanskanen *et al.* [34] combine vision and mobile sensory data to track the phone and reconstruct the 3D representation of an object, and Chen *et al.* [8] employ camera networks for multi-target tracking. However, vision approaches always encounter computation and energy bottlenecks on commodity mobile devices, and they are sensitive to lighting conditions and involve privacy issues. Compared with vision approaches, our method uses acoustic and inertial data, thus it can be easily deployed on commodity mobile devices, and audio signal processing is much more lightweight than images. Acoustic sensing is not subject to light conditions, and has no privacy concerns.

RF Based Tracking. Currently, mainstream indoor localization research depends on RF signatures from certain IT infrastructures. Among them, Cricket [30] assigns 6 indoor beacon nodes for RF and ultrasound transmission, ArrayTrack [37] deploys an array of 16 WiFi antennas, WiDraw [32] uses 25 WiFi transmitters and angle-of-time estimation to enable hands-free drawing, and Tagoram [39] leverages RFID for accurate tracking. However, they all rely on specialized devices to obtain high tracking accuracy. Besides, WiSee [27] uses Doppler shift of the WIFI signal for gesture recognition, while tracking a device in 3D space is much more complicated with much higher accuracy requirements. BatTracker leverages acoustics, which has a significantly lower propagation speed compared to wireless signal for high accuracy tracking on commodity devices.

Acoustic Based Tracking. Due to its slow propagation speed, acoustic signals do seem the best fit for 3D device tracking, and there are plenty of work pursuing higher tracking accuracy. Using Doppler shift of audio signals, researchers have designed Swadloon [11] for fine-grained indoor localization, and Spartacus [33] for gesture recognition. Besides, UbiK [35], AAMouse [40], LLAP [36], and FingerIO [21] leverage phase shift in received signals for near field finger gesture tracking. However, they can only track moving object within a small range around half meter, which is not suitable for tracking in room level. CAT [19] uses Frequency Modulated Continuous Waveform (FMCW) to track the phone movements at sub-cm accuracy, but it relies on external speakers and needs configuration efforts. In comparison, our method is a mobile device only approach with comparable sub-cm accuracy and combines inertial and acoustic data for their complementary strength, thus obtaining high tracking accuracy.

Despite device tracking, acoustics have also been widely used for ranging, indoor localization, and context sensing. BeepBeep [26] and SwordFight [42] estimate the distance between two mobile devices. Liu *et al.* [16] estimate the acoustic ranging distances between peer phones and treat them as constraints to improve localization accuracy. GuoGuo [18] uses an anchor network that transmits spatial beacon signals and obtains centimeter-level localization accuracy. BatMapper [44] measures the distances to multiple surrounding walls for indoor floor plan construction, and leverages acoustic for space classification. In terms of context sensing from audio signals, Yang *et al.* [38] detect driver phone use leveraging car speakers, ApenaApp [20] monitors chest and abdomen breathing movements, and DopEnc [41] identifies person encounters. Compared to the above work, BatTracker shares cross-correlation based echo detection similar to some work [10, 16, 17]. However, we focus on robust echo-object association despite echoes from many objects in cluttered environments, which has not been addressed in previous work.

9 CONCLUSION

In this paper, we propose *BatTracker*, which incorporates inertial and acoustic data for robust, high precision and infrastructure-free tracking in indoor environments. BatTracker leverages echoes from nearby objects instead of external infrastructure, thus requires less cost, deployment efforts, and is more convenient to use. A probabilistic multi-hypothesis tracking algorithm creates, prunes and

evolves multiple track hypotheses based on measurement evidences to accommodate uncertainty in device position. Experiments in real environments show that BatTracker can track a mobile device's movement in 3D space at sub-*cm* accuracy, comparable to the state-of-the-art infrastructure based approaches, while eliminating the needs of any additional hardware.

ACKNOWLEDGMENTS

We thank our shepherd Pei Zhang from CMU, as well as the anonymous reviewers who helped to improve this paper with their valuable feedback and comments. This work is supported in part by US NSF CSR 1513719, CCF 1652276, CCF 1730291, a 2016 Google Faculty Research Award, and China Postdoctoral Science Foundation (Grant 2017M610759 and 2017T100033).

