mmKey: Universal Virtual Keyboard using A Single Millimeter Wave Radio

Yuqian Hu, *Student Member, IEEE,* Beibei Wang, *Senior Member, IEEE,* Chenshu Wu, *Senior Member, IEEE,* and K. J. Ray Liu, *Fellow, IEEE*

***Abstract*—Keyboard acts as one of the most commonly used mediums for human-computer interaction. Today, massive Internet of Things (IoT) devices are designed without a physical keyboard as they go tiny, but are almost all equipped with a wireless module for networks. In this work, we aim to enable a universal virtual keyboard using wireless signals, which would allow a typing interface for tiny IoT devices or serve as a portable alternative to the unwieldy physical keyboards. To this end, we present *mmKey*, the first universal virtual keyboard system using a single millimeter-wave (mmWave) radio. By leveraging the unique advantages of mmWave signals, *mmKey* converts any flat surface, with a printed paper keyboard, into an effective typing medium. *mmKey* enables concurrent keystrokes and supports multiple keyboard layouts (e.g., computer keyboard, piano keyboard, or phone keypad). We design a novel signal processing pipeline to detect, segment and separate, and finally recognize keystrokes. *mmKey* does not need any training except for a minimal one-time effort of only three key-presses for keyboard calibration upon the initial setup. We prototype *mmKey* using a commodity 802.11ad/ay chipset, customized to support radar-like operations, and evaluate it with different keyboard layouts under various settings. Experimental results with 10 participants demonstrate a keystroke recognition accuracy of $> 95\%$ for single-key case and $> 90\%$ for multi-key scenario, which leads to a word recognition accuracy of $> 97\%$.**

***Index Terms*—Virtual keyboard, wireless sensing, millimeter-wave radio**

I. Introduction

Keyboard, as the primary and most integrated computer peripheral, has become an indispensable part of our daily lives. However, the physical keyboards have been suffering from the poor portability issue. Additionally, as IoT devices go smaller, they are typically not allowed to have a bulky physical keyboard. Therefore, virtual keyboards have been greatly demanded as handy substitutes for the ordinary physical keyboards or to enable typing experience for billions of IoT devices without a keyboard.

Virtual keyboards can be implemented either actively or passively. Active approaches require users to be equipped with wearable devices such as potentiometer, pressure sensor, vibration sensor and micro inertial measurement unit (mIMU) [1]–[4]. Differently, in passive systems, there are no sensors attached to users such as vision-based approaches [5]–[8] that utilize vision technologies based on outputs from cameras, laser or infrared, acoustic-based [9]–[11], and electromagnetic emission-based approaches [12]–[14]. However, the vision-based approaches raise concerns about privacy invasion and are sensitive to the lighting condition, while the acoustic-based approaches suffer from false alarms due to the ambient interference.

Y. Hu, B. Wang, C. Wu and K. J. R. Liu are with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD, 20742, and Origin Wireless, Inc., Greenbelt, MD, 20770
E-mail: {yhu1109, bebewang, cswu, kjrliu}@umd.edu.

While not all computing devices have a camera or a speaker, almost all of them contain one or more wireless modules. Therefore, RF-based approaches become particularly attractive to enable ubiquitous virtual keyboard systems. The widely deployed WiFi infrastructure has inspired extensive research on passive WiFi sensing [15], [16] and many applications have spawned, including keystroke recognition [17]–[19], mainly using 2.4GHz/5GHz WiFi radios. However, the capabilities of these approaches are fundamentally limited by the narrow bandwidth, large wavelength and limited number of antennas. For example, limited by the 20MHz/40MHz bandwidth, the range resolution of the above systems can be several meters. Therefore, the reflected signals from all targets and the background environment are superimposed together and hard to separate. To achieve a reasonable recognition accuracy, the existing approaches resort to data-driven learning but require cumbersome training [17]–[19]. Moreover, they cannot recognize multiple concurrent keystrokes due to the mixed signals and are usually trained for a single fixed keyboard layout.

