RNN-Based Room Scale Hand Motion Tracking

Anonymous Authors
Affiliation
Email

ABSTRACT

Smart speakers allow users to interact with home appliances using voice commands and are becoming increasingly popular. While voice-based interface is intuitive, it is insufficient in many scenarios, such as in noisy or quiet environments, for users with language barriers, or in applications that require continuous motion tracking. Motion-based control is attractive and complementary to existing voice-based control. However, accurate and reliable room-scale motion tracking poses a significant challenge due to low SNR, interference, and varying mobility.

To this end, we develop a novel recurrent neural network (RNN) based system that uses speakers and microphones to realize accurate room-scale tracking. Our system jointly estimates propagation distance and angle-of-arrival (AoA) of signals reflected by hand, based on AoA-distance profiles generated by 2D MUSIC. We design a series of techniques to significantly enhance the profile quality under low SNR. We feed the profiles in a recent history to our RNN to estimate the distance and AoA, which exploits the temporal structure among consecutive profiles to remove the impact of noise, interference and mobility. With extensive evaluation, we show our system achieves 1.2–3.7 cm error within 4.5 m range, supports tracking multiple users, and is robust against ambient sound. To our knowledge, this is the first acoustic device-free room-scale tracking system.

CCS CONCEPTS

• **Human-centered computing → Interaction techniques**; **Ubiquitous and mobile computing**; • **Computing methodologies → Neural networks**.

KEYWORDS

Acoustic motion tracking, recurrent neural network, MUSIC

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6169-9/19/10…$15.00
https://doi.org/10.1145/XXXXXX.XXXXXX

ACM Reference Format:
Anonymous Authors. 2019. RNN-Based Room Scale Hand Motion Tracking. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 INTRODUCTION

Motivation: Talking with everyday appliance seemed like a fantasy a few years ago. With the emergence of smart speakers, the fantasy has become a reality. Users can issue voice commands to smart speakers, which control home appliances. Since Amazon Echo entered the market in 2015, there has been a rapid growth in the demand for smart speakers.

While voice-based control is attractive, it is not always suitable. For example, it is undesirable in quiet environment and causes disruption. Voice control also degrades significantly in noisy environment. In addition, developing smart speakers that can understand multiple languages for families with members speaking different languages and automatically identifying the language being used is still challenging and costly. Moreover, in several usage scenarios, such as interacting with a screen or selecting from many menu options, voice-based interface can be cumbersome. In comparison, motion-based control is appealing in these cases, and complements well with voice-based control.

Challenges: However, tracking hand motion poses significant challenges. A desirable approach should be (i) low cost, (ii) easy to deploy on smart speaker, (iii) accurate and reliable, and (iv) supporting room-scale tracking. (i) and (ii) suggest that we should use existing hardware on smart speakers and do not rely on specialized hardware. The current smart speakers come with Bluetooth, WiFi, and audio. WiFi based tracking has the longest range, but only achieves decimeter-level accuracy [1, 25, 28] due to its fast propagation speed and large wavelength, and falls short with respect to (iii). Audio based tracking has higher accuracy but limited range. Existing acoustic device-free tracking can only support a range up to 50 cm [41, 57, 62], which is insufficient for (iv) room scale tracking (i.e., supporting 2–4 m working range). This is not surprising due to fast signal attenuation over distance and multipath.

Approach: We use acoustic signals for tracking due to three reasons. First, smart speakers come with multiple microphones and speakers. Effectively exploiting this setup can

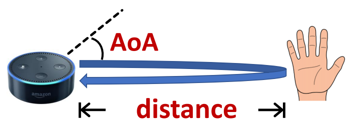


Figure 1: Tracking a hand with a smart speaker.

not only significantly enhance the tracking performance but also make it easy to deploy. Second, audio-based tracking provides high accuracy due to slow sound speed. Third, the sampling rate of acoustic signals is low so that all processing can be done in software. This makes it possible to customize the transmission signals and implement it on commodity devices.

We first investigate a variety of tracking algorithms and find that 2D MUSIC [58], which jointly estimates the distance and angle-of-arrival (AoA), is promising for room-scale tracking for two reasons. First, the SNR from hand reflection can be very low at room scale which significantly degrades the accuracy. Joint estimation increases the effective number of sensors, thereby improving accuracy. Second, multipath can dominate received signals at room scale and cause severe interference. Jointly estimating distance and AoA makes it easier to resolve multipath and mitigate interference since either distance or AoA of these paths is different.

While promising, 2D MUSIC alone is insufficient to achieve accurate room-scale tracking. Essentially, 2D MUSIC generates an AoA-distance profile based on received signals, and peaks in this profile indicates the positions of reflectors. Under low SNR, the 2D profile is noisy and peaks are severely biased from actual positions. Also, multipath and noise may introduce multiple peaks near the target reflector. It is challenging to identify the correct one. Moreover, movement may cause the actual position to deviate from the peak in the 2D profile. It is hard to directly measure the moving speed and compensate its effect under low SNR.

To address these issues, we develop complementary approaches based on machine learning and signal processing. On one hand, we design an recurrent neural network (RNN) to map the 2D profile to the AoA and distance of target reflector. We choose RNN to exploit the temporal structure between consecutive 2D profiles, and propose a network architecture to effectively learn and correct the impact of noise, multipath, and mobility. We also develop mechanisms to improve robustness against environments and reduce training effort.

Meanwhile, we design a series of signal processing techniques to improve the quality of AoA-distance profiles. They include i) time-domain beamforming to enhance SNR of received signals, ii) a framework to find the optimal MIC array that improves resolution while minimizing ambiguity under low SNR, iii) introducing middle chirps to increase the number of samples used for estimation, and iv) removing negative frequencies in signals to half the sinusoids seen by 2D MUSIC.

Furthermore, we develop a simple yet effective initialization procedure to reliably identify the user's hand in the presence of other reflectors at the beginning. This ensures the system to track the right target.

We implement our system on a Bela platform [9] and conduct extensive evaluation. We demonstrate our system achieves 1.2–3.7 cm median position error within 4.5 m range. Our RNN only requires 40 minutes for training and generalizes well to different users and locations. In addition, our system can track multiple users simultaneously, and is robust to ambient sound.

Our key contribution is to combine RNN with signal processing for motion tracking. While this combination may appear intuitive, choosing the right way of combining the two is important. To make RNN effective, we need to have appropriate features. We find that first applying signal processing to generate 2D profiles significantly helps simplify learning and remove environmental dependency. On the other hand, applying RNN to process the 2D profile improves robustness against noise, multipath, and mobility. Based on our framework of combining RNN with signal processing, we implement the first acoustic device-free room-scale tracking system and demonstrate its effectiveness through extensive evaluation. Moreover, the framework developed in our paper can be easily extended to other sensing tasks such as 1D ranging and AoA estimation.

2 PRIMER ON JOINT ESTIMATION

To localize a target, we can measure the propagation distance and angle-of-arrival (AoA) of its reflected signals using an array of microphones (MIC) and a speaker as shown in Figure 1. In this section, we introduce 2D MUSIC [58] to jointly estimate the distance and AoA. The key idea is to transform received signals into a 2D sinusoid whose frequencies are proportional to the AoA and distance. For this purpose, we let speaker send chirps whose frequency linearly sweeps from f to $f + B$ during period T. The signals reflected by the target are received by the MICs. For each MIC, we multiply the transmitted and received chirps and apply a low pass filter. The derived signals are given by

$$s(n, t) = \cos[-\frac{2\pi\Delta\cos(\theta)}{\lambda} \cdot n + \frac{4\pi B d}{T v_s} \cdot t + \phi], \quad (1)$$

where n is the MIC index, t is the sampling time, θ is the AoA, and d is the distance. Δ, λ, B, T, and v_s are the MIC separation, wavelength, bandwidth, chirp length, and sound speed, respectively. ϕ is a constant phase term. We see that $s(n, t)$ is a 2D sinusoid with frequencies $\Omega = -2\pi\Delta\cos(\theta)/\lambda$ and $\omega = 4\pi B d/(T v_s)$.

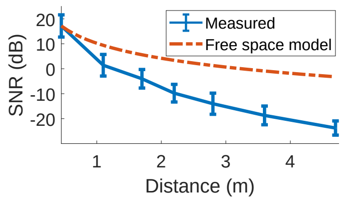


Figure 2: SNR vs distance.

2D MUSIC is widely used to estimate frequencies in the sum of sinusoids ($\sum_i e^{j(\Omega_i n+\omega_i t)}$) [58]. It generates a pseudo spectrum $P(\Omega, \omega)$ based on signal structure, and each peak in the spectrum indicates one frequency pair (Ω_i, ω_i). Since the frequencies are determined by AoA and distance in our context, we can replace (Ω, ω) with (d, θ) and derive an AoA-distance profile [32] as

$$P(\theta, d) = \frac{1}{(v(d) \otimes u(\theta))^H \mathcal{M}(v(d) \otimes u(\theta))}, \qquad (2)$$

where $u(\theta) = [1, e^{-j2\pi\Delta\cos(\theta)/\lambda}, \ldots, e^{-j(N-1)2\pi\Delta\cos(\theta)/\lambda}]$ and $v(d) = [1, e^{j4\pi BdT_s/(Tv_s)}, \ldots, e^{j(X-1)4\pi BdT_s/(Tv_s)}]$ (X is the time span). $\otimes$ is the Kronecker product. $\mathcal{M}$ is a matrix determined by signals. The peaks in the AoA-distance profile indicates the reflector positions.

