UltraGesture: Fine-Grained Gesture Sensing and Recognition

Kang Ling[1] Haipeng Dai[1] Yuntang Liu[1] Alex X. Liu[1,2]
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China[1]
Dept. of Computer Science & Engineering, Michigan State University, East Lansing, MI, 48824, USA[2]
{lingkang,yuntangliu}@smail.nju.edu.cn, haipengdai@nju.edu.cn

***Abstract*—With the rising of AR/VR technology and miniaturization of mobile devices, gesture is becoming an increasingly popular means of interacting with smart devices. Some pioneer ultrasound based human gesture recognition systems have been proposed. They mostly rely on low resolution Doppler Effect, and hence focus on whole hand motion and cannot deal with minor finger motions. In this paper, we present UltraGesture, a Channel Impulse Response (CIR) based ultrasonic finger motion perception and recognition system. CIR measurements can provide with 7 mm resolution, rendering it sufficient for minor finger motion recognition. UltraGesture encapsulates CIR measurements into an image, and builds a Convolutional Neural Network model to classify these images into different categories, which corresponding to distinct gestures. Our system runs on commercial speakers and microphones that already exist on most mobile devices without hardware modification. Our results show that UltraGesture achieves an average accuracy of greater than 97% for 12 gestures including finger click and rotation.**

I. Introduction

User gesture recognition is an important technology in Human Computer Interaction (HCI) area, and gains greater awareness in recent years with the rising of Augmented Reality (AR), Virtual Reality (VR) technologies and Internet of Things (IOT) [1]. Meanwhile, with the miniaturization of mobile devices, such as smart phones, smart watches, *etc.*, the input pattern with fingers on the screen gradually becomes cumbersome. In addition, trying to use smart devices while wearing gloves or with wet hands often leads to annoying user experiences.

Many user gesture recognition systems have been proposed. Traditional approaches use wearable sensors [2, 3], cameras [4, 5], or Radio Frequency (RF) signals [6, 7]. However, wearable sensors based approaches are normally burdensome because of the inconvenience of wearing sensors. Cameras based approaches suffer from bad lighting conditions or occlusion. RF based approaches use either Wi-Fi or specialized devices. Wi-Fi measurement is too coarse grained for recognizing minor gestures with its inherent long wavelength; while specialized devices such as USRP are often not cost effective. Recently some Ultrasound based gesture recognition systems have been proposed, such as Doplink [8], Soundwave [9], and AudioGest [10]. Their advantages over sensors, cameras, and RF based approaches are that they do not require users to wear any devices or good lighting conditions, and provide fine-grained hand gesture recognition.

The key limitation of the pioneer ultrasound based human gesture recognition systems is the lack of high resolution measurements. These systems mostly rely on Doppler Effect in received signals induced by human palm movements. Nevertheless, Doppler Effect based methods fall short in low resolution either in time or frequency domain theoretically in ultrasound band, and therefore, adopting Doppler Effect as features limits the accuracy for recognizing human gestures, especially minor finger gestures.

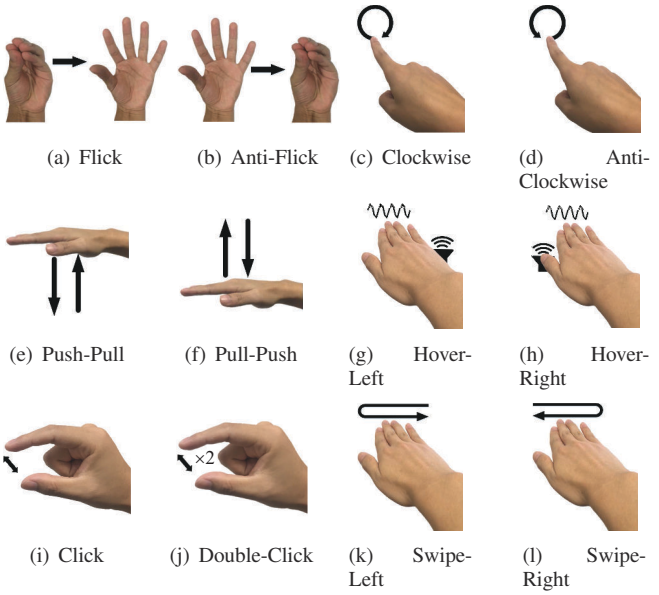
In this paper, we propose UltraGesture, a device-free gesture recognition system which can run on devices embedded with commercial speakers and microphones. Different from ultrasound based gesture recognition systems which rely on Doppler Effect, we leverage Channel Impulse Response (CIR) information for gesture recognition, which outperforms Doppler and FMCW in representing gestures.

We first introduce the principles of CIR measurements, and sketch the mechanisms of UltraGesture. Similar to other wireless channels, such as electromagnetic channels, an ultrasound channel can be viewed as a Linear Time Invariant (LTI) system within coherence time (about 8.5 ms in our case). Ultrasound wave received at microphones is a combination of copies of original signals prorogating along multiple reflect paths. CIR measurements actually indicate the information of prorogation path length and corresponding reflected signal strength. When fingers move around the speaker and microphones, the CIR measurement changes with specific patterns corresponding to the gestures. We capture these patterns as a feature to recognize what gesture the user performs. Next, we describe the recognition process of UltraGesture as follows. First, we send an inaudible known training sequence using a speaker for channel estimation and keep recording at microphones. Then, combining the received signal and the known training sequence, we build a system of linear equations, where the unknowns are CIR measurements. And we solve this system of linear equations with Least Square (LS) estimation to obtain the CIR measurements. Last, we adopt a machine learning approach to build a classifier based on collected training data and then use it for gesture recognition. As the resolution of path length in CIR measurements is up to 7 mm, we can effectively detect and recognize minor finger gestures.

Yet, leveraging CIR measurements requires UltraGesture to overcome the following three challenges:

Changing Multipath: CIR measurements contain not only

978-1-5386-4281-8/18/$31.00 ©2018 IEEE



(a) Flick (b) Anti-Flick (c) Clockwise (d) Anti-Clockwise

(e) Push-Pull (f) Pull-Push (g) Hover-Left (h) Hover-Right

(i) Click (j) Double-Click (k) Swipe-Left (l) Swipe-Right

Fig. 1. Twelve gestures used in this work.

reflected signal propagation path information caused by finger motions but also surrounding static objects layout information. For example, CIR measurements will endure a significant deformation when mobile phones are placed in different places. Hence, CIR measurement changes caused by finger motions may be buried in the large upheaval by displacement of devices. To address this challenge, we propose to use the differential of CIR measurements, denoted as dCIR, to capture the characteristics of gestures. We show that dCIR measurement will not be affected as long as the surrounding object arrangement doesn't change when a gesture is performed,and meanwhile, dCIR measurements are effective features when used to detect a user gesture.