Recently, mmWave radio, which has emerged as the next-generation wireless communication technique, has become available on commodity networking devices [20], [21]. mmWave radio breaks down the above limitations of conventional WiFi by offering finer range resolution by large bandwidth, highly directional signals with large phased array, and short wavelength on high-frequency band. In this paper, we leverage this opportunity and present *mmKey*, the first passive virtual keyboard system using a single mmWave radio. Without any extra hardware, *mmKey* can transform any flat surface such as a printed paper or a painted area into an interactive typing medium. Compared with conventional approaches, *mmKey* enables distinct features of concurrent keystrokes support and user-defined keyboard layouts. *mmKey* achieves all these features in a universal virtual keyboard system by capturing the mmWave signals reflected off moving fingers and employing a novel pipeline of signal processing, without requiring any training. Consequently, *mmKey* is environment- and location-independent, can work anywhere, and can easily adapt to different keyboards, such as computer keyboards, piano keyboards, phone keypads, or other user-customized layouts, with zero cost.

mmKey overcomes multiple challenges to deliver a practical system on commodity mmWave radio. First, before keystroke recognition is possible, it is critical to design a robust motion detector that can capture the micro motions on the keyboard. To address this challenge, *mmKey* applies an anomaly detection on the differential amplitude of the channel impulse response (CIR) to sense the signal fluctuations and infer the presence of motions. Due to the high carrier frequency, signals attenuate rapidly over the propagation distance, and therefore the thresholds for keystroke detection should adapt to the distances. We achieve an adaptive z-score detector by referring to the empty CIR measured in absence of targets. We further leverage multiple antennas and different ranges to improve the robustness.

Second, since keystrokes involve not only the movements of fingers but also the shifts of palms and potentially arms, it is difficult to distinguish between the keystrokes (finger motions) and other motions. In addition, there are also irrelevant reflections from the background objects, which are mixed together with the keystroke motions. To overcome this challenge, we first devise a novel motion filter by leveraging the sensitivity of CIR phase as well as the differences in the spatial distribution of dynamic signals between keystrokes and other types of motions. Then we further employ adaptive background cancellation to extract only the dynamic reflections by tracking the CIR changes.

Last but most importantly, despite the many antennas of the mmWave device, the spatial resolution is physically limited due to the small effective aperture of the receive antenna array. Specifically, the on-chip analog beamforming provides an angular resolution of 15° on our experimental device with an array size of 1.8 cm $\times$ 1.8 cm, which is inadequate to localize and recognize a keystroke, especially when the key size is very small or simultaneous keystrokes are close to each other. To boost the spatial resolution, *mmKey* performs MUltiple Signal Classification (MUSIC) algorithm on the received CIR and enables precise localization of the keystroke. In addition, by initial finger localization we only know the location of motions relative to the device. To determine the keys pressed by a user, we employ a low-effort one-time calibration stage upon initial setup, which involves as simple as three key-presses, so that the estimated locations by MUSIC can be mapped onto the corresponding keys of the keyboard.

We prototype *mmKey* on commodity 60GHz 802.11ad/ay networking chipset sponsored by Qualcomm, which is attached with an additional array to enable radar-like operations and report CIR. We validate the performance of *mmKey* by extensive experiments on three different virtual keyboards, including a computer keyboard, a piano keyboard, and a phone keypad. We conduct experiments at different locations in both home and office environments, with ten volunteers involved. Experimental results demonstrate a remarkable accuracy of $> 95\%$ for single-keystroke scenario and $> 90\%$ for multiple concurrent keystrokes. Furthermore, by feeding *mmKey*'s output to commercial text correction tools, we achieve a considerable word recognition accuracy of $> 97\%$ for natural typing on a printed computer keyboard. With the great performance, *mmKey* promises universal virtual keyboards for computers, mobiles, wearables, and IoT devices, should they be equipped with a mmWave radio.