3 POTENTIAL AND CHALLENGES

We do measurement-based study to understand the potential and limitations of using 2D MUSIC for room-scale tracking. For experiments, we use our Bela testbed as described in Section 5. We use a uniform MIC array with 4 elements and 4 cm span. We use 18–20 KHz chirps with 40 ms duration for 2D MUSIC, FMCW, and correlation. We use sinusoids at 18 KHz for phase-based method, AoA estimation, and Doppler measurement.

3.1 Potential

2D MUSIC is promising for room-scale tracking for two reasons. First, at the room scale, the signal-to-noise ratio (SNR) of reflected signals is very low. In our context, the SNR is defined as the ratio between the average magnitude of received chirps and noise. As shown in Figure 2, the SNR of hand reflection decays much faster than the free space model [49], because the hand is not flat and a portion of signals are scattered. According to Cramer-Rao bound [52], the minimum achievable error for AoA estimation is $6/(N^3 \cdot \text{SNR})$. Given -10 dB SNR and 4 MICs, this bound is 1°, which severely limits the tracking accuracy. Using 2D MUSIC can increase the effective number of sensors up to X times [28], where X is the number of sampling snapshots. This significantly reduces the Cramer-Rao bound and results in better AoA estimation accuracy than other approaches (*e.g.*, 1D MUSIC, ESPRIT, and beamforming).

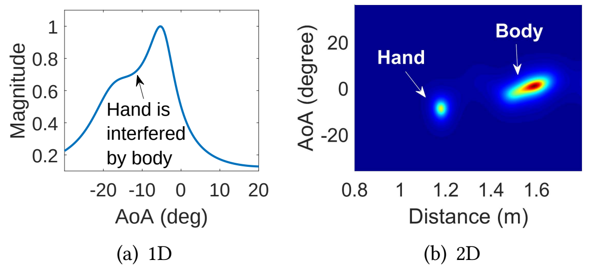


Figure 3: The profiles generated by MUSIC.

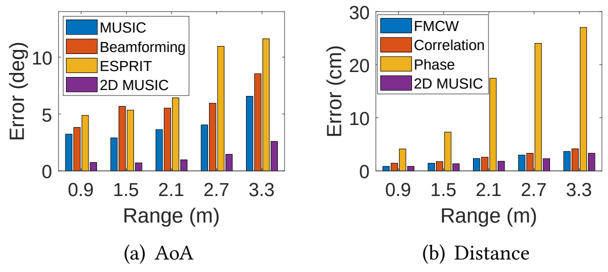


Figure 4: Comparison among various methods.

Second, the impact of multipath is severe in the room-scale tracking as hand reflection does not dominate received signals. 2D MUSIC can resolve multipath better since if either the distance or AoA of these paths is different, they are differentiable on the 2D profile. Figure 3 shows the profiles generated by 1D and 2D MUSIC, when a user stands at 1.5 m and raises his hand for tracking. Since the AoAs of hand and body reflection is close, their peaks are merged in the 1D profile. However, since they have different distances, 2D MUSIC can easily separate them and remove the impact of interference. Moreover, since 2D MUSIC increases the effective number of sensors [28], it works even if the number of multipath is larger than the number of MICs. 1D MUSIC and ESPRIT cannot support that case.

Due to the above reasons, 2D MUSIC achieves better tracking accuracy than alternative methods. Figure 4(a) compares 2D MUSIC with 1D MUSIC, beamforming, and ESPRIT for AoA estimation. Figure 4(b) compares it with phase [57], correlation [41, 64], and FMCW [33, 40] for distance measurement. In both cases, 2D MUSIC achieves the lowest estimation errors.

3.2 Limitations

2D MUSIC is promising but still insufficient for fine-grained tracking. It estimates the distance and AoA by locating the peaks in 2D profiles. While intuitive, this method can fail for the following reasons.

First, the peak can be wide at low SNR. Without noise, $\mathcal{M}$ and $v(d) \otimes u(\theta)$ in Eq. 2 is orthogonal if there is a reflection at (d, θ) [32]. As the SNR decreases, the orthogonality degrades,

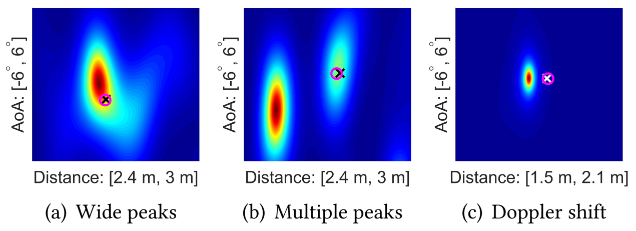


Figure 5: The positions of peaks, ground truth (crosses) and our RNN estimation (circles).

Radial speed (cm/s)	0.9	5.1	9.4	17
Distance error (cm)	2.3	4.5	6.7	9.4

Table 1: Distance error vs speed.

which makes the peaks become wide and low. In this case, the ground truth position may not be at the maximum point and is challenging to localize it. Figure 5(a) shows an example, where the ground truth is labeled by a cross.

Second, there can be multiple peaks near the ground truth due to multipath or noise, as shown in Figure 5(b). Since the ground truth peak may not be the highest, it is tricky to determine which peak is the correct one.

Third, under mobility, the ground truth position may lie outside the peak as shown in Figure 5(c). This happens as the frequencies in Eq. 1 are also affected by the moving speed. When the speed is greater than 5 cm/s, its impact is not negligible and can push the peak away from the ground truth position. Table 1 shows the distance errors estimated by 2D MUSIC under various speeds. We see the errors increase with the speed.

One way to handle this issue is to estimate the speed by sending sinusoids and measuring Doppler shift [61]. However, due to the low SNR and multipath, the estimation accuracy is low. At 2.7 m, the speed estimation error is 14.5 cm/s given the average hand speed as 17 cm/s. Another way to measure the speed is to send triangle chirps [50], where the chirp frequency first linearly increases and then decreases with the same rate. Ideally, we will observe two peaks in the profile for each reflector, and the difference between them is proportional to the speed. However, due to limited bandwidth (*i.e.*, 2 KHz), we can only observe one merged peak in our experiments. Thus, it cannot solve our problem.

In addition, 2D MUSIC also needs an effective initialization to ensure it tracks the right target.

4 RTRACK

We propose our tracking system RTrack. We use 2D MUSIC to generate profiles and then estimate distance and AoA based on them. Since manual peak selection from profiles is not effective due to noise, multipath, and mobility, we develop a

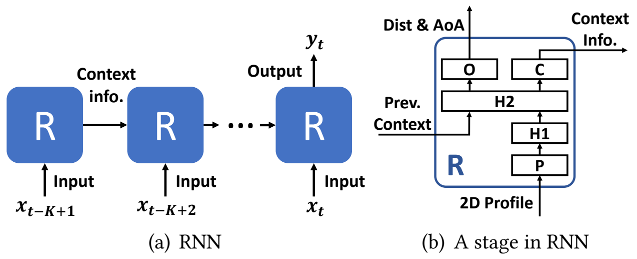


Figure 6: Network architecture.

RNN to automatically learn the mapping between the profile and target position (Section 4.1). Also, because the effectiveness of RNN depends on the profile quality, we further develop a series of signal processing techniques to improve the profile (Section 4.2). Finally, we design an initialization scheme to ensure tracking the right target (Section 4.3).

4.1 Joint Estimation using RNN

We design a recurrent neural network (RNN) [21] to automatically learn the mapping from a 2D profile to the AoA and distance of hand reflection. The key insight of using RNN is to exploit the temporal locality among consecutive tracking periods, as the hand position will not change too much over a short time. Even if the profile derived from the current period fails to provide clear information about the hand position, we can always get hints from the most recent periods. As a result, by jointly considering all profiles over these periods, RNN becomes robust to wide or multiple peaks on a single profile and achieves higher estimation accuracy, as shown in Figure 5. We see that the ground truth (marked by crosses) and the positions estimated by our RNN (marked by circles) match well in these challenging scenarios. Moreover, RNN can learn the Doppler shift by analyzing the consecutive 2D profiles and compensate its effect. We verify this point by investigating the hidden neurons in our RNN as shown in Table 2.

Our RNN contains K stages. These stages correspond to the 2D profiles extracted in the latest K periods. As shown in Figure 6(a), each stage has the same network structure (denoted as R). The last stage takes the current 2D profile as the input, and the earlier stages take the previous profiles as the input. Each stage also feeds certain *context* information to the next stage.

Network structure: An RNN consists of a series of identical network structure R. The design of R has profound impact on the effectiveness of RNN. Figure 6(b) illustrates our design of R, which we will explain below.

The network takes two inputs: the context from the previous stages and the current 2D profile. First, the 2D profile goes through an average-pooling layer P. This layer segments the input profile into multiple blocks, and each block contains $B \times B$ pixels. The output from P consists of average pixel magnitude of each block, Effectively, the P layer compresses the 2D profile by a factor $B \times B$. This significantly reduces the number of parameters in the network and prevents overfitting. Feeding the original 2D profile to the P layer is better than directly using a 2D profile at a coarse granularity because the latter may miss sharp peaks, which are common under high SNR.

The pooling layer is followed by a hidden layer $H1$. The layer extracts features from the compressed profile. Its output is concatenated with the context from the previous stage, and forwarded to the second hidden layer $H2$. This layer further extracts features from both the current and previous information. The sizes of $H1$, $H2$, and pooling factor B are determined during the network development and are specified in Section 5.

The output of $H2$ serves as the input to the output layer O and the context layer C. Layer O contains two neurons that represent the estimated distance and AoA. The output of Layer O are only used at the last stage, since they represent the current position.