Minor Gestures: It's challenging to classify similar gestures such as clockwise and anti-clockwise rotation because of symmetry. We propose to address this question by adding more microphones as it can provide additional distance perspective. Experimental results show that more microphones can significantly improve classification performance.

Features Extracting: Existing works tend to extract features that include frequency shift, signal amplitude, etc., as classification basis. But such manually chosen features are prone to miss important properties or lack generality. We address this problem by designing a Convolutional Neural Network (CNN) classifier and taking a period of continues CIR measurements as its input. As we will see later, when we consider the CIR measurements for a period of time as an image (data from different microphones can be considered as different image channels), CNN becomes a natural choice.

The key technical novelty of this paper is two-fold. First, to the best of our knowledge, we are the first to propose using CIR measurement as gesture recognition features in ultrasound channels. By leveraging this fine resolution measurement method, we push the ultrasound based gesture recognition into finger motion level. Second, we encapsulate the CIR measurements from multiple microphones into an image and leverage CNN model to build a classifier to recognize gestures with high robustness. We implemented UltraGesture on ultrasound band supported by commercial speakers and microphones without hardware modification and evaluated it in multiple environments and usage scenarios. Our results show that UltraGesture achieves an average accuracy greater than 98% among 12 gestures (See Figure 1), and keeps over 95% accuracy for real scenarios such as users wearing gloves.

II. Related Work

Generally, existing researches related to ultrasound based gesture recognition can be classified into the following four categories: wearable sensor based, camera based, Radio Frequency (RF) based, and ultrasound based.

Wearable Sensor Based: Many research efforts have been devoted to gesture recognition using wearable sensors [2, 3, 11–13]. For example, RF-IDraw [3] allows a user to interact with a desired computing device by gesturing or writing his/her commands in the air by attaching an RFID tag on one of hig/her fingers. Risq [2] detects the smoking gesture by the inertial unit embedded in a wristband. WristWhirl [11] achieves recognizing eight gestures with 93.8% accuracy on a smartwatch. Wearable Devices such as smartwatches [12] and rings [13] can also be used to enable text input. All these methods require users to wear a physical device which is usually burdensome. In contrast, UltraGesture works in a device-free manner, and a user can operate with bare hands *without* wearing any devices.

Camera Based: Prior work has built a firm foundation in vision based gesture recognition [14, 15]. In the past few years, Microsoft Kinect [4] and Leap Motion [5] became popular commercial products which support computer human interaction by body activities. Although having high performance, they suffer from interference from surrounding infrared radiation sources or dim lighting conditions despite of their high equipment costs [16].

RF Based: RF signal is now widely used for presence detection [17, 18], human localization [19–21], and activity recognition [22–24]. In addition to commodity device based systems [17, 18, 22–28] that can only identify large scale human activities, recently some minor gesture recognition systems relying on RF signal processing have been proposed [6, 7, 29, 30]. For example, Fadel Adib *et al.* [30] built Vital-Radio to measure human breathing rate and heartbeat using FMCW signals sent and received by USRP devices. WiFinger [6] shows that two different periodic finger gestures (Circle Left and Zoom In) near the laptop possess different patterns in received CSI signals. WiGest [7] achieves recognizing seven whole-hand gestures with 87.5% accuracy for one AP and 96% accuracy for four APs. However, all the aforementioned gesture recognition systems have their limitations either in robustness or in gesture fineness. For instance, Vital-Radio and WiFinger cannot measure the breath rate while there are other people walking around, and WiGest can only deal with whole-hand gestures rather than finger motions. In conclusion, RF

(especially Wi-Fi frequency band) based gesture recognition can only handle large scale activities because of the long wavelength of RF, and cannot be applied to the scenario considered in this work.

Ultrasound Based: First, several ultrasound based gesture recognition schemes have been proposed, *e.g.*, [8–10, 31–35]. Airlink [33] achieves file transmission between two devices with a bare hand in air gesture. Soundwave [9] leverages Doppler effect to classify five different gestures, such as slow tap and fast tap. AudioGest [10] identifies six whole-hand motions with 95.1% accuracy using Doppler effect too. Rather than using Doppler effect, we measure the channel impulse response in multiple microphones and can recognize more subtle and complex gestures, *i.e.*, fingers' clockwise and anticlockwise rotation. Second, there are many works based on ultrasound tracking in recent years [36–43], some of which works in device-free manner are very relevant to our work. For example, LLAP [36] tracks a finger leveraging the phase changes in received baseband signals induced by finger motion. Strata [39] and FingerIO [37] use chirps to track a finger or a palm with 10 mm and 8 mm 2D tracking error. Although these technologies show great progress in tracking accuracy, the biggest limitation of prior device-free solutions is that they cannot handle multi-targets, especially when these targets are not move in a same direction [36]. Hence, we cannot use these technologies for complex gesture recognition. Unlike these works, our system focuses on 3D gesture recognition which deals with multiple fingers that move at same time, such as flick and click gestures, see Figures 1(a) and 1(i).

III. CIR Measurement

Figure 2 shows a brief overview of our UltraGesture system. First, we send a pre-defined training sequence in ultrasound band with 100 periods per second. In received end, we perform CIR calculation using Least Square (LS) estimation every 10 ms. The second step of UltraGesture is gesture detection. We propose to use Gaussian smooth variance over dCIR feature to achieve robustness gesture detection. After gesture detection, the last step of UltraGesture is recognizing and classifying user input gestures. To distinguish between different gestures, we build a CNN (Convolutional Neural Network) based model as our gesture image classifier.

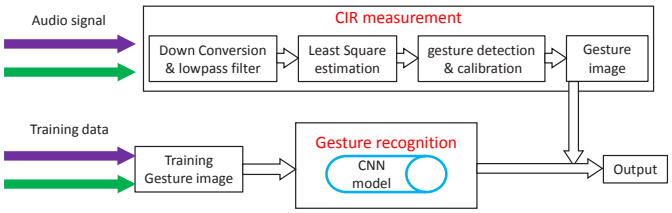


Fig. 2. System overview

A. Limitations of Other 1D Measurement Methods

Before we go into our CIR estimation design, we first explain why other existing 1D measurement methods are not appropriate for the scenario considered in this work.