In summary, the main contributions are as follows:

- We design *mmKey*, the first virtual keyboard system using a single mmWave radio. With minimal infrastructure support, *mmKey* can turn any flat surface, with a printed paper keyboard, into an effective interactive tool.
- We present a novel signal processing pipeline to detect, segment and recognize both single keystroke and multi-finger concurrent keystrokes without any training.
- We prototype *mmKey* by reusing a commodity 60GHz WiFi radio as a mmWave radar and evaluate its performance on various types of virtual keyboards through extensive experiments.

The rest of this paper is organized as follows. Section II reviews the related works. III presents an overview of *mmKey* and preliminaries about mmWave radio, followed by the motion detection and distinction in Section IV and keystroke localization in Section V. Section VI details keyboard calibration and keystroke recognition. Performance is evaluated in Section VII. We discuss limitations and future directions in Section VIII and conclude the paper in Section IX.

II. Related Work

Virtual keyboard system. As a portable alternative to physical keyboards, various virtual keyboard systems based on different modalities have been proposed as summarized in Table I. The existing approaches can be divided into two categories: active and passive virtual keyboards.

The active virtual keyboard systems require users to wear specialized devices to track the motion of fingers. For instance, [2] requires the user to wear a glove with attached sensors to track the motions of fingers, while [3] integrates the sensors in the wristband. [4] embeds the mIMU into the ring and utilizes the extracted angle and acceleration features for recognition. However, it is cumbersome for users to wear specialized sensors and more false alarms will occur due to the high sensitivity, which inspires the development of passive virtual keyboards.

The most common passive virtual keyboard systems are vision-based. [5] and [6] employ cameras to detect and localize the keystroke by shape-based fingertip tracking, while [7] utilizes Real-Adaboost for depth estimation from images of the user's hands to support keystroke recognition. However, camera-based systems are limited by their privacy invasion and the requirement for ambient light. Besides traditional RGB cameras, optics-based sensors such as LiDAR and depth sensors in Kinect have also been integrated for virtual input [8], [26], [27]. However, LiDAR is too expensive for home use and lacks strict international protocols that guide data collection and processing. Also, optics-based sensors usually have smaller wavelengths (800 $\sim$ 1550 nm) compared with the wavelength (5 mm) of mmWave radio and thus can only work in a line-of-sight (LOS) scenarios, while mmWave radio is able to penetrate thin layers of different kinds of materials [28] and track the motions in a non-intrusive way [29], [30].

Acoustic- and ambient-based sensing have also been considered to enable passive virtual keyboard implementation.

TABLE I: Summary of related works on virtual keyboard systems/keystroke recognition

Modality	Reference	Method	Hardware	Training	Limitation
Vision	Murase *et al.* [7]	Contour extraction, Real-Adaboost	Camera	Yes	Requirement of good visibility Specialized devices Privacy issue
	Ji *et al.* [6]	Contour feature restriction	RGB camera	No	
	CamK [5]	Shape-based fingertip tracking	Camera	Yes	
	Su *et al.* [8]	Morphology processing, ellipse fitting	Image sensor	No	
Acoustic	Zhuang *et al.* [9]	MFCC, HMM, linear classification	Microphone	Yes	High sensitivity to ambient sounds Applicable for only single-key keystroke Specialized devices
	Zhu *et al.* [10]	Time difference of arrival (TDoA)	Smart phone	No	
	UbiK [11]	Multipath fading of audio signals	Smart phone	Yes	
Ambient	Marquardt *et al.* [22]	Accelerometer, FFT, MFCC	Smart phone	Yes	Applicable for only single-key keystroke Specialized devices
	VibKeyboard [23]	Power spectral density, SVM	Vibration sensor	Yes	
Wearable	Zhao *et al.* [4]	Angle complementary filter, kNN	mIMU	Yes	Inconvenience to users High false alarm rate Specialized devices
	Wu *et al.* [2]	Velocity and acceleration measurement	Pressure sensor	Yes	
	iKey [3]	MFCC, class-center classification	Vibration sensor	Yes	
	Scherer *et al.* [13]	LDA, feedback training	EEG sensor	Yes	
RF-based	WiKey [18]	PCA, DWT, DTW, kNN	2×3 transceivers	Yes	Re-training required in new environments Applicable for only single-key keystroke Non-portable devices
	Windtalker [19]	DWT, PCA, STFT, DTW	Directional antenna	Yes	
	Chen *et al.* [17]	FIR filter, phase/amplitude matching	1×2 transceivers	Yes	
	SpiderMon [24]	PCA, Wavelet decomposition, SVM, HMM	Directional antenna	Yes	
	WiPass [25]	Symlet filter, DCASW, DTW	1×2 transceivers	Yes	