REFERENCES

[1] 2017. HTC vive. (2017). http://www.vive.com.
[2] 2017. Leap Motion. (2017). http://www.leapmotion.com.
[3] 2017. Microsoft X-box Kinect. (2017). http://www.xbox.com/xbox-one/accessories/kinect.
[4] 2017. Nintendo Wii. (2017). http://www.nintendo.com/wii.
[5] 2017. Oculus. (2017). https://www.oculus.com/.
[6] Sandip Agrawal, Ionut Constandache, Shravan Gaonkar, Romit Roy Choudhury, Kevin Caves, and Frank DeRuyter. 2011. Using Mobile Phones to Write in Air. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*. 15–28.
[7] Samuel S Blackman. 2004. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine* 19, 1 (2004), 5–18.
[8] Y. Chen, Q. Zhao, Z. An, P. Lv, and L. Zhao. 2016. Distributed Multi-Target Tracking Based on the K-MTSCF Algorithm in Camera Networks. *IEEE Sensors Journal* 16, 13 (2016), 5481–5490.
[9] Eric Foxlin. 2005. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications* 25, 6 (2005), 38–46.
[10] Daniel Graham, George Simmons, David T Nguyen, and Gang Zhou. 2015. A Software-Based Sonar Ranging Sensor for Smart Phones. *IEEE Internet of Things Journal* 2, 6 (2015), 479–489.
[11] W. Huang, Y. Xiong, X. Y. Li, H. Lin, X. Mao, P. Yang, and Y. Liu. 2014. Shake and walk: Acoustic direction finding and fine-grained indoor localization using smartphones. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 370–378.
[12] Carine Hue, J-P Le Cadre, and Patrick Pérez. 2002. Tracking multiple objects with particle filtering. *IEEE transactions on aerospace and electronic systems* 38, 3 (2002), 791–812.
[13] Emmanuel C Ifeachor and Barrie W Jervis. 2002. *Digital signal processing: a practical approach*. Pearson Education.
[14] Rickard Karlsson and Fredrik Gustafsson. 2001. Monte Carlo data association for multiple target tracking. *Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE* 1, 13 (2001), 1–13.
[15] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. 2014. Accurate indoor localization with zero start-up cost. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 483–494.
[16] Hongbo Liu, Yu Gan, Jie Yang, Simon Sidhom, Yan Wang, Yingying Chen, and Fan Ye. Push the limit of WiFi based localization for smartphones. In *ACM Mobicom 2012*.
[17] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. 2015. Snooping keystrokes with mm-level audio ranging on a single phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 142–154.
[18] Kaikai Liu, Xinxin Liu, and Xiaolin Li. 2013. Guoguo: Enabling fine-grained indoor localization via smartphone. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 235–248.
[19] Wenguang Mao, Jian He, and Lili Qiu. 2016. CAT: high-precision acoustic motion tracking. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 69–81.
[20] Rajalakshmi Nandakumar, Shyamnath Gollakota, and Nathaniel Watson. 2015. Contactless sleep apnea detection on smartphones. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 45–57.
[21] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1515–1525.
[22] Hanna Nyqvist and Fredrik Gustafsson. 2013. A high-performance tracking system based on camera and IMU. In *Information Fusion (FUSION), 2013 16th International Conference on*. IEEE, 2065–2072.
[23] S. J. Orfanidis. 1996. *Optimum signal processing: An introduction*. 2nd Edition, Prentice-Hall, Englewood Cliffs, NJ.
[24] Savvas Papaioannou, Hongkai Wen, Zhuoling Xiao, Andrew Markham, and Niki Trigoni. 2015. Accurate positioning via cross-modality training. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 239–251.
[25] Taiwoo Park, Jinwon Lee, Inseok Hwang, Chungkuk Yoo, Lama Nachman, and Junehwa Song. 2011. E-Gesture: A Collaborative Architecture for Energy-efficient Gesture Recognition with Hand-worn Sensor and Mobile Devices. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. 260–273.
[26] Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan. 2007. BeepBeep: A High Accuracy Acoustic Ranging System using COTS Mobile Devices. In *ACM SenSys*.
[27] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home Gesture Recognition Using Wireless Signals. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*. 27–38.
[28] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. 2012. Zee: Zero-effort Crowdsourcing for Indoor Localization. In *Mobicom*.
[29] Donald Reid. 1979. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control* 24, 6 (1979), 843–854.
[30] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha. 2004. Tracking Moving Devices with the Cricket Location System. In *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*. 190–202.
[31] Andrew G Stove. 1992. Linear FMCW radar techniques. In *IEE Proceedings F-Radar and Signal Processing*, Vol. 139. IET, 343–350.
[32] Li Sun, Souvik Sen, Dimitrios Koutsonikolas, and Kyu-Han Kim. 2015. WiDraw: Enabling Hands-free Drawing in the Air on Commodity WiFi Devices. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. 77–89.
[33] Zheng Sun, Aveek Purohit, Raja Bose, and Pei Zhang. 2013. Spartacus: Spatially-aware Interaction for Mobile Devices Through Energy-efficient Audio Sensing. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '13)*. ACM, New York, NY, USA, 263–276. DOI: https://doi.org/10.1145/2462456.2464437
[34] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. 2013. Live Metric 3D Reconstruction on Mobile Phones. In *2013 IEEE International Conference on Computer Vision*. 65–72.
[35] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous Keyboard for Small Mobile Devices: Harnessing Multipath Fading for Fine-grained Keystroke Localization. In *Proceedings of ACM MobiSys*. 14–27.
[36] Wei Wang, Alex X Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 82–94.
[37] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: a fine-grained indoor location system. In *USENIX NSDI*.
[38] Jie Yang, Simon Sidhom, Gayathri Chandrasekaran, Tam Vu, Hongbo Liu, Nicolae Cecan, Yingying Chen, Marco Gruteser, and Richard P. Martin. 2011. Detecting Driver Phone Use Leveraging Car Speakers. In *ACM MobiCom*.
[39] Lei Yang, Yekui Chen, Xiang-Yang Li, Chaowei Xiao, Mo Li, and Yunhao Liu. 2014. Tagoram: Real-time Tracking of Mobile RFID Tags to High Precision Using COTS Devices. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom '14)*. 237–248.
[40] Sangki Yun, Yi-Chao Chen, and Lili Qiu. 2015. Turning a mobile device into a mouse in the air. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 15–29.
[41] Huanle Zhang, Wan Du, Pengfei Zhou, Mo Li, and Prasant Mohapatra. 2016. DopEnc: acoustic-based encounter profiling using smartphones. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. ACM, 294–307.
[42] Zengbin Zhang, David Chu, Xiaomeng Chen, and Thomas Moscibroda. 2012. SwordFight: enabling a new class of phone-to-phone action games on commodity phones. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 1–14.
[43] Mingmin Zhao, Tao Ye, Ruipeng Gao, Fan Ye, Yizhou Wang, and Guojie Luo. 2015. Vetrack: Real time vehicle tracking in uninstrumented indoor environments. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 99–112.
[44] Bing Zhou, Mohammed Elbadry, Ruipeng Gao, and Fan Ye. 2017. BatMapper: Acoustic Sensing Based Indoor Floor Plan Construction Using Smartphones. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 42–55.