Basically, existing related works commonly use methods including Doppler Effect, FMCW, and phase measurement. In Doppler shift based 1D measurement, as a moving object induces a frequency shift in the corresponding received signal, which is called Doppler shift, we can calculate the speed of the moving object by measuring the frequency shift using STFT (Short-Time Fourier Transform). For Doppler shift, we have frequency shift $f = \frac{F_c \cdot V}{c}$ where F_c is the center frequency which is 20 kHz in our case, V is object's moving speed, and c is the speed of sound which is 340 m/s. Moreover, we have minimal frequency resolution by STFT $\Delta f = \frac{F_s}{W}$ where F_s is sampling rate which is 48 kHz and W is the STFT window length which is set to 2048. Hence, the minimal speed resolution Δv is given by

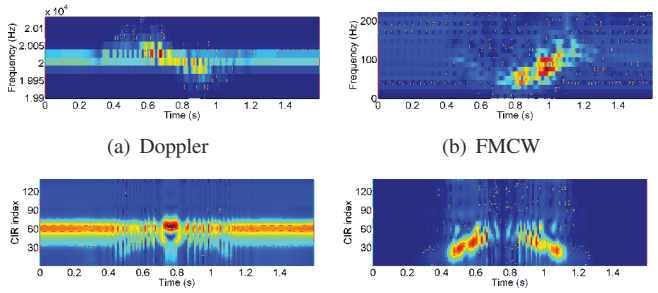
$$\Delta v = \frac{c \cdot |\Delta f|}{F_c} = \frac{F_s \cdot c}{W \cdot F_c}. \tag{1}$$

Substituting the parameters into Equation (1), we have Δv being about 0.398 m/s, which means Doppler shift has a low speed resolution and a user has to move his/her hand/finger very fast to be detected. Note that this resolution cannot be compensated by zero padding the STFT window as no information are added, and the window length W is set to 2048 which corresponds to 40 ms and thus greatly exceeds coherence time (about 8.5 ms) in our scenario.

In FMCW based 1D measurement, a frequency modulated chirp is transmitted, the receiver can calculate the path distance according to the frequency shift. For FMCW, we have minimal distance resolution as

$$\Delta d = \frac{c \cdot |\Delta t|}{2} = \frac{c \cdot |\Delta f|}{2 \cdot \frac{B}{T}} = \frac{c \cdot \frac{F_s}{W}}{2 \cdot \frac{B \cdot F_s}{L}} = \frac{c \cdot L}{2 \cdot B \cdot W}, \tag{2}$$

where B is the FMCW chirp bandwidth, T is the chirp period, and L is the corresponding number of samples in one cycle of the chirp. Typical value for B is 4 kHz considering the narrow inaudible ultrasound band in commercial devices constrained by the sampling rate, W is set to be the same as L. Plugging the parameters into Equation (2), we have minimal distance resolution Δd equal to 4.25 cm and the time resolution of about 43 ms. Note that obviously neither the speed resolution using Doppler effect nor the distance resolution using FMCW ranging is sufficient for finger motions.



(a) Doppler (b) FMCW

(c) CIR (d) dCIR

Fig. 3. Comparing features extracted by Doppler, FMCW, and CIR for a Push-Pull gesture

Figure 3 shows the features extracted by different methods for a common Push-Pull gesture. Figure 3(a) and Figrue

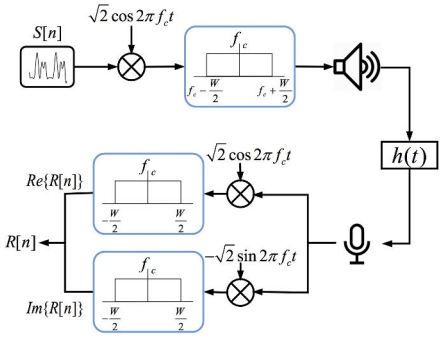


Fig. 4. Up/down conversion

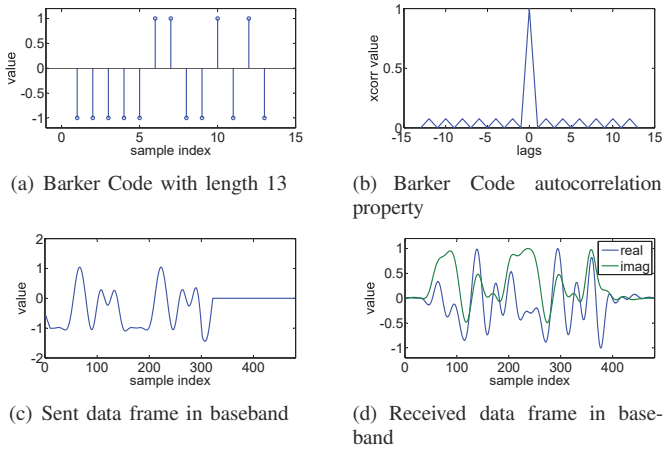
3(b) demonstrate features extracted from Doppler and FMCW methods, respectively, with parameters we just stated. Different from the aforementioned methods, we leverage CIR information for gesture recognition which outperforms Doppler and FMCW in representing gestures. Figure 3(c) and its differential version Figure 3(d) shows features extracted from CIR methods for same gesture. One can tell the performance difference by comparing the resolutions (grid sizes in y-axis) between CIR and aforementioned methods.

Besides, some phase based measurement methods have been proposed in the last two years related to Ultrasound tracking [36], and phase based 1D measurement is shown to have high distance resolution. Nevertheless, when multiple objects move at same time, the phase information at the receive signal will entangled together and cannot be decoupled. Therefore, these methods cannot be applied to our considered case.

B. CIR Measurement

In this section, we present our approach to measuring the one dimensional CIR information. Wireless communication theory shows that wireless channel state could be approximately viewed as constant within coherence time, and meanwhile the wireless channel could be viewed as a Linear Time Invariant (LTI) system between transmitted and received signal [44]. We measure the CIR of this LTI system by sending an ultrasound training sequence. The transmit and receive system for our CIR measurement is illustrated in Figure 4. Our CIR estimation method aims to measure the Impulse Response of this LTI system, which is $h(t)$ in the figure.

1) Training Sequence Design: Typical training sequence frame includes two parts: data section and empty section. Data section is designed for training usage, where signal received at receiver can be viewed as a linear combination of data section signal, and empty section (with all zeros) is designed to avoid signal confusion from last frame. We use Barker Code as our training data basis. A Barker Code or Barker Sequence is a finite sequence of +1 and -1 with the ideal autocorrelation properties. Barker Codes with length N equal to 11 and 13 are used in direct-sequence spread spectrum (DSSS) and pulse compression radar systems because of their autocorrelation properties (other sequences such as complementary sequences which have similar property work well too, we just choose Barker Code in this paper). Figure 5(a) and 5(b) show the Barker Sequence with length 13 and its autocorrelation property. Then we modulate these signals in a narrow ultrasonic band of 18 kHz $\sim$ 22 kHz (human normal audible scope is 20 Hz $\sim$ 18 kHz). To achieve enough length and avoid frequency leakage, we copy it twice and up interpolate the sequence by 12 times. Our system measures CIR information at a frequency of 100 Hz, which means the length of a chirp is 480 samples. The length of data section d in our training sequence is set to 312, and the remaining 168 samples are left for empty section. Note that the length 168 of empty section is enough for empty gapping, that is, a frame received at microphones either from current frame or signal reflected from path length 1.2 m away from last frame sent by speaker. Meanwhile, with length 312, the data section occupies 6.5 ms in time domain, which is less than the coherence time of 8.5 ms when typical speed of finger motion is below $0.5m/s$ (moving speed has an inverse proportional relationship with coherence time). The ultimate training sequence in baseband is shown in Figure 5(c), which corresponds to $S[n]$ in Figure 4.