[9]–[11] classify the acoustic signals when typing different keys for keystroke recognition while [22], [23] use either the accelerometer in mobile phones or vibration sensor to capture and decode the vibrations from nearby keystrokes. Nevertheless, the sensitivity to ambient sounds or vibrations prevents these approaches from being widely deployed in practical applications.

By analyzing the variations of the channel state information (CSI) due to the influence of human activities on ubiquitous WiFi signals, WiFi sensing has gained popularity in recent years [15], [16]. [17]–[19] show the potential of using the 2.4GHz/5GHz wireless radios to distinguish different keystrokes. More specifically, by analyzing the unique patterns of CSI when pressing different keys, [18] explores the feasibility of using 2.4GHz WiFi radios for keystroke recognition. [17] localizes the keystrokes by matching and canceling the signal amplitude/phase over different antennas. Extracting features in both time and frequency domains, Windtalker in [19] utilizes network layer traffic information and physical layer CSI information to recognize keystrokes. Limited by the fundamental characteristics of 2.4GHz/5GHz signals, however, all of them can only work for a single keystroke, are vulnerable to surrounding motion interference, and require significant effort for training and learning, preventing them from easily generalizing to new keyboards or new environments.

Wireless sensing with WiFi and mmWave. WiFi signals have also been extensively explored for various applications beyond virtual keyboards, such as gesture/handwriting recognition [31]–[35], sign language recognition [36], [37], motion detection [38], breathing and sleep monitoring [39], [40]. Recently, 60GHz WiFi technology and commodity mmWave indoor radars have emerged and shown distinct advantages for wireless sensing. Researchers have exploited the potential of mmWave radios for accurate indoor tracking [41] and gesture recognition [42]. Besides, mmWave signals also enable other fine-grained applications including imaging [43], gait recognition [44], speech sensing [45], hand-writing [46] and even target material sensing [47]. In this work, we reveal the possibility of using a single 60GHz WiFi radio for virtual keyboards. To the best of our knowledge, *mmKey* is the first virtual keyboard using a mmWave radio.

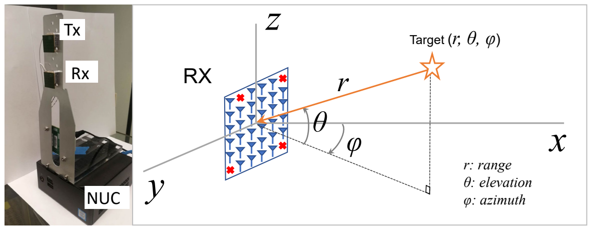


Fig. 1: Device and coordinate system. The antenna array contains 32 elements in a 6×6 layout, with 4 reserved locations marked by red crosses.

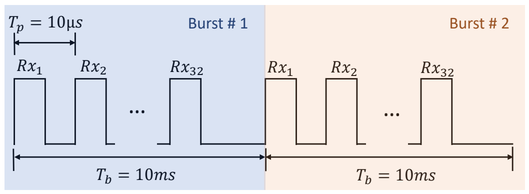


Fig. 2: Frame structure in mmWave radio.