Context layer: We introduce the context layer to automatically learn what information needs to be forwarded into next stage. Essentially, it is another hidden layer, which consists of multiple neurons and connects the consecutive stages in RNN. We use principal component analysis (PCA) [24] to design and analyze this layer. To understand our RNN in an intuitive way, we want to answer two questions: how many neurons are required to carry information from the previous stages and what information these neurons carry.

For this purpose, we train our RNN with n_c neurons in the context layer. For each training sample (n_s samples in total), we record the value of each neuron in the context layer at the last stage. We use a $n_s \times n_c$ matrix to represent the recorded context information. If there are dependent columns in this matrix, it indicates that some neurons are redundant and a fewer neurons can also carry all information. Thus, we perform PCA on the matrix to identify the major components. Interestingly, we find that no matter how large n_c is, we always have 5 major components, while the variance of the other components is 3 orders of magnitude lower. This indicates 5 neurons are sufficient to carry all context information required by our application.

To examine what information is carried, we project the context neurons onto their major components, and use p_i to represent coefficients on i-th component. We correlate p_i of all training samples with their ground truth distance (d), AoA (θ), radial and tangential speed (v_r and v_t), and radial and tangential acceleration (a_r and a_t). The results are summarized in Table 2. We find that the first three major components are strongly correlated with the distance, AoA,

	d	θ	v_r	v_t	a_r	a_t
p_1	0.94	0.04	0.06	0.00	-0.34	-0.09
p_2	0.20	-0.76	-0.14	-0.06	-0.08	0.30
p_3	-0.03	-0.24	0.79	0.13	0.28	0.10
p_4	-0.10	0.27	0.31	-0.22	0.03	-0.07
p_5	-0.01	0.10	-0.04	0.06	-0.06	-0.07

Table 2: Correlation between p_i and motion info.

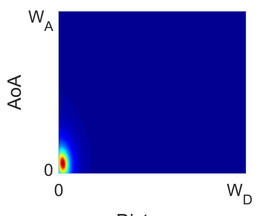
and radial speed. These results make intuitive sense and indicate that our RNN extracts appropriate information. The previous position is the most critical context information due to the continuity of the target movement. This helps narrow down the searching range for the current position. The radial speed provides the information about Doppler shift so that the network can remove its impact. The last two major components are less obvious.

Using detection window: To enhance both accuracy and efficiency, we only generate the profile over a small area around the target position, instead of covering the whole room. Specifically, assuming that we roughly know the target position, we select a *detection window* around it that covers the distance from B_D to $B_D + W_D$ and the AoA from B_A to $B_A + W_D$. We only evaluate the 2D profile in this window, and use it as RNN input. The output from RNN is the distance and AoA relative to the boundary of detection window. As shown in Figure 7, when the peak appears at the lower left corner of the profile, the RNN estimates both the distance and AoA as zeros. When the peak is at the upper right corner, the network estimates them as W_D and W_A. We can derive the final distance and AoA as the sum of the RNN output and the lower left corner of window (*i.e.*, (B_D, B_A)).

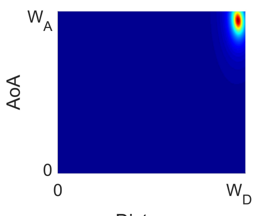
The selection of the detection window size (*i.e.*, W_D) depends on two factors. If the window is too large, the computation cost is high since a large area in the profile needs to be evaluated. If the window is too small, the hand may move out of the window frequently when the moving speed is high. In this case, we cannot detect the hand reflection in the cropped region. We choose the window size as 60 cm to balance these two factors.

We use the initialization scheme in Section 4.3 to determine the position of the initial window. During tracking, once detecting the hand close to the window boundary (within 5 cm), we reset the center of the window with the current hand position.

The benefits of using the detection window is manifold. First, it makes the network less sensitive to environment as the peaks for most reflections are outside the window. Second, the RNN trained at a specific distance or AoA can be applied to other distance or AoA by selecting a proper detection window. Third, we can track multiple users simultaneously by using multiple detection windows. Fourth, it



(a) Case 1



(b) Case 2

Figure 7: Using detection window.

significantly reduces the computation cost by only evaluating the profile in a local area.

Data augmentation: Data augmentation generates more training data using the existing training data. It generates new cases that may not be seen in the existing training data and prevents overfitting. It is cheap to generate lots of new data in our system. For each training sample, we can add a random shift $(\Delta d, \Delta\theta)$ to the actual distance and AoA, then shift the 2D profile by the same amount. From the RNN perspective, this is a new training sample where the target is located at a different position.

While we develop our RNN for joint estimation, our design is general and can be potentially applied to other approaches based on 1D profiles generated by FMCW, MUSIC, and correlation, or power delay profiles derived from CSI. The existing works generally select the highest peak from these profiles, but our RNN-based framework is more effective and robust against noise and interference in the profiles.

4.2 Enhance Profile Quality at Low SNR

According to [47, 52], the MUSIC performance is improved by (i) enhancing SNR, (ii) increasing the separation between MICs, (iii) using more samples for estimation, and (iv) reducing the number of sinusoids seen by MUSIC. Based on these insights, we develop several techniques to improve the quality of 2D profiles.

Time-domain beamforming: Our sample matrix (Eq. 1) has two dimensions: MIC indices and time. Given 4 MICs, 80 ms chirps, and 44.1 KHz sampling rate, the matrix size is 4×3528. Applying 2D MUSIC on a large matrix poses significant challenges. In fact, 2D MUSIC requires eigenvalue decomposition. Decomposing a large matrix incurs significant numerical errors [48], which degrades its performance. Moreover, the decomposition takes prohibitively long time since the algorithm has $O(n^3)$ complexity.

A natural approach is to down-sample time-domain signals. As illustrated in Figure 8, we take one column every K columns in the sample matrix (highlighted by blue), and apply 2D MUSIC to the down-sampled matrix. This approach

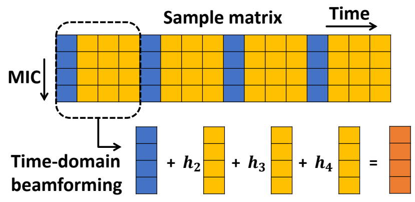


Figure 8: Time-domain beamforming.

is appropriate for high SNR cases, but is not ideal in low SNR regime because not all samples are used for tracking.

To effectively use all samples to enhance SNR while reducing the sample matrix size, we develop a mechanism based on *time-domain beamforming*. It is inspired by antenna beamforming where signals received by different antennas are added together to improve SNR. In our case, we combine signals sampled from different time as shown in Figure 8. Specifically, we divide the matrix into sub-matrices consisting of K consecutive columns. For each sub-matrix, we calculate the weighted sum of all its columns as illustrated by Figure 8. The weight of i-th column is denoted by h_i. Without loss of generality, we let $h_1 = 1$. To determine the other weights, we make the following important observations: the desired signal is a 2D sinusoid as described by Eq. 1 and the consecutive time-domain samples only differ by a phase shift equal to ωT_s, where $\omega = 4\pi Bd/(Tv_s)$ and T_s is the sampling interval. Thus, by assigning $h_i = e^{-j(i-1)\omega T_s}$, the desired signals in different columns will add up constructively, but the noise may cancel out each other. In this way, the SNR of samples is enhanced while the matrix size is reduced by a factor K. The selection of K depends on two factors. If K is too small, the derived matrix is still large. A too large K means only a few columns are left, which also causes degraded performance. Based on our experiments, we choose $K = 60$ to balance these factors.

As discussed, the weights h_i depend on ω, which is further determined by the propagation distance d. Since the user's hand position will not change significantly during consecutive tracking periods, we use the distance measured in the last period to approximate current d. To justify this approximation, we study the sensitivity of tracking performance when d is inaccurate. As shown in Figure 9, even if there is 30 cm distance error, the tracking error only increases marginally (*i.e.*, 4%) compared to the ideal case. In fact, given a distance error d_e, h_i differs from the optimal weights by $e^{-j(i-1)w_e T_s}$, where $w_e = 4\pi Bd_e/(Tv_s)$. When d_e is not large, h_i is still close to the optimal weights.

Our approach is different from previous time-domain beamforming [12, 17] by (i) designing weights based on special characters of signals (*i.e.*, a sinusoid), (ii) calculating weights using domain knowledge (*i.e.*, target positions in previous periods), (iii) selecting proper chunk size K used for beamforming under 2D MUSIC context.

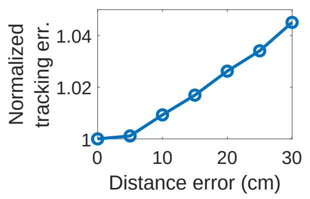
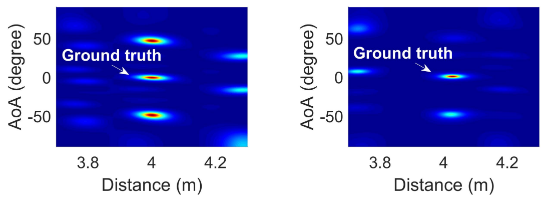


Figure 9: The sensitivity analysis.



(a) Uniform array (b) Non-uniform

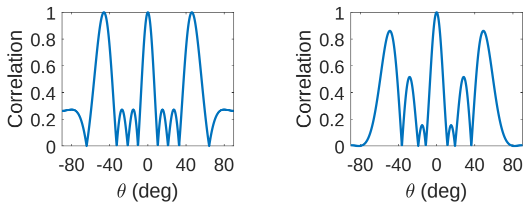
Figure 10: Ambiguity in 2D profiles.