(a) Barker Code with length 13

(b) Barker Code autocorrelation property

(c) Sent data frame in baseband

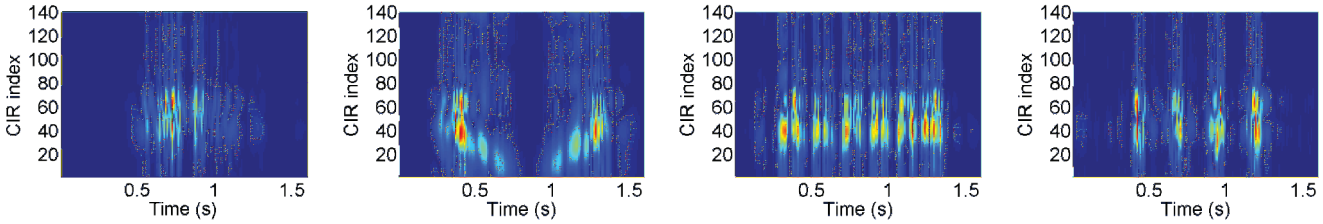
(d) Received data frame in baseband

Fig. 5. Barker Code and its autocorrelation property

2) Transmitter and Receiver Design: At transmitter end, we first multiply the training sequence with $\sqrt{2}\cos 2\pi f_c t$ to convert it into ultrasound frequency band. Then, a passband filter is designed to remove out-of-band noise. Finally, we send it with a speaker. Reversely, at receiver end, we first perform down conversion to received signal by multiplying it with $\sqrt{2}\cos 2\pi f_c t$ and $-\sqrt{2}\sin 2\pi f_c t$, respectively. Then, we pass the baseband signal through a low-pass filter to remove noises such as music and ordinary talking. An example of received data frame in baseband ($R[n]$ in Figure 4) is shown in Figure 5(d), note that $R[n]$ is a complex vector.

3) CIR Estimation: According to wireless communication theory, signal at baseband undergoes the same wireless channels with passband. Hence, we have $R[n] = S[n] * h[n]$, where $*$ means convolution, $S[n]$ and $R[n]$ are signals sent and received at baseband, and $h[n]$ is channel impulse response we aim to derive. To resolve $h[n]$ from $S[n]$ and $R[n]$, we use Least Square (LS) estimation [45] which only involves matrix calculation and thus has low computation cost.

In LS estimation, L is the most important argument which indicates the Channel impulse response length in discrete



(a) Clockwise Rotation (b) Pull-Push (c) Hover (d) Double-Click

Fig. 6. Different gesture samples shown in dCIR measurement

representation, P is always chosen to make $P + L = d$ where d is the length of data section in training sequence. In our system, we set L to 140 as 140 samples approximately correspond to 1 m when the sound speed is 340 m/s . Let $D = \{s_1, s_2, ..., s_d\}$ denote the data section of the training sequence, then the training matrix M is the first $P * L$ blocks of the circulant matrix generated by vector D. The matrix representation of LS estimation is shown in Equation (3).

$$\underbrace{\begin{pmatrix} s_1 & s_2 & \cdots & s_L \\ s_2 & s_3 & \cdots & s_{L+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_P & s_{P+1} & \cdots & s_{P+L-1} \end{pmatrix}}_{TrainingMatrix} \underbrace{\begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_L \end{pmatrix}}_{CIR} = \underbrace{\begin{pmatrix} r_{L+1} \\ r_{L+2} \\ \vdots \\ r_{L+P} \end{pmatrix}}_{R[n]} \quad (3)$$

Resolving such a matrix equation requires $O(L^3 + LP^2)$ complexity. In consideration of that the training matrix M is known in priori, we can use

$$\hat{h} = (M^T M)^{-1} MR \quad (4)$$

to convert the question into a matrix-vector multiplication problem by pre-computing $(M^T M)^{-1} M$ with slightly round-off error induced by computing matrix inverse.

The LS estimation outputs an 140 valued complex CIR for every received baseband data frame. Because we have a frame rate of 100, we can get a $[140 * 100]$ complex CIR matrix every second. Figure 3(c) shows an image of Push-Pull gesture example with time span of about 1.6 seconds, where each column in the image corresponds an 1D CIR measurement, and the brightness represents CIR magnitudes. We can recognize the disturbance caused by user gestures between $0.4 - 1.2$ s.

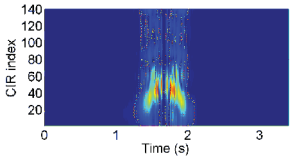


Fig. 7. dCIR measurement image

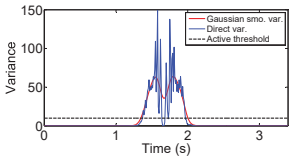


Fig. 8. Gaussian smoothing variance

C. Realtime Detection and Calibration

We design two steps for gesture detecting. In the first step, we perform 1st-order difference upon CIR (dCIR) value to detect CIR changes. As we have shown in Figure 3(c), CIR measurement disturbance indicates user gestures, so detecting gestures can be carried out by subtracting CIR measurement in the current frame from that in the last frame and analyzing the magnitude of the obtained result. After a differential operation, an obvious bright spot can be found in a 2D dCIR image when a gesture occurs, see Figure 7.

The reason that we choose dCIR rather than CIR measurement as our CNN model input is mainly two-fold. First, gesture data is highlighted in dCIR image, while inconspicuous in CIR image. We can easily get this conclusion by comparing Figure 3(c) with Figure 3(d). Second, CIR measurement images indeed contain information from many other surrounding objects. Arrangement of these objects in training and test stages may have significant difference, which will inevitably reduce recognition performance. While for dCIR images, gesture data will not be affected by surrounding objects as long as it keeps static during the gesture process.