III. Overview

A. CIR on 60GHz Radio

As shown in Fig. 1, *mmKey* is built upon a mmWave testbed provided by Qualcomm, which enables a radar mode on a commodity 802.11ad chipset by attaching an additional antenna array to the radio. The device operates at 60GHz frequency band with a bandwidth of 3.52GHz. The transmitter (Tx) and receiver (Rx) arrays both have 32 antennas assembled in a 6 $\times$ 6 layout. To extract CIR, each Tx antenna transmits a burst

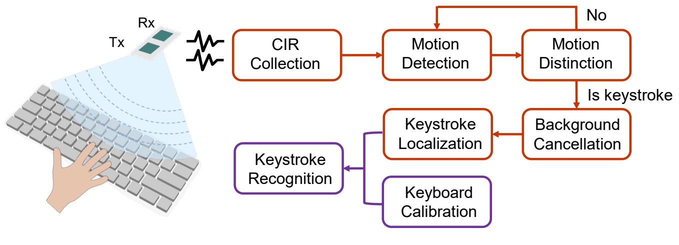


Fig. 3: An overview of *mmKey*.

consisting of a group of 32 pulses, which are then received by 32 Rx antennas sequentially, and the corresponding CIR is recorded. In *mmKey*, we use 1 Tx antenna and 32 Rx antennas and it takes T_b to finish one period of transmission as Fig. 2 shows. The duration of each pulse is $T_p = 10\ \mu s$, and the duration of each burst is $T_b = 10$ ms. Each pulse can be reflected by reflectors at different ranges producing a difference in time-of-arrival (ToA). A single impulse tap of CIR corresponds to the smallest ToA that can be separated. Specifically, the bandwidth of 3.52GHz on our experimental device gives us a time resolution of 0.28 ns, which means two reflection paths with a delay difference larger than 0.28 ns can be distinguished. Therefore the range of reflectors could be measured by CIR taps with a range resolution of 4.26 cm. The CIR reported by the n-th antenna at time slot t can be expressed as

$$h_n(t) = \sum_{l=0}^{L-1} h_{n,l}(t-\tau_l) = \sum_{l=0}^{L-1} g_{n,l}(t)\delta(t-\tau_l), \quad (1)$$

where L is the number of range taps. $\delta(\cdot)$ is the Delta function which represents the unit impulse. $g_{n,l}$ and τ_l are complex channel gain and propagation delay of the l-th range tap, respectively.

The transmitted pulse signals get reflected by the surrounding objects and finally received by Rx as CIR. Denote the number of antennas as N. For each time slot t, the captured CIR is an $N \times L$ complex matrix. By analyzing the received signals, we can monitor the activities around including keystrokes and hand movements. We use a coordinate system as illustrated in Fig. 1, where the reflected signals impinge on the receive antenna array with different azimuths φ and elevations θ.

B. mmKey Overview

The main challenge for *mmKey* is to promptly and robustly recognize the keystrokes from the RF signals reflected from not only the fingers but also the hands and other static objects. As illustrated by Fig. 3, *mmKey* addresses the challenge by the following procedures: 1) *Motion detection* that detects the presence of motions adaptively and robustly; 2) *Motion distinction* that distinguishes keystrokes by fingers from non-interested motions caused by hands, arms, and others; 3) *Adaptive background cancellation* that extracts dynamic reflections from the mixture of the superimposed reflected signals; and 4) *Keystroke localization* that localizes the keystrokes with high resolution. Note that a one-time calibration is used for key-location mapping upon initial setup, yet the effort is minimized as only three key presses.

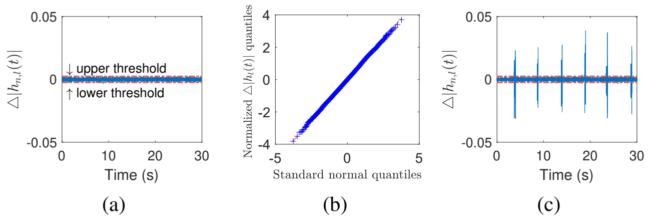


Fig. 4: (a) Differential CIR amplitude of stationary reference frame; (b) Q-Q plot of differential CIR amplitude of reference frame; (c) Differential CIR amplitude of frame involving keystrokes.