Non-uniform microphone array: Another way to improve MUSIC performance is to increase the separation Δ between two consecutive MICs. However, if Δ is larger than half of the wavelength λ, there is ambiguity in estimating AoA. In this case, there exist two different angles θ_1 and θ_2 such that $-2\pi\Delta\cos(\theta_1)/\lambda = -2\pi\Delta\cos(\theta_2)/\lambda + 2\pi$. Based on Eq. 2, $P(\theta_1, d) = P(\theta_2, d)$ since $u(\theta_1) = u(\theta_2)$. If a peak is located at (θ_1, d_1) in the AoA-distance profile, there will be an identical peak at (θ_2, d_1). We cannot determine which one is due to a real signal. Figure 10(a) shows an example 2D profile with ambiguity, where Δ is 2.7 cm and λ is 2.1 cm. We observe two ambiguities at 50 and -50 degree with the same magnitude and pattern as the ground truth peak at 0 degree.

To increase MIC separation without introducing ambiguity, we use non-uniform array, which includes pairs of MICs with small separation to reduce ambiguity, and pairs of MICs with large separation to improve resolution [55]. Different from [55], we develop an optimization framework to determine array placement to minimize ambiguity under low SNR.

We use a vector $[m_1, m_2, \ldots, m_N]$ to describe a linear MIC array, where m_i is the distance between the i-th and first MICs. Thus, m_1 is always zero and m_N is the total size of the array. For a uniform array, $m_i = (i-1)\Delta$. Given this notation, $u(\theta)$ in Eq. 2 is generalized as $[e^{-j2\pi m_1 \cos(\theta)/\lambda}, e^{-j2\pi m_2 \cos(\theta)/\lambda}, \ldots, e^{-j2\pi m_N \cos(\theta)/\lambda}]$.

To determine the presence of ambiguity, we evaluate the correlation between $u(\theta)$ for the ground truth AoA θ_1 and all other angles. Figure 11 plots the correlation and shows many correlation peaks. The peak at the ground truth angle is the main peak, while the others are side peaks. If there is a side peak at θ_2 with magnitude equal to one, $u(\theta_2)$ is identical to $u(\theta_1)$. In this case, $P(\theta_1, d)$ and $P(\theta_2, d)$ have the same value based on Eq. 2 and θ_2 is an ambiguity for θ_1. Reducing the



(a) Uniform array (b) Non-uniform

Figure 11: Correlation between $u(0)$ and $u(\theta)$.

magnitude of side peaks reduces ambiguity. How much the side peak is away from 1 determines the margin to tolerate noise under low SNR scenarios. Therefore, we select the MIC array that maximizes the noise margin by solving following optimization:

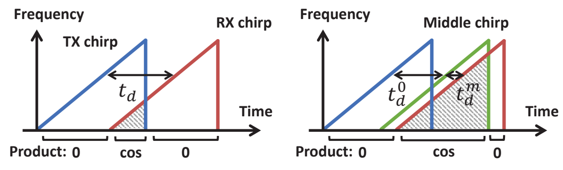
$$\begin{aligned} \max_{[m_1, m_2, \ldots, m_N]} \quad & \{1 - \max_{\theta_1}\{\text{side}(\theta_1)\}\}, \\ s.t. \quad & m_N = M, \\ & m_i + m_{N+1-i} = M, \end{aligned} \tag{3}$$

where the function side(θ_1) returns the highest side peak in the correlation plot for θ_1, and we search over all θ_1 to find the maximum side peak given a MIC array. The first constraint requires that the array size is a constant M, which guarantees that all candidate arrays give similar MUSIC accuracy [19]. The second constraint requires that the array be symmetric to the center so that the forward-backward smoothing can be applied, which removes the coherence among multipath signals and improves the MUSIC performance [42].

In our implementation, we select M as 8 cm so that the array can easily fit into a smart speaker. Solving the optimization gives us an optimal array [0, 3 cm, 5 cm, 8 cm]. Its correlation plot for the ground truth AoA at 0 degree is shown in Figure 11(b). The 2D profile is shown in Figure 10(b), where the ground truth peak is 6x as that of ambiguous peaks at ±50 degree, which makes it easy to identify the real one.

The above approach gives us the optimal MIC configuration. However, our system works for other setups albeit at the cost of moderate increase in the tracking errors or ambiguity.

Using middle chirps: To apply 2D MUSIC, we multiply transmitted and received chirps and use low-pass filtering to derive sinusoid signals (Eq. 1). The duration of sinusoid depends on the overlap between two chirps, since the product outside is simply zero, as shown in Figure 12(a). For room-scale tracking, the propagation delay of sounds is large (*e.g.*, around 30 ms for 5 m round trip). This significantly reduces the duration of derived sinusoid signals, since a typical chirp lasts for only tens of milliseconds. The reduced signal duration has a detrimental impact on MUSIC performance.



(a) Chirp overlap (b) Middle chirp

Figure 12: TX, RX, and middle chirps.

To address this issue, we introduce a middle chirp as shown in Figure 12(b). It shifts the transmitted chirp by t_d^0. If t_d^0 is close to the propagation delay of the received chirp (*i.e.*, t_d), the middle and received chirps have large overlap. To determine t_d^0, we need a rough knowledge about t_d. We use t_d estimated in the previous period for this purpose, since the hand position will not change much between consecutive periods.

We apply 2D MUSIC to sinusoid signals derived by multiplying middle and received chirps. The AoA estimated in this way is equal to that of the received chirp, because AoA is determined by the difference between the propagation delay to various MICs and shifting the transmitted chirp by the same amount for all MICs does not change the difference. The propagation delay t_d^m estimated in this way is the delay between the middle chirp and received chirp. The total delay for the received chirp is the sum of t_d^0 and t_d^m, as shown in Figure 12(b). Using middle chirps improves accuracy due to an increased number of samples used for estimation.

Another method is to use an extended version of transmitted chirp to multiply with the received chirp. The extended chirp has the same slope as the original one but sweeps a longer time so that it overlaps with the entire received chirp. However, by using middle chirps, the derived sinusoids has much lower frequencies, since $t_d^m \ll t_d$. This allows us to apply a low-pass filter with narrower passband, which helps remove more noise and improves the performance as shown by our experiment.

Removing negative frequencies: Different from RF signals, acoustic samples reported from hardware are real numbers. Therefore, we obtain real sinusoids as Eq. 1 after multiplying transmitted and received chirps. However, MUSIC is an approach to estimate frequencies for complex sinusoids [58], while a real sinusoid $\cos(p)$ consists of two complex sinusoids $e^{\pm jp}$. As result, the number of complex sinusoids seen by MUSIC is actually twice that of received reflections. According to [47], the increased number of sinusoids reduces MUSIC performance. To avoid that problem, we remove negative frequencies of signals by 1) performing FFT on the signals, 2) setting negative frequency bins as zeros, and 3) performing IFFT. In this way, each reflection corresponds to only one complex sinusoid. Removing negative frequencies reduces both signals and noise by half, so it does not change the SNR.

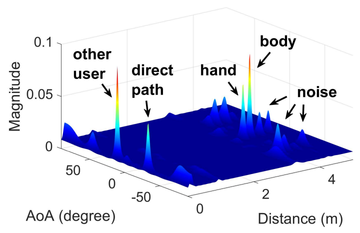


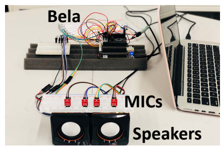
Figure 13: An AoA-distance profile for a room.

4.3 Initialization

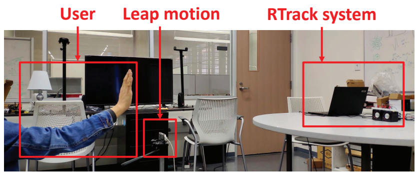
As discussed, each reflection corresponds to a peak in the AoA-distance profile. Our task is to track the one for hand reflection. This is challenging when the user is far away from the speaker, since hand reflection is not dominant. Figure 13 shows the profile for a room. After applying interference cancellation [34, 39] to remove static multipath, there are still many peaks due to dynamic background reflection (*e.g.*, user's body) and noise. The peak for the hand is not necessarily the strongest one.

Since it is hard to determine which peak corresponds to the hand based on one tracking period, we exploit the temporal relationship over multiple periods. We ask the user to perform an initial gesture and identify the peaks whose trajectory over time matches that gesture. Specifically, we let the user push and pull his hand during initialization. The target peaks should change accordingly over time. We use this pattern to identify the hand. The peaks for the direct path remain unchanged and those for dynamic background reflection and noise exhibit random patterns. Our evaluation in Section 6 shows that most time the user only needs to push once to clearly determine the initial hand position. For reliability, we let the user push twice.

Based on these observations, we extract peaks from 2D profiles during initialization. We cluster all peaks using the k-means algorithm [22], where k is the maximum number of peaks detected in one tracking period. We match the pattern of each cluster with the template gesture by applying dynamic time wrapping (DTW) to evaluate the distance between them. Also, we impose the following constraints to avoid false match: 1) the stretch ratio of time wrapping should be within 0.6 to 1.6 so that a candidate cluster is evenly stretched for matching; 2) the size of a cluster should be larger than 15 cm, as our initial gestures span 20–30 cm; 3) the number of peaks in a cluster should be larger than a threshold, as a cluster with less peaks is usually due to noise but it is easier to get a good match with the template gesture because there is only a small number of points in it. We rank the clusters satisfying these constraints based on their distance to the template gesture. If the distance of the



(a) System



(b) Testbed

Figure 14: Experiment setup.

best cluster is larger than a threshold, the system detects no initial gesture. Otherwise, we choose the last peak from that cluster as the initial hand position. Once the initial position is determined, we use it as the center of detection window and apply our RNN for continuous tracking.

4.4 Summary

Algorithm 1 summarizes RTrack. We use interference cancellation [34, 39] to remove background reflection from static objects (*e.g.*, furniture and walls in the room).