In the second step, we use Gaussian smoothed variance to calculate activeness at each frame. We calculate the variance of dCIR for each frame, then conduct convolution operation with a Gaussian smoothing filter. We set the length and the standard variance of the Gaussian filter to 40 and 5, respectively, based on empirical values. The advantage using Gaussian smoothed variance rather than raw variance is to avoid recognition failures stemming from instantaneous/short-term pause or tiny movements during the process of a gesture, *e.g.*, instantaneous pause when a user performs Double-Click or Pull-Push gesture. We then use a uniform threshold to determine whether a time period is active or silent. Figure 8 illustrates the difference between Gaussian smoothed variance and raw variance. With Gaussian smoothed variance, we can detect a gesture with high robustness. Figure 6 shows the gesture data examples in dCIR images.

In realtime scenario, problems arise when a user performs two gestures in a short time period. For example, it is hard to distinguish two Click gestures with a small empty gap (say, 0.5 s) between a Double-Click gesture. To address this issue, we assume that the pause in a gesture is less than 0.4 s, and thus two active windows separated longer than 0.4 s in time will be identified as two gestures. Moreover, to avoid ambiguity, after detecting a gesture, we normally align the gesture data in the middle of the dCIR image before feeding it into the CNN classifier. Another problem we may confront is clock drifting. The impact of clock drifting in CIR estimation is several samples of drifting in overall CIR values, which will not cause big problem for scenarios with a single microphone; in contrast, relation between multi-microphone CIR data will be

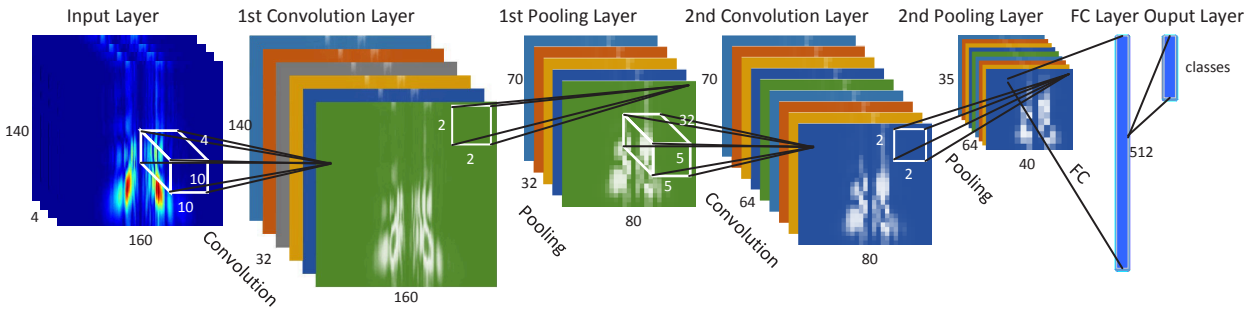


Fig. 9. Convolution neural network architecture

distorted if these microphone data are not clock synchronized. We handle this issue from the fact that distances between the speaker and different microphones are different but constant values. Constant distance means fixed transmission time and hence same CIR ridge index. We leverage this information as a criterion to calibrate all microphone CIR data at the silent stage regularly when no gesture is performed.

IV. Deep Neural Network Model

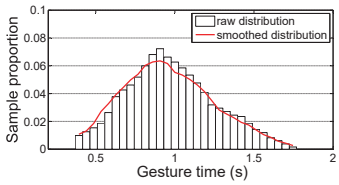


Fig. 10. Gesture time distribution

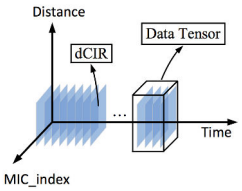


Fig. 11. dCIR tensor

UltraGesture adopts Convolution Neural Network (CNN) for robust gesture recognition. Existing activity and gesture recognition systems apply machine learning algorithms such as K-Nearest Neighbor (KNN) [23] and Hidden Markov Model (HMM) [22]. In our usage case, the input data is a 3D tensor spreading from three distinct dimensions (space, time, and microphone index) as shown in Figure 11 and can be viewed as an image. Traditional machine learning algorithms either treat an image as raw material for feature extraction or directly view every pixel in the image as a feature, which may miss valuable features or destroy valuable relationship between neighboring pixels. In contrast, CNN is suitable for classifying high dimension tensors with preferable performance in image classification area. Next, we present the details of the CNN model used in UltraGesture.

1) CNN Primer: In addition to Input and Output Layer, a CNN architecture normally includes three main types of layers: Convolution Layer (Conv), Pooling Layer (Pool), and Fully-Connected Layer (FC).

Convolution Layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region that they are connected to in the input volume.

Generally, an elementwise RELU Layer will follow after a Convolution Layer. Such as the $\max(0, x)$ we used in our CNN model, this layer will neither change the volume of the previous layer nor introduce extra parameters.

Pooling Layer will perform a down sampling operation along the spatial dimensions (in our case are time and CIR index dimensions), resulting in volume smaller than previous layer. Typical Pooling Layer size are set to [2*2] or [3*3].

FC Layer will compute the class scores for classification. Each neuron in this layer will be connected to all the numbers in the previous layer.

2) CNN Architecture Used in UltraGesture: Figure 9 shows a typical architecture used in CNN model, a short representation of this model is $Input \rightarrow [Conv \rightarrow Relu \rightarrow Pool] * layer\_no \rightarrow [FC] \rightarrow Output$ (note that $layer\_no$ denotes the number of Convolution-Pooling Layers, we set $layer\_no$ to 2 in Figure 9 for convenience of illustration. We set $layer\_no = 5$ in practice). Note that the RELU Layers are ignored in the figure for better illustration as it does not induce extra parameters.

In its input layer, CNN model takes a fixed scale dCIR tensor/image as input. We set the image scale as $[160{*}140{*}mics]$, where 160 is the length in time domain (this value is chosen according to the gesture time distribution, see Figure 10), 140 is the space domain CIR distribution we have mentioned in Section III-B, and $mics$ represents the microphone numbers. Technically, in our CNN based recognition model, data from different microphones are viewed as different color channel, *i.e.*, RGB values in an image.

In a Convolution Layer, the basic operation is 2D convolution. For example, in the first Convolution Layer of our model, the input data is convoluted with the parameter $W1$ with size $[10 * 10 * mics * 32]$ in the first two dimensions. The output size of the first Convolution Layer is $[160 * 140 * 32]$ as we choose the 2D-convolution operation that returns the central part of the convolution of the same size as input in the first two dimensions.

Operation after a Convolution Layer is a Pooling Layer, we use $[2 * 2]$ max pooling in our implementation. As we can see from Figure 9, the size of the first pooling layer is $[80 * 70 * 32]$. The following Convolution and Pooling Layers are similar to the first layer. In practice, we adopt a model with 5 Convolution-Pooling Layers. The 2D (first two dimensions) convolutional filter sizes in the 1st, 2nd, 3rd, 4th, and 5th layer are $[10 * 10]$, $[5 * 5]$, $[5 * 5]$, $[5 * 5]$, and $[3 * 3]$, respectively, considering the gradually reduced image size.