IV. Keystroke Detection and Distinction

A. Motion Detection

We capture the real-time motion by observing the fluctuations of the CIR amplitude $|h_{n,l}(t)|$. The CIR amplitude can be modeled as

$$|h_{n,l}(t)| = |h_{n,l}(t-1)| + a^d_{n,l}(t) + a^{noise}_{n,l}(t), \quad (2)$$

where $a^d_{n,l}(t)$ reflects the variation of amplitude contribution from dynamic signals and $a^{noise}_{n,l}(t)$ is due to measurement noise. Therefore, the differential CIR amplitude can be calculated as

$$\triangle|h_{n,l}(t)| = |h_{n,l}(t)| - |h_{n,l}(t-1)| = a^d_{n,l}(t) + a^{noise}_{n,l}(t). \quad (3)$$

When there is no motion, *i.e.*, $a^d_{n,l}(t) = 0$ in equation (2) and we have $\triangle|h_{n,l}(t)| = a^{noise}_{n,l}(t)$ in equation (3). Without loss of generality, the amplitude change caused by measurement noise $a^{noise}_{n,l}(t)$ only can be assumed to follow a Gaussian distribution. Then by collecting a sequence of $\triangle|h_{n,l}(t)|$ in "no motion" scenario, we are able to construct a "stationary" frame denoted as $\triangle|\mathbf{h}_{ref,n,l}|$ and employ the z-score anomaly detection method to detect motion in real time by comparing the incoming differential CIR amplitudes $\triangle|h_{n,l}(t)|$ with $\triangle|\mathbf{h}_{ref,n,l}|$.

More specifically, by centering and normalizing $\triangle|h_{n,l}(t)|$ with the sample mean and sample standard deviation of $\triangle|\mathbf{h}_{ref,n,l}|$, we evaluate the z-score of $\triangle|h_{n,l}(t)|$ as

$$Z_{n,l}(t) = \frac{|\triangle|h_{n,l}(t)| - \hat{\mu}_{ref,n,l}|}{\hat{s}_{ref,n,l}}, \quad (4)$$

where $\hat{\mu}_{ref}$ and $\hat{s}_{ref}$ are the sample mean and standard deviation of $\triangle|\mathbf{h}_{ref,n,l}|$. The larger the value of $Z_{n,l}(t)$ is, the more the sample diverges from the reference frame, and the higher chance a motion occurs at time t.

Fig. 4a presents an instance of the reference frame $\triangle|\mathbf{h}_{ref,n,l}|$. z-score based anomaly detection assumes the reference sample sequence follows a Gaussian distribution. Thus, we examine the quantile-quantile (Q-Q) plot of the normalized samples in $\triangle|\mathbf{h}_{ref,n,l}|$, as shown in Fig. 4b. As seen, the distribution of normalized $\triangle|\mathbf{h}_{ref,n,l}|$ is very close to a normal distribution and satisfies the requirements of z-score computation. Fig. 4c shows $\triangle|h_{n,l}(t)|$ including six keystroke motions. Every time there is a keystroke, $\triangle|h_{n,l}(t)|$ experiences obvious fluctuations, which can be captured by evaluating $\triangle|h_{n,l}(t)|$ with a threshold calculated by equation (4) as indicated by the red dotted line.