Algorithm 1 RTrack system.

```
1:  Place MIC array using our optimal configuration;
2:  Measure background reflection for cancellation;
3:  Perform initialization;
4:  while TRUE do
5:      Fetch new audio samples;
6:      Apply band-pass filter to remove out-of-band noise;
7:      Use interference cancellation to remove static multipath;
8:      Remove negative frequencies in signals;
9:      Multiply the middle and received chirps;
10:     Apply low-pass filter to obtain desired sinusoids;
11:     Derive 2D profile in the detection window with 2D MUSIC;
12:     Use RNN to map the profile to the distance and AoA;
13: end while
```

5 IMPLEMENTATION

Setup: As shown in Figure 14(a), we use two miniature speakers to send audios and four MICs [51] to receive signals. Many commercial smart speakers have similar numbers of speakers and microphones [4]. The separations between MICs are 3 cm, 2 cm, and 3 cm as derived in Section 4.2. The speakers and MICs are connected to a Bela platform [7–9], where TX and RX samples are cached. The platform is connected to a laptop with an Intel I5 processor and 8 GB memory. The processing is performed on the laptop using MATLAB. Note that our Bela board is equipped with a 1 GHz ARM Cortex-A8 single-core processor, but its capability is too weak to support on-board processing.

Audio: To send chirps, we generate a WAV file and use our speakers to play it. The chirp frequency is 18–20 KHz and the sampling rate is 44.1 KHz. The sound pressure at 0.5 m from the speakers is measured as 29.8 dB when our chirps are transmitted and 29.1 dB without playing anything. This indicates that our signals are barely noticeable. When sending such signals, the power consumption of each speaker is 0.22 W, measured using a multimeter. This value is significantly lower than the nominal maximum power of our speaker (2.5 W), because the speaker gain at the inaudible band is much lower than that at normal frequencies. We let two speakers send identical 80 ms chirps but make one start 40 ms behind the other. In this way, we can get one measurement every 40 ms alternatively from two speakers while benefiting from higher accuracy from the 80 ms chirps. Although the chirps from two speakers are overlapped, we are able to differentiate them based on the propagation delay [1].

Ground truth: As shown in Figure 14(b), we use Leap Motion [29] (LM) to get the ground truth of hand movements. However, LM has very limited working range. Our measurements show it can track the hand within 40 cm, and may lose tracking outside that range. To avoid losing tracking, we put LM directly below the hand in experiments. To get the ground truth, we measure the relative position between the speakers and LM, and add that to the coordinates reported by LM.

Training: The data collection is performed in our lab by a *single* user. The room size is 6 m×4.5 m. There are lots of equipments and furniture in the lab. Since they are static, we use interference cancellation [34, 39] to remove their impact. Our system is placed on a table as shown in Figure 14(b). The user stands 3 m away from the speakers and moves his hand in random patterns on a horizontal plane. The movement speed varies from 0 cm/s to 62 cm/s, and the average is 16 cm/s.

We collect 20 traces, which last about 40 minutes in total. These traces contain 58922 tracking periods. Each period is associated with the ground truth of hand position measured by LM. We use network time protocol [5] to synchronize the Bela platform and LM. For each trace, we augment four more traces through shifting AoAs and distances of the entire trace by a random amount selected from [1, 3] deg and [5, 15] cm, as described in Section 4.1. Note that we shift the entire trace by the same amount of AoA or distance to keeps the continuity of hand movements. After augmentation, we have 294610 samples. 90% of them are used for training and others are used for parameter tuning.

We use PyTorch [46] to build and train our network. We use Adam algorithm [27] for finding the optimal weights of our network. The batch size is 128. The learning rate is 0.002 initially and decays by a factor of 0.2 every 60 epochs. Refer to [23] for the definitions of batch size, learning rate, and epochs.

RNN: After tuning, our RNN is finalized as follows. The input 2D profile is a 60×60 image and each pixel covers 1 cm×1 cm region in the horizontal space. The pooling layer has the block size 3×3 so that the output is a 20×20 image. The first hidden layer has 20 neurons while the second one has 25 neurons. Both hidden layers are followed by ReLU activation functions. The output layer has 2 neurons and the context layer has 5 neurons. There are 15 RNN stages.

Testing: As mentioned, we collect training data from one user at our lab. For testing, we collect additional traces from 9 users including 7 men and 2 women. Their ages are between 20s to 40s. Testing traces are collected at 4 different locations, including our lab, a conference room with the size 7 m×4 m, a corridor with 1.5 m width, and an office area with a lot of cubes. For each user, we collect 50 traces at different distances ($[1, 4.5]$ m) and orientations ($[0, 70]$ deg) from the speakers. Each trace lasts 10 s. The users are asked to draw circles, triangles, rectangles, or random patterns. The users can move their hands with any speed they want. The maximum speed recorded is 67 cm/s, and the average is 13 cm/s. Before testing, we do NOT require users to practice our system. Therefore hand reflections from different users have diverse qualities. At the same distance, the SNR for different users varies up to 9 dB.

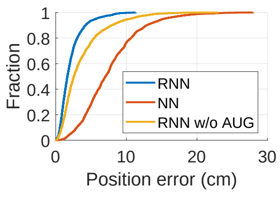
Metrics: We quantify the tracking error using three metrics: distance error, angle of arrival (AoA) error, and position error. The distance error is the difference between the measured and actual distance between the speaker and hand. Similarly define AoA error. The position error is the Euclidean distance between estimated hand position and the ground truth. We report the median and CDFs of these errors.

6 EVALUATION

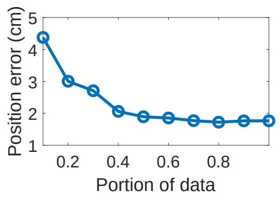
We first evaluate each design in our system and then overall performance under various scenarios. The experiment setup is described in Section 5. Unless otherwise specified, the default evaluation range is 2.7 m. The user is not walking during the experiments. Otherwise, the tracked trajectory is the net movement of the hand and body. In that case, one needs to decompose the hand and body movement by simultaneously tracking the peaks for hand and body in the 2D profile.

6.1 Micro Benchmark

Recurrent structure: We compare our RNN with classic neural network (NN). NN has the same structure as our RNN except that the context layer in Figure 6 is removed and it only takes the current 2D profile as the input. The performance of two networks is shown in Figure 15(a). Since the recurrent structure is removed, NN cannot leverage history information to improve the peak selection and compensate

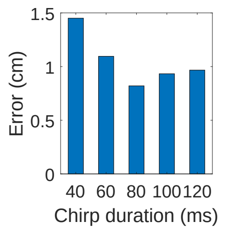


(a) RNN and data aug.

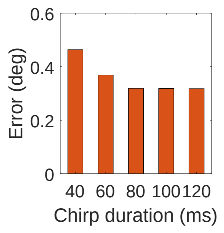


(b) Training sizes

Figure 15: The network performance.



(a) Distance error



(b) AoA error

Figure 16: Estimation error vs chirp length.

for Doppler shift. Thus, the median position error of NN is 7.1 cm and much higher than that of RNN, which is only 1.7 cm.

Training size: We evaluate the RNN performance using various portions of real training data. The selected data is augmented as described in Section 5 and then used for training. As shown in Figure 15(b), the position errors first reduce and then taper off when 60% of data is used. This indicates our training data is enough.

Data augmentation: We evaluate the benefit of using data augmentation. Figure 15(a) compares the tracking errors with and without data augmentation. Data augmentation reduces the median error from 2.5 cm to 1.7 cm due to less overfitting.

Chirp length: Increasing chirp length improves MUSIC performance in static scenarios by using more samples for tracking, but a too long chirp not only reduces responsiveness of tracking system, but also degrades accuracy when the distance or AoA has a noticeable change during a single chirp period. This explains why distance (AoA) errors first decrease and then increase with the chirp length in Figure 16. Based on the results, we use 80 ms chirps in our system.

Non-uniform array: We compare non-uniform versus uniform MIC arrays. We use three setups: (i) ULA8: uniform linear array spanning 8 cm in length, (ii) ULA4: uniform linear array spanning 4 cm, and (iii) NLA8: non-uniform array spanning 8 cm as specified in Section 5. Since increasing the MIC separation improves accuracy but increases ambiguity region, a uniform array has to trade off between accuracy and ambiguity. In comparison, a non-uniform array achieves the best of both worlds: it yields similar accuracy as ULA8

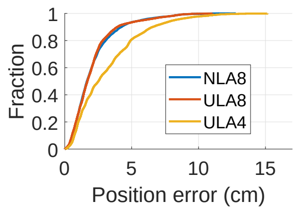
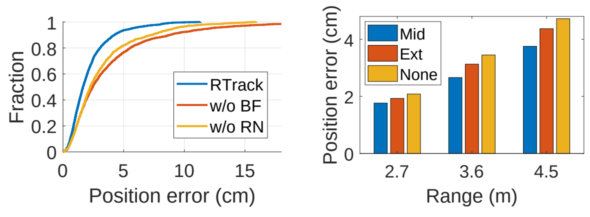


Figure 17: Position error vs MIC setups.

	True peak	Ambiguity 1	Ambiguity 2
ULA8	1	1.000	1.000
NLA8	1	0.082	0.052
ULA4	1	0.067	0.047

Table 3: Normalized magnitude of ambiguity peaks.



(a) BF and RN (b) Using middle chirps

Figure 18: Profile enhancement.

(shown in Figure 17) and low ambiguity as ULA4 (shown in Table 3). In this table, we show the magnitude of two highest ambiguity peaks normalized by the magnitude of the ground truth peak.

Time-domain beamforming (BF): We evaluate the benefit of BF by comparing it with down-sampling as described in Section 4.2. As shown in Figure 18(a), the median position error is reduced from 2.3 cm to 1.7 cm by using BF, while 80-percentile error is reduced from 4.7 cm to 3.0 cm. The improvement is higher when tracking errors are large. In fact, large errors happen at low SNR, and the SNR enhancement provided by BF is most beneficial in this case.

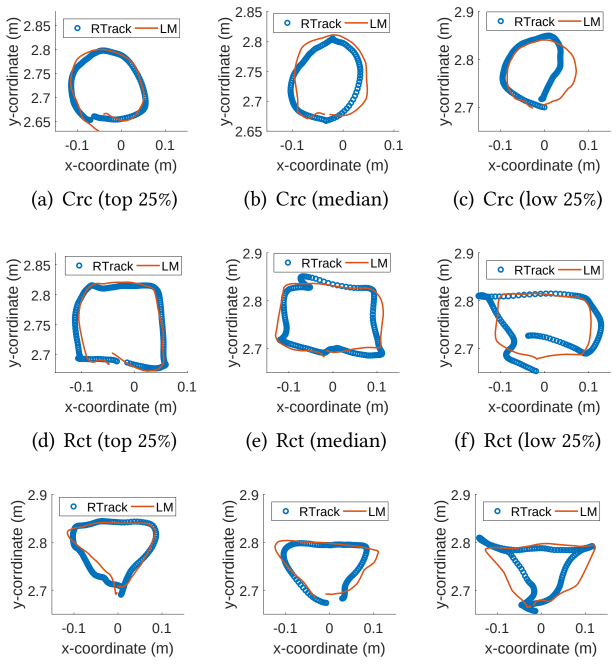
Removing negative frequencies (RN): Figure 18(a) shows that applying RN decreases the position errors by reducing the number of complex sinusoids in the signals.

Using middle chirps: We compare the performance with and without middle chirps. We also evaluate the extended chirp (Ext) as described in Section 4.2. As shown in Figure 18(b), using middle chirps (Mid) reduces the position errors by 18–29% at different distances. Compared with Ext, using middle chirps achieves 9–17% gain due to more aggressive low-pass filtering.

Initialization: We evaluate our initialization. To test the success rate (*i.e., the percentage of initial gestures are recognized*), we collect the traces where the user pushes and pulls at 2.1 m, 2.7 m, 3.6 m, and 4.5 m away from the speaker. To test the false positive (denoted by "FP", *i.e., how many times per minute the system detects initial gestures when no user does them*), we collect two 30-minute traces where the user plays games on mobile phone in the first one (denoted by "St"), and randomly walks in the room in the second (denoted by "Mv"). We set the DTW distance threshold as 5 cm to balance the success rate and false positive. As shown in Table 4, the success rate drops as the distance increases due to the reduced SNR. The false positive in the second trace (Mv) is higher because some movements have similar patterns to the initial gestures. Pushing and pulling twice helps significantly improve the performance.

Processing time: Table 5 shows the breakdown of the processing time for RTrack. Pre-processing (Pre-proc) includes interference cancellation, removing negative frequencies, applying middle chirps, and filtering. 2D MUSIC involves estimation of covariance matrix (Covar), eigenvalue decomposition (Eig), and generating 2D profile. The total time is 36.3 ms, which supports real-time tracking given the 40 ms tracking period. The average CPU usage is 54% on our laptop.

Beside the processing delay, our system also experiences the latency due to 1) collecting audio samples using Bela,



(a) Crc (top 25%) (b) Crc (median) (c) Crc (low 25%)

(d) Rct (top 25%) (e) Rct (median) (f) Rct (low 25%)

(g) Tri (top 25%) (h) Tri (median) (i) Tri (low 25%)

Figure 19: Drawing samples.

	Success rate (%)				FP (times/min)	
Push	2.1 m	2.7 m	3.6 m	4.5 m	St	Mv
Once	100	100	97	94	0.0	0.33
Twice	100	100	100	97	0.0	0.03

Table 4: The initialization performance.

Pre-proc	Covar	Eig	Profile	RNN	Total
10.4	5.2	2.2	17.6	0.9	36.3

Table 5: Running time of RTrack (unit: ms).

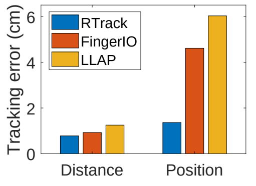
which is 1 ms [10]; 2) sending samples to the laptop using USB connection, which is less than 1 ms [43].

6.2 System Benchmark

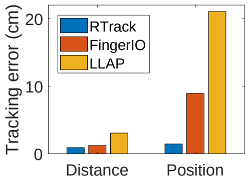
Drawing samples: To illustrate the performance of RTrack, we show drawing traces for circles, rectangles, and triangles. For each shape, we sort the traces of different users based on the median position errors. The traces ranked at 25%, 50%, and 75% are shown in Figure 19, and the corresponding errors are around 0.9 cm, 1.7 cm, and 2.7 cm, respectively.

Different approaches: We first compare RTrack with near-field tracking methods: FingerIO [41] and LLAP [57]. For fair comparison, all methods use one speaker and four MICs. RTrack sends 40 ms chirps. FingerIO sends 40 ms OFDM symbols at 18–20 KHz. LLAP sends 5 sine waves in 18–20 KHz. For FingerIO and LLAP, we first estimate the distance between the hand and each microphone, and then find the hand position where the distances to all MICs match our measurement best (*i.e.*, the mean square error is minimized). The tracking errors of these methods are shown in Figure 20(a) and (b). At 0.5 m, the ranging errors for RTrack, FingerIO, and LLAP are 0.8 cm, 0.9 cm, 1.2 cm, respectively, while the position errors are 1.3 cm, 4.6 cm, 6.0 cm, respectively. At 1 m, the ranging errors are 0.9 cm, 1.2 cm, 3.0 cm, respectively, while position errors are 1.4 cm, 8.9 cm, 21 cm, respectively. In fact, both FingerIO and LLAP use triangulation to find the hand position. However, when the MIC span is small (*e.g.*, 8 cm), the triangulation error is large. Instead, RTrack jointly estimates AoA and distance based on which we can derive the hand position, which is more accurate than triangulation.

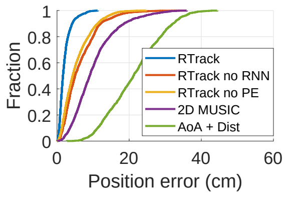
We also compare RTrack with the following methods at 2.7 m: 1) RTrack replacing RNN with the "optimal" peak selection (*i.e.*, select the peak closest to the ground truth and hence is better than any peak selection heuristics) but keeps the profile enhancement techniques (PE) in Section 4.2; 2) RTrack without PE (except the non-uniform array) but keep RNN; 3) a plain 2D MUSIC but using non-uniform array and initial localization to ensure tracking the right target; 4) estimating AoA and distance individually but using PE techniques. These results are shown in Finger 20(c) We see that RTrack perform the best, and achieves 1.7 cm median position error. Removing RNN or PE increases the median error to 4.3 cm and 5.0 cm respectively, which demonstrates their benefits. 2D MUSIC has the median error of 9.0 cm, much higher than RTrack. The median error of using individual AoA and distance estimation is 21.3 cm due to erroneous AoA estimation.



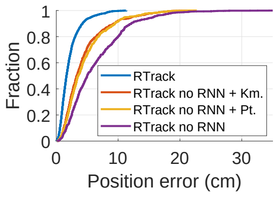
(a) Near field (0.5 m)



(b) Near field (1 m)

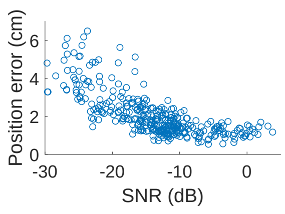


(c) Disable RNN or PE

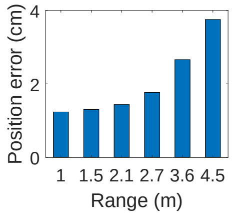


(d) Kalman & particle

Figure 20: Performance of various approaches.



(a) SNR

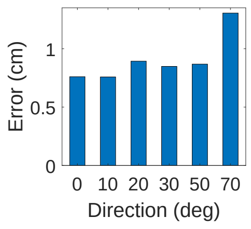


(b) Range

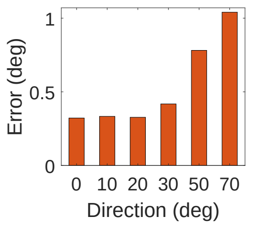
Figure 21: Position error vs SNR and range.

In addition, we compare RNN with Kalman and particle filters [44]. We modify RTrack by replacing RNN with the optimal peak selection followed by Kalman (Km.) or particle filters (Pt.). As shown in Figure 20(d), RNN outperforms filter-based approaches. The Kalman and particle filters are sensitive to the choice of parameters (*i.e.*, the covariance matrix of state and measurement equations), and we choose these parameters that give the lowest median errors over all traces.

Impact of SNR and range: Our experiments show that hand reflection from different users has different SNR even at the same distance. The difference comes from: (i) hand size (a large hand tends to have large SNR), (ii) hand posture (*e.g.*, the SNR tends to be high if the hand lines up with the speakers and MICs). Thus, we characterize the performance of RTrack in terms of both SNR and range. In Figure 21(a), we evaluate the SNR and median position error of each testing trace, and plot SNR-error pairs for all traces. In Figure 21(b), we show the performance for traces collected at varying ranges. As we can see, the position errors first reduce with SNR (distance decreases), and then taper off. In this case, factors like signal distortion and calibration errors (*e.g.*, the MIC separation) dominate.