We adopt a fully connection layer at the end of Convolution and Pooling Layer with size 512. At last, a softmax output layer yields the output.

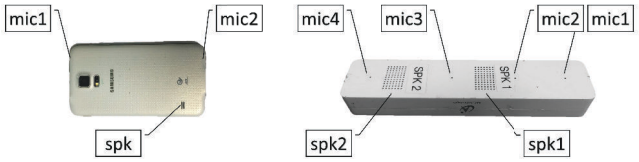
3) CNN Memory Consumption: Deep learning based methods typically have high performance along with huge numbers of parameters. For example, the famous VGGNET [46] contain about 140 M parameters. Nevertheless, our CNN model is constrained in Convolution Layer depth (5) and FC Layer size (512), which significantly reduces the model memory consumption. Our UltraGesture recognition model takes about 2.48 M parameters, which is comparable to some existing deep learning based systems running on mobile devices, such as DeepEar [47] with 2.70 M parameters and SCNN [48] with 2.54 M parameters. Note that the memory consumption can be further reduced with smaller FC Layer size, we will examine this in section V.

4) Updating CNN Model: We envision the UltraGesture system can become an open ultrasound based gesture recognition platform. We can train the model at cloud servers and download the model to smart phones or specific devices for gesture recognition. To improve the generalization for supporting more gesture types and improve robustness for dynamic and harsh environments, UltraGesture retrains the CNN model if new labeled gesture types or samples are added. For newly added gesture samples for existing gesture types, we will first examine the similarity of existing samples, and add it to our gesture library if large enough difference is observed.

V. Experimental Results

A. Experimental Setup

We implement our system on a Sumsung S5 smartphone and a speaker-microphone kit designed by us, which is equipped with 2 commercial speakers (1 used) and 4 commercial microphones (to evaluate system performance along with microphone number).



(a) Sumsung S5 (b) Self-designed kit

Fig. 12. Hardware used in the experiments

We collect data for 12 basic gestures performed by 10 volunteers (8 males and 2 female) with a time span of over a month. All volunteers are asked to perform each 12 gestures 100 times, some of them are asked to repeat more times in different scenarios, including different noise level, left hand, with gloves on, etc. The deep learning model is trained in a PC equipped with 16 GB memory, Intel i7-3770K CPU @3.5 GHz and Nvidia GTX 1080 Graphics Card.

In the realtime recognition stage, the microphone data collected and forwarded to the PC for processing and recognition. For the CNN model, we use 5 Convolution-Pooling Layers and 512 nodes in FC Layer if no otherwise stated.

B. Recognition Accuracy

1) Recognition Accuracy Analysis: The analyzed gesture data is obtained under normal office environment with sound from air conditioners and servers on Sumsung S5. Figure 13 shows the ten-fold cross validation confusion matrix for 12 gestures. The average recognition accuracy is 97.92%, and large error occurs to gesture pairs Swipe Left/Right and Hover Left/right which are both position sensitive. The main reason for these misclassification is due to the poor placement of speakers and microphones on the mobile phone. We can see later that there is no such tendency on self-designed kit, see Figure 20, where microphones and speakers face in the same direction.

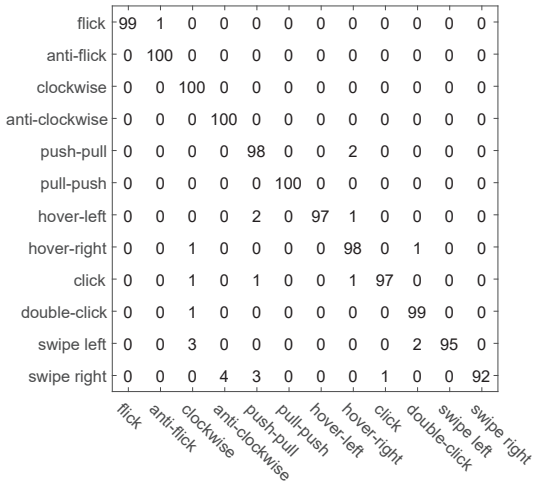
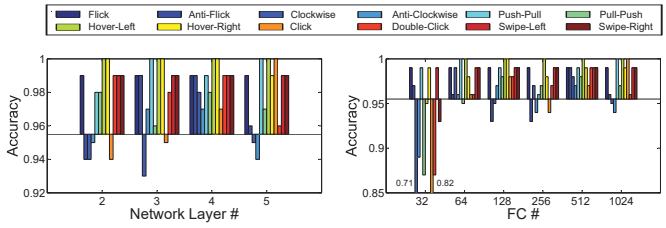


Fig. 13. Confusion matrix for mobile phone



(a) Accuracy vs. CNN layer number (b) Accuracy vs. FC number

Fig. 14. Recognition accuracy vs. CNN model Convolution Layer number and FC Layer node numbers

2) Impact of CNN parameters: We examine the impact of CNN architecture in terms of two parameters, *i.e.*, Convolution Layer number and FC Layer node number. Figure 14(a) shows the recognition accuracy with different layer numbers. Note that the accuracies for 12 different gestures are represented in different colors as shown by the legend of Figure 14, and we use such representations in the remaining figures but omit the corresponding legends for better illustration. The overall accuracies across all test samples are 97.42%, 97.92%, 98.67%, and 97.83% for layer depth 2, 3, 4, and 5, respectively. Figure 14(b) shows the recognition accuracy with different FC Layer sizes varying from 32 to 1024 (note that exponent of 2 is preferred in CNN architecture to improve computational efficiency). We can observe the performance enhancement as the FC Layer size increases. When FC Layer size is set to 32, the overall accuracy is 91.42% with lowest gesture accuracy being 71% for Clockwise gesture. As a comparison to FC Layer size 512, the overall accuracy is 98.58%.

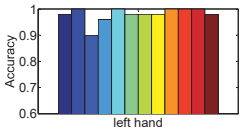


Fig. 15. Left hand

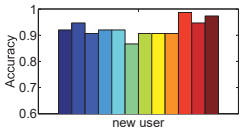


Fig. 16. New users

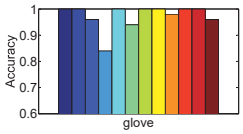


Fig. 17. With glove on

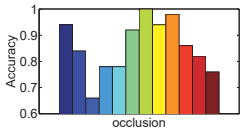


Fig. 18. With occlusion

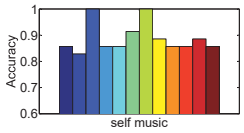


Fig. 19. With self playing music

3) Impact of Mic 一个麦克风正确率为92.75% se the kit in Figure 12(b) to study the stem performance and the number of microphones. The overall accuracies for one to four microphone(s) scenarios are 92.75%, 95.00%, 96.83%, and 98.58%, respectively. The performance drop is more conspicuous for minor gestures with lesser microphones. For example, Clockwise and Anti-Clockwise gestures are hard to distinguish based on data from one single microphone, as their distance change patterns are quite similar to each other.