Motion detection aims to detect the start and end time of a keystroke and its corresponding range. Instead of relying on the z-score calculated from one single antenna, we leverage all available antennas and range taps to improve the robustness. Specifically, we apply a sliding window, with length W, to the incoming CIR stream and obtain the CIR for each window as a $N \times L \times W$ complex-valued matrix. To reduce the false alarms, we employ majority voting to the corresponding $\triangle|h_{n,l}|$ values and construct an indicator matrix $I(t)$ with dimension $N \times L \times W$, in which each element $I_{n,l}(t) = \mathbb{1}\{Z_{n,l}(t) > \upsilon\}$ where $\mathbb{1}$ is the indicator function and $\upsilon = 3$ is the commonly used value for z-score anomaly detection. Then motion is detected if the majority of the elements of $I(t)$ are ones. We further estimate the range tap of motion as the one that satisfies $\hat{l} = \arg\max_{l} \sum_{n=1}^{N} \sum_{t=t_0}^{t_0+W} Z_{n,l}(t)$. The start and end points of motion can be determined by searching the first and last anomaly time slot over consecutive windows on the $\hat{l}$-th tap.

B. Keystroke Distinction

Although the motion detector can identify which range tap is affected by motion, it cannot distinguish whether the motion is caused by a keystroke or by hand movements. Our key idea to distinguish keystrokes from hand motions is inspired by two observations: 1) Hand movements usually involve shifts of hand location while finger keystrokes do not, and 2) hand movements impact a much larger reflection area than finger motions. Accordingly, we devise two features for motion distinction: *CIR phase* and *dynamic level*.

1) Raw CIR phase: Compared with CIR amplitude, CIR phase is more sensitive to tiny location shifts of reflectors. Note that the CIR phase is already synchronized between all antennas and all samples. With the carrier frequency operated at 60.48GHz, the wavelength is $\lambda = \frac{c}{f} = 5$ mm, meaning that a tiny shift of the reflector of 2.5 mm towards/away from the radio in the radial direction will produce a change of 2π in the CIR phase, underpinning accurate classification of large (e.g., hand) and micro (e.g., fingertips) motions.

Fig. 5 shows the CIR amplitude differential $\triangle|h_{n,l}(t)|$ as well as the CIR phase $\angle h_{n,l}(t)$ from a sequence of CIR involving three palm movements indicated by the red rectangles, each followed by a single finger keystroke indicated by the green rectangle. As Fig. 5a shows, based on the evident fluctuations of $\triangle|h_{n,l}(t)|$, all the six motions can be detected. However, from $\triangle|h_{n,l}(t)|$ it is hard to tell whether the motion is a finger keystroke or not, which could be more distinguishable by measuring $\angle h_{n,l}(t)$. As shown in Fig. 5b, hand motions produce much higher peaks due to larger location changes than finger keystrokes. Therefore, the peak height acts as a promising feature for distinguishing these two motions. As Fig. 5c illustrates, we define the peak height as the average of the heights on both sides of a peak.

Since hand shifts impact more antennas and may cross multiple taps, we integrate the CIR phase $\angle h_{n,l}(t)$ over all antennas and three neighbour taps (corresponding to a range of about 13 cm) centered at the target tap, i.e., $[\hat{l}-1, \hat{l}, \hat{l}+1]$.

2) Dynamic level: Observing that hand shifts also impact a larger reflection area than finger keystroke, we develop a novel feature of dynamic level to describe such a difference. Dynamic level is defined as the ratio of non-DC power to the total power of the CIR. Denoted as γ, it can be calculated by

$$\gamma = \frac{\sum_{l=\hat{l}-1}^{\hat{l}+1} \sum_{n=1}^{N} \sum_{f=1}^{K} |H_{l,n}(f)|^2}{\sum_{l=\hat{l}-1}^{\hat{l}+1} \sum_{n=1}^{N} \sum_{f=0}^{K} |H_{l,n}(f)|^2}, \tag{5}$$

where $H_{l,n}(f) = FFT(h_{l,n}(t))$. The denominator is the total power of signals reflected from both static background and dynamic hands/fingers, while the numerator is the power reflected only by the moving objects (with the DC components excluded). Therefore, dynamic level increases as the size of the reflection area increases. In other words, hand movements should yield higher dynamic levels