(a) Distance error



(b) AoA error

Figure 22: Tracking error vs direction.

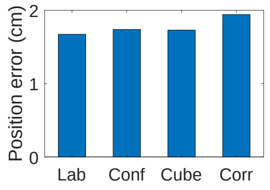


Figure 23: Performance at various locations.

Impact of direction: Figure 22 plots the distance and AoA errors as the users stand at various directions, where 0 deg means that the users exactly face the microphone array and 90 deg means that the user is on the line defined by the array. As we can see, the distance errors are relatively stable across all directions, while the AoA errors significantly increase when the direction is larger than 30 deg. In fact, when the AoA (*i.e.*, θ) is large, a small estimation error on the frequency $\Omega = -2\pi\Delta\cos(\theta)/\lambda$ in Eq. 1 will translate to a large error on θ, since $\frac{d\theta}{d\Omega}$ is large in this case.

Impact of locations: We evaluate RTrack by training in our lab and testing at 4 locations: our lab, a conference room (Conf), an office area (Cube), and a narrow corridor (Corr). The median position errors at 4 locations are shown in Figure 23. We see the results are similar across different locations: the median errors are all in the range of 1.7–2.0 cm, which indicates that our system is robust to the environment. The reasons are two-fold. First, we perform interference cancellation so that reflections due to static objects in the environment are removed and have little impact on our RNN. Second, the input of our RNN is a cropped area of the 2D profile, while most reflections from the environment are not inside this area and hence have no effect on our RNN.

Impact of speakers: We evaluate RTrack with various speakers, represented by Speaker E [18], A [6], D [15], and M [37]. The first speaker is our default one, and only that one is used for collecting training data. We tune the volume of these speakers so that they have the same transmission power. For each speaker, we use its highest available frequency band. *i.e.*, 18–20 Khz, 18–20 Khz, 19–21 Khz, and 17–19 Khz, for Speaker E, A, D, and M, respectively. The median position errors with these speakers are shown in Figure 24. Although the last three speakers are not seen by the training data, they perform closely to our default speaker. This indicates that our RNN does not overfit to a specific speaker.

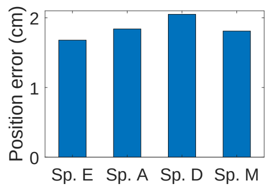


Figure 24: Performance with different speakers.

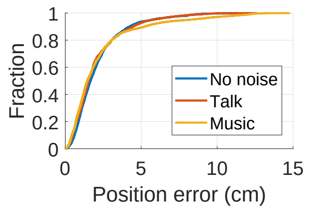


Figure 25: Performance with ambient sounds.

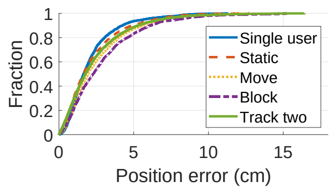


Figure 26: Tracking with multiple users.

Impact of ambient sound: To study the impact of ambient sound, we play music or talk while performing the tracking experiments. The sound sources are at 0.5 m from the MIC array. The volume for music playing is the same as that used for sending chirp signals, and people talk using their normal volume. As shown in Figure 25, the tracking accuracy is similar. This is not surprising since our tracking system uses frequencies higher than those in music and speech.

Tracking multiple users: Next, we evaluate RTrack with multiple users in following cases: 1) tracking one user while the other user is almost static (*e.g.*, playing games on mobile phones); 2) tracking one user while the other randomly walks in the room but never block the direct path between the tracked user and speakers; 3) tracking one user while the other walks around and blocks the direct path every 15s; 4) tracking two users simultaneously who are at the same distance from the speakers and separated by 1 m. The results are shown in Figure 26, and four cases are labeled as "static", "move", "block", and "track two" respectively. We see that RTrack works in all cases, but the accuracy degrades when the interference is too close to the hand or the direct path is blocked. Also, RTrack can track two users simultaneously because two users are separated by the AoA and can be tracked independently using two detection windows just as a single user case.

7 RELATED WORK

We describe existing work related to tracking, gesture recognition, neural network, and 2D MUSIC.

Acoustic based tracking: A number of systems have been proposed to track motion based on acoustic signals. For example, AAMouse [61] uses Doppler shift to track a mobile device. CAT [33] further uses FMCW to enhance the accuracy. LLAP [57] and Strata [62] use the phase of audio signals for device-free tracking, while FingerIO [41] uses correlation. DroneTrack [35] estimates the distance between a drone and a user using MUSIC algorithm. Different from existing device-free acoustic tracking systems, our system is the first that enables room-scale device-free tracking by applying RNN and developing signal processing techniques to support low SNR.

RF based tracking: RF signals have also been widely used for motion tracking. WiTrack [2] uses FMCW to achieve decimeter accuracy. WiDraw [53] uses CSI to estimate AoA and achieves 5cm median tracking error using 25 WiFi APs. WiDeo [25] uses phase synchronized RX chains to achieve 7 cm median error. [66] uses 60 GHz radios to track static objects with line-of-sight at a cm-level accuracy. [59] improves the accuracy by using 60 GHz steerable antennas with narrow beamwidth and fine-grained azimuth rotation. [55] uses the phase of RFID signals for motion tracking to achieve 3.7 cm accuracy. [31] emulates a very large bandwidth by leveraging the underlying physical properties to achieve sub-centimeter tracking accuracy. [56] exploits variation in the phase and RSS changes of RFID signals when penetrating different materials to identify material type and image the internal horizontal cut of a target.

Gesture recognition: There has been significant work on gesture recognition. WiSee [45] extracts the Doppler shift from WiFi signals to recognize nine gestures. [3] uses WiFi CSI based signature to determine the key typed by a user. AllSee [26] relies on RFID tags and power-harvesting sensors for gesture recognition. WiAG [54] recognizes user gestures irrespective of their orientations. Mudra [63] uses two WiFi antennas to recognize gestures without user training. Compared with gesture recognition, motion tracking provides more degree of freedom for human-computer interaction.

Neural network: RF-Echo [14] applies a neural network (NN) for time-of-flight estimation. Compare to NN, RNN can leverage the information from multiple tracking periods, which greatly improves the performance. RF-Pose [65] develops a CNN to detect users' poses based on RF signals. It concatenates all RF measurements over multiple periods as the network input. Differently, RNN takes the measurement in each period as the input of one stage, and each stage shares the same parameters. This greatly helps reduce the network size and prevent overfitting. [13, 30] measure RSS to multiple APs and train a RNN to recognize the location based RSS measurements over time. RSS is highly sensitive to environment. To avoid that, we train our RNN to understand 2D profiles, where each multipath in the environment is separated by different AoAs or distances. We develop an effective network structure to learn how to locate the target reflection from the profile. [38] estimates the AoA of an audio source based on TDoA of two microphones. It trains a RNN using estimated AoAs as input to predict a possible AoA range in near future. In contrast, our RNN is used to estimate AoA itself. DeepSense [60] develops a deep NN to fuse raw measurements of different sensors for sensing applications. In RTrack, we first apply 2D MUSIC to extract AoA-distance profiles from raw measurements, and design an efficient and compact network architecture accordingly. This significantly reduces the network size and saves the effort for collecting data and training. [16, 20, 36] apply RNN to detect and associate multiple users in consecutive video frames. Since these applications of RNN are different from us, their network structure cannot be applied to our case.

2D MUSIC: 2D MUSIC is applied to estimate AoA and propagation distance of incoming signals [11, 28, 32]. Our system differs from them in the following ways. First, directly applying 2D MUSIC does not provide sufficient accuracy as shown by our experiments. We develop several signal processing techniques to significantly reduce tracking errors under low SNR. Second, we design a RNN to read the profiles generated by 2D MUSIC and automatically determine AoA and distance. This greatly helps reduce the errors caused by Doppler effect, peak bias due to strong noise, and multiple peaks near the ground truth. Third, [28] is device-based tracking, requiring users to hold RF transceivers, while our system is device-free tracking. Also, [28] needs multiple APs at various positions, while our system uses speakers and MICs on a single device.

8 CONCLUSION

In this paper, we develop the first room-scale tracking system that can enable a user to interact with smart speakers using hand movement. The key technique is a novel framework combining recurrent neural network with 2D MUSIC for jointly estimating the AoA and distance of hand reflection. We implement our system on a development platform for smart speakers and demonstrate the feasibility of room-scale tracking using extensive experiments.

There are a few limitations in our system. First, we require the user's hand facing the microphone array during tracking to prevent the reflection signals from being too weak, which may affect user experience in some scenarios. Second, our system does not support the case where the light-of-sight path is blocked for a long period. Moving forward, we will explore effective solutions for these problems.

REFERENCES

[1] Fadel Adib, Zachary Kabelac, and Dina Katabi. 2015. Multi-person localization via RF body reflections. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. 279–292.

[2] Fadel Adib, Zach Kabelac, Dina Katabi, and Rob Miller. 2014. WiTrack: Motion Tracking via Radio Reflections off the Body. In *Proc. of NSDI*.

[3] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. 2015. Keystroke Recongition using WiFi Signals. In *Proc. of ACM MobiCom*.

[4] Apple 2018. HomePod Environmental Report. https://images.apple.com/environment/pdf/products/homepod/HomePod_PER_jan2018.pdf.

[5] Arch Linux 2019. Network Time Protocol. https://wiki.archlinux.org/index.php/Network_Time_Protocol_daemon.

[6] Arvicka 2016. Computer Speakers. https://www.amazon.com/ARVICKA-Computer-Multimedia-Smartphones-Projectors/dp/B01KC7WGQQ.

[7] BeagleBoard.org 2017. BeagleBoard. http://beagleboard.org/bone.

[8] Bela Team 2017. Bela Audio Expander. https://github.com/BelaPlatform/Bela/wiki/Using-the-Audio-Expander-Capelet.

[9] Bela Team 2017. Bela Platform. https://bela.io.

[10] Bela Team 2017. What is Bela. https://github.com/BelaPlatform/Bela/wiki/What-is-Bela%3F.

[11] Francesco Belfiori, Wim van Rossum, and Peter Hoogeboom. 2012. 2D-MUSIC technique applied to a coherent FMCW MIMO radar. (2012).

[12] Jacob Benesty, Israel Cohen, and Jingdong Chen. 2017. *Fundamentals of Signal Enhancement and Array Signal Processing*. John Wiley and Sons.

[13] Luis A Castro and Jesus Favela. 2005. Continuous tracking of user location in WLANs using recurrent neural networks. In *null*. IEEE, 174–181.

[14] Li-Xuan Chuo, Zhihong Luo, Dennis Sylvester, David Blaauw, and Hun-Seok Kim. 2017. RF-Echo: A Non-Line-of-Sight Indoor Localization System Using a Low-Power Active RF Reflector ASIC Tag. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. ACM, 222–234.

[15] Dell 2009. AX510 Computer Speakers. https://www.amazon.com/Dell-AX510-2-0-Speaker-System/dp/B002R5AO1G.

[16] Julie Dequaire, Peter Ondrúška, Dushyant Rao, Dominic Wang, and Ingmar Posner. 2018. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *The International Journal of Robotics Research* 37, 4-5 (2018), 492–512.

[17] Robert Dougherty. 2004. Advanced time-domain beamforming techniques. In *10th AIAA/CEAS Aeroacoustics Conference*. 2955.

[18] Earise 2013. AL202 Computer Speakers. https://www.amazon.com/AL-202-High-fidelity-Acoustics-Gemini-Doctor/dp/B00CXLQGSY.

[19] Carine El Kassis, José Picheral, and Chafic Mokbel. 2010. Advantages of nonuniform arrays using root-MUSIC. *Signal Processing* 90, 2 (2010), 689–695.

[20] Kuan Fang. 2016. Track-RNN: Joint Detection and Tracking Using Recurrent Neural Networks. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain*.

[21] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[22] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques*. Elsevier.

[23] Jason Brownlee 2019. A gentle introduction to Mini-Batch Gradient Descent and How to Configure Batch Size. https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/.

[24] Ian Jolliffe. 2011. Principal component analysis. In *International encyclopedia of statistical science*. Springer, 1094–1096.

[25] Kiran Joshi, Dinesh Bharadia, Manikanta Kotaru, and Sachin Katti. 2015. WiDeo: Fine-grained Device-free Motion Tracing using RF Backscatter. In *Proc. of NSDI*.

[26] Bryce Kellogg, Vamsi Talla, and Shyamnath Gollakota. 2014. Bringing Gesture Recognition to All Devices.. In *NSDI*, Vol. 14. 303–316.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. Spotfi: Decimeter level localization using WiFi. In *ACM SIGCOMM Computer Communication Review*, Vol. 45(4). ACM, 269–282.

[29] Leap Motion, Inc. 2010. Leap Motion. http://www.leapmotion.com.

[30] Yuan Lukito and Antonius Rachmat Chrismanto. 2017. Recurrent neural networks model for WiFi-based indoor positioning system. In *Smart Cities, Automation and Intelligent Computing Systems (ICON-SONICS), 2017 International Conference on*. IEEE, 121–125.

[31] Yunfei Ma, Nicholas Selby, and Fadel Adib. 2017. Minding the Billions: Ultra-wideband Localization for Deployed RFIDs. In *Proc. of ACM MobiCom*.

[32] Gleb O Manokhin, Zhargal T Erdyneev, Andrey A Geltser, and Evgeny A Monastyrev. 2015. MUSIC-based algorithm for range-azimuth FMCW radar data processing without estimating number of targets. In *Microwave Symposium (MMS), 2015 IEEE 15th Mediterranean*. IEEE, 1–4.

[33] Wenguang Mao, Jian He, and Lili Qiu. 2016. CAT: High-Precision Acoustic Motion Tracking. In *Proc. of ACM MobiCom*.

[34] Wenguang Mao, Mei Wang, and Lili Qiu. 2018. AIM: Acoustic Imaging on a Mobile. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 468–481.

[35] Wenguang Mao, Zaiwei Zhang, Lili Qiu, Jian He, Yuchen Cui, and Sangki Yun. 2017. Indoor Follow Me Drone. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 345–358.

[36] Anton Milan, Seyed Hamid Rezatofighi, Anthony R Dick, Ian D Reid, and Konrad Schindler. 2017. Online Multi-Target Tracking Using Recurrent Neural Networks.. In *AAAI*, Vol. 2. 4.

[37] Moloroll 2018. Computer Speakers. https://www.amazon.com/Moloroll-Computer-Speakers-Desktop-Distortion/dp/B07BD1Q4DD.

[38] John C Murray, Harry Erwin, and Stefan Wermter. 2005. A hybrid architecture using cross-correlation and recurrent neural networks for acoustic tracking in robots. In *Biomimetic Neural Learning for Intelligent Robots*. Springer, 73–87.

[39] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, and Venkata N. Padmanabhan. 2013. Dhwani : Secure Peer-to-Peer Acoustic NFC. In *Proc. of ACM SIGCOMM*.

[40] Rajalakshmi Nandakumar, Shyam Gollakota, and Nathaniel Watson. 2015. Contactless Sleep Apnea Detection on Smartphones. In *Proc. of ACM MobiSys*.

[41] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proc. of ACM CHI*. 1515–1525.

[42] S Unnikrishna Pillai and Byung Ho Kwon. 1989. Forward/backward spatial smoothing techniques for coherent signal identification. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37, 1 (1989), 8–15.

[43] Ploytec GmbH 2017. USB Audio driver. https://www.usb-audio.com/.

[44] Simon JD Prince. 2012. *Computer vision: models, learning, and inference*. Cambridge University Press.

[45] Qifan Pu, Sidhant Gupta, Shyam Gollakota, and Shwetak Patel. 2013. Whole-Home Gesture Recognition Using Wireless Signals. In *Proc. of ACM MobiCom*.

[46] PyTorch Team 2019. PyTorch. https://pytorch.org/.

[47] Bhaskar D Rao and KVS Hari. 1989. Performance analysis of root-MUSIC. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37, 12 (1989), 1939–1949.

[48] H Ren. 1996. On the Error Analysis and Implementation of Some Eigenvalue Decomposition and Singular Value Decomposition. (1996).

[49] Sjoerd W Rienstra and Avraham Hirschberg. 2004. An introduction to acoustics. *Eindhoven University of Technology* 18 (2004), 19.

[50] Rohde and Schwarz Group 2018. Automated Measurements of 77 GHz FMCW Radar Signals. https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/ application_notes/1ef88/1EF88_0e_Automated_Measurements_of_77_GHz_FMCW_Radar.pdf.

[51] SparkFun Electronics 2017. ADMP401 MEMS microphones. https://www.sparkfun.com/products/9868.

[52] Petre Stoica and Arye Nehorai. 1989. MUSIC, maximum likelihood, and Cramer-Rao bound. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37, 5 (1989), 720–741.

[53] L. Sun, S. Sen, D. Koutsonikolas, and K. Kim. 2015. WiDraw: Enabling Hands-free Drawing in the Air on Commodity WiFi Devices. In *Proc. of ACM MobiCom.*

[54] Aditya Virmani and Muhammad Shahzad. 2017. Position and Orientation Agnostic Gesture Recognition Using WiFi. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 252–264.

[55] Jue Wang, Deepak Vasisht, and Dina Katabi. 2014. RF-IDraw: Virtual Touch Screen in the Air Using RF Signals. In *Proc. of ACM SIGCOMM.*

[56] Ju Wang, Jie Xiong, Xiaojiang Chen, Hongbo Jiang, Rajesh Krishna Balan, and Dingyi Fang. 2017. TagScan: Simultaneous target imaging and material identification with commodity RFID devices. In *Proc. of ACM MobiCom.* ACM, 288–300.

[57] 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.

[58] Mati Wax, Tie-Jun Shan, and Thomas Kailath. 1984. Spatio-temporal spectral analysis by eigenstructure methods. *IEEE transactions on acoustics, speech, and signal processing* 32, 4 (1984), 817–827.

[59] Teng Wei and Xinyu Zhang. 2015. mTrack: High Precision Passive Tracking Using Millimeter Wave Radios. In *Proc. of ACM MobiCom.*

[60] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 351–360.

[61] Sangki Yun, Yi chao Chen, and Lili Qiu. 2015. Turning a Mobile Device into a Mouse in the Air. In *Proc. of ACM MobiSys.*

[62] Sangki Yun, Yi-Chao Chen, Huihuang Zheng, Lili Qiu, and Wenguang Mao. 2017. Strata: Fine-Grained Acoustic-based Device-Free Tracking. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services.* ACM, 15–28.

[63] Ouyang Zhang and Kannan Srinivasan. 2016. Mudra: User-friendly Fine-grained Gesture Recognition using WiFi Signals. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies.* ACM, 83–96.

[64] 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 *Proc. of ACM MobiSys.*

[65] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. 2018. Through-wall human pose estimation using radio signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 7356–7365.

[66] Yanzi Zhu, Yibo Zhu, Ben Y. Zhao, and Haitao Zheng. 2015. Reusing 60 GHz Radios for Mobile Radar Imaging. In *Proc. of ACM MobiCom.*