4) Impact of Noise Level: Figure 21 shows the recognition accuracies under different ambient noise levels. We simulate different environment noises at three levels, 55dB, 60dB, and 65dB by playing music near the device at different volumes. Note that the standard dataset are collected under the laboratory environment with noise level of about $45 \sim 55$dB. We can observe that, with the increase of noise level, recognition accuracy slightly declines. The overall gesture recognition accuracy drops slightly from 98.33% to 93.67% when the noise level increases from 55dB to 65dB.

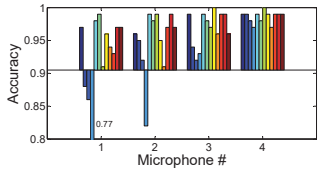


Fig. 20. Recognition accuracy vs. microphone number

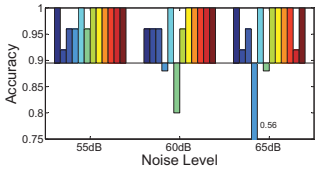


Fig. 21. Recognition accuracy vs. noise level

5) Impact of Usage Scenarios: Besides noise, we instigate our system performance under several possible and important usage scenarios, including: left hand, new user, with a glove on, with occlusion and device playing music itself.

- *Left hand (98.00%):* Although most of our gestures are performed with the right hand, result shows that recognition accuracy with the left hand is generally not effected by what hand a user uses (classifier retrained with 100 left hand samples for each gesture in training set). Figure 15 shows the recognition results for 12 gestures, 50 samples for each gesture, the overall accuracy is 98.00% and almost all gestures have recognition accuracies higher than 95%.
- *New users (92.56%):* New users denote the users who haven't contributed gesture samples to our system in CNN model training stage. We use one volunteer's data as the test dataset and others as the test set alternately in this scenario. The recognition results are shown in Figure 16. We can see that the mean accuracy across all gesture is 92.56%. The accuracy drops are mainly due to the lack of diversity in gesture samples. This can be improved by adding more training samples.
- *With a glove on (97.33%):* An annoying scenario often happens when we try to control our mobile phone with wet hands or a glove on. We test the system performance with a user performing all the gestures while wearing a 5-fingered glove. Beyond our expectations, with 600 (50 for each gestures) training samples in the training set, UltraGesture achieves 97.33% mean accuracy, detailed accuracies for every gestures are shown in Figure 17.
- *Occlusion (85.67%):* Sometimes, we attempt to control our device even if it is occluded by other things, for example, in the pocket. We simulate this scene by occluding microphones and speaker with a sheet of A4 paper. Figure 18 shows the recognition accuracies for all gestures under this scenario. The overall accuracy is 85.67%, which may not satisfactory but still acceptable compared with vision based gesture recognition systems.
- *Using UltraGesture while playing music (88.81%):* A typical usage scenario of UltraGesture is using it to control a music player or video player. We add a music sound (about 55dB detected from 20 cm distance and can be heard from 5 m away) onto the transmitted signal after the up conversion operation to simulate this case. Although ordinary music and video sound have different frequency bands with UltraGesture, our gesture recognition method while playing these sounds endures a significant performance drop. Different with external noise, self sound playing induces about 10 percents accuracy fall, the overall accuracy across all gesture is 88.81%. We conjecture the reason comes from the minor vibration caused by self sound playing.

C. System Performance

1) Training Time: Our CNN model training process takes about 15 minutes for a training set of 10,800 samples. In our experiments, we find that the test accuracy of the CNN model generally converged after traversing 30 rounds of training data. Therefore, it is easy to re-train the model in the server end when new gestures are added.

TABLE I
TIME COST FOR EACH STAGE

Operation	Call period	Computational time (4 mics)
down_convert	every 10 ms	0.106 ms
lowpass filter	every 10 ms	0.481 ms
LS estimation	every 10 ms	0.495 ms
Classification	$>$1.6 s	28.6 ms

2) Response Time: For a dCIR data of size $[160*140*4]$, it takes about 28.6 ms for UltraGesture to classify a gesture. Note that a delay lower than 30 ms is often unnoticeable for users. The time cost for each stage of data process and classification stage is shown in Table I.

3) Memory Consumption: Table II shows the memory consumption of our CNN model under different parameters (4 microphones). Considering the memory consumption and recognition accuracy (see Section V-B), we choose $layer\_no = 5$

TABLE II
MEMORY CONSUMPTION UNDER DIFFERENT CNN PARAMETERS

fc_no (layer_no=5)	32	64	128	256	512
Parameters	1.74 M	1.79 M	1.89 M	2.08 M	2.48 M
Mean Accuracy	91.42%	97.75%	97.92%	96.92%	98.58%

and $fc\_no = 512$ as final CNN architecture parameters, which contains 2.48 M parameters.

4) Operation Distance: UltraGesture is designed to respond to a gesture within 20 cm, when the operation distance is beyond 25 cm, we observe a quick drop in the detection and recognition accuracy. For example, in a 50-time click experiment at a distance of 30 cm, the motion can only be successfully detected with 36% rate and the classification accuracy is 74% (we feed the data into classifier manually).

5) Power Consumption: UltraGesture's power consumption is also acceptable. In our experiment, a fully charged Samsung S5 mobile phone can continuously transmit and record ultrasound for about 7 hours. Power consumption can be further reduced by setting an idle state in which the system runs at a low frame rate until an activation gesture is detected. We leave this as a future work.

VI. CONCLUSIONS

In this paper, we make the following three contributions. First, we proposed to use CIR measurements as gesture recognition measurements, which is more reliable and achieves higher resolution than most Doppler based methods. Benefiting from high resolution in CIR measurement, our proposed system UltraGesture can effectively sense finer motions such as finger click and rotations. Second, we proposed to convert dCIR measurements of gestures from multiple microphones into an image and further use CNN to classify the obtained image into different categories corresponding to different gestures with high robustness. Third, we conducted extensive experiments in several different environments and usage scenarios. Our results show that UltraGesture achieves an average accuracy greater than 97% for 12 gestures.

ACKNOWLEDGEMENT

This work is partially supported by the National Natural Science Foundation of China under Grant 61502229, 61672276, 61472184 and 61321491, and the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program.

REFERENCES

[1] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, and S. He, "Narrowband internet of things: Implementations and applications," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2309–2314, 2017.
[2] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis, "Risq: Recognizing smoking gestures with inertial sensors on a wristband," in *ACM MobiSys*, 2014.
[3] J. Wang, D. Vasisht, and D. Katabi, "RF-IDraw: virtual touch screen in the air using rf signals," in *ACM SIGCOMM*, 2014.
[4] "Microsoft kinect," https://www.microsoft.com/en-us/research/publication/microsoft-kinect-sensor-and-its-effect/, 2010.
[5] "Leap motion," https://www.leapmotion.com/, 2012.
[6] S. Tan and J. Yang, "Wifinger: leveraging commodity WiFi for fine-grained finger gesture recognition," in *ACM Mobihoc*, 2016.
[7] H. Abdelnasser, M. Youssef, and K. A. Harras, "WiGest: A ubiquitous wifi-based gesture recognition system," in *IEEE INFOCOM*, 2015.
[8] M. T. I. Aumi, S. Gupta, M. Goel, E. Larson, and S. Patel, "Doplink: Using the doppler effect for multi-device interaction," in *ACM UbiComp*, 2013.
[9] S. Gupta, D. Morris, S. Patel, and D. Tan, "Soundwave: using the doppler effect to sense gestures," in *ACM CHI*, 2012.
[10] W. Ruan, Q. Z. Sheng, L. Yang, T. Gu, P. Xu, and L. Shangguan, "Audiogest: enabling fine-grained hand gesture detection by decoding echo signal," in *ACM UbiComp*, 2016.
[11] J. Gong, X.-D. Yang, and P. Irani, "Wristwhirl: One-handed continuous smartwatch input using wrist gestures," in *ACM UIST*, 2016.
[12] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch," in *ACM HotMobile*, 2015.
[13] S. Nirjon, J. Gummeson, D. Gelb, and K.-H. Kim, "Typingring: A wearable ring platform for text input," in *ACM MobiSys*, 2015.
[14] T. Starner and A. Pentland, "Real-time american sign language recognition from video using hidden markov models," in *ISCV*, 1997.
[15] J. M. Rehg and T. Kanade, "Visual tracking of high dof articulated structures: an application to human hand tracking," in *ECCV*, 1994.
[16] D. Cronin, "Usability of micro-vs. macro-gestures in camera-based gesture interaction," Ph.D. dissertation, Diplomarbeit, California Polytechnic State University, San Luis Obispo, 2013., 2014.
[17] A. E. Kosba, A. Saeed, and M. Youssef, "Rasid: A robust wlan device-free passive motion detection system," in *IEEE PerCom*, 2012.
[18] M. Youssef, M. Mah, and A. Agrawala, "Challenges: device-free passive localization for wireless environments," in *ACM MobiCom*, 2007.
[19] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller, "3D tracking via body radio reflections," in *Usenix NSDI*, 2013.
[20] F. Adib, Z. Kabelac, and D. Katabi, "Multi-person motion tracking via RF body reflections," in *Usenix NSDI*, 2015.
[21] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You are facing the mona lisa: spot localization using phy layer information," in *ACM MobiSys*, 2012.
[22] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of WiFi signal based human activity recognition," in *ACM MobiCom*, 2015.
[23] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, "E-eyes: In-home device-free activity identification using fine-grained WiFi signatures," in *ACM MobiCom*, 2014.
[24] G. Wang, Y. Zou, Z. Zhou, K. Wu, and L. M. Ni, "We can hear you with Wi-Fi!" in *ACM MobiCom*, 2014.
[25] F. Adib and D. Katabi, "See through walls with WiFi!" in *ACM SIGCOMM*, 2013.
[26] L. Sun, S. Sen, D. Koutsonikolas, and K.-H. Kim, "WiDraw: Enabling hands-free drawing in the air on commodity wifi devices," in *ACM MobiCom*, 2015.
[27] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home gesture recognition using wireless signals," in *ACM MobiCom*, 2013.
[28] B. Kellogg, V. Talla, and S. Gollakota, "Bringing gesture recognition to all devices," in *NSDI*, 2014.
[29] J. Liu, Y. Wang, Y. Chen, J. Yang, X. Chen, and J. Cheng, "Tracking vital signs during sleep leveraging off-the-shelf wifi," in *ACM MobiHoc*, 2015.
[30] F. Adib, H. Mao, Z. Kabelac, D. Katabi, and R. C. Miller, "Smart homes that monitor breathing and heart rate," in *ACM CHI*, 2015.
[31] B. Kreczmer, "Gestures recognition by using ultrasonic range-finders," in *MMAR*, 2011.
[32] M. Babaei, M. Al-Jemeli, and I. Avazpour, "Ugi: a multi-dimensional ultrasonic-based interaction approach," in *ACM CHI*, 2016.
[33] K.-Y. Chen, D. Ashbrook, M. Goel, S.-H. Lee, and S. Patel, "Airlink: sharing files between multiple devices using in-air gestures," in *ACM UbiComp*, 2014.
[34] Y.-t. Liu, C.-y. Wu, P.-h. Shih, H.-s. Liang, and M. Y. Chen, "SoundSense : 3D Gesture Sensing using Ultrasound on Mobile Devices," 2013.
[35] C. R. Pittman and J. J. LaViola Jr, "Multiwave: Complex hand gesture recognition using the doppler effect," in *ACM GI*, 2017.
[36] W. Wang, A. X. Liu, and K. Sun, "Device-free gesture tracking using acoustic signals," in *ACM MobiCom*, 2016.
[37] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "FingerIO: Using active sonar for fine-grained finger tracking," in *ACM CHI*, 2016.
[38] C. Huijie, L. Fan, and W. Yu, "EchoTrack: Acoustic Device-free Hand Tracking on Smart Phones," in *InfoCom*, 2017.
[39] Y. SangKi, C. Yi-Chao, Z. Huihuang, Q. Lili, and M. Wengguang, "Strata: Fine-grained acoustic-based device-free tracking," in *MobiSys*, 2017.
[40] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: a high accuracy acoustic ranging system using cots mobile devices," in *ACM SenSys*, 2007.
[41] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin, "Detecting driver phone use leveraging car speakers," in *ACM MobiCom*, 2011.
[42] S. Yun, Y.-C. Chen, and L. Qiu, "Turning a mobile device into a mouse in the air," in *ACM MobiSys*, 2015.
[43] [illegible] : Accurate Infrastructure-Free Indoor Location [illegible] *M MobiCom*, 2015.
[44] [illegible] *fundamentals of wireless communication*. Cambridge university press, 2005.
[45] M. Pukkila, "Channel estimation modeling," *Nokia Research Center*, 2000.
[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[47] N. D. Lane, P. Georgiev, and L. Qendro, "Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *ACM UbiComp*, 2015.
[48] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *IEEE CVPR*, 2015.

LS信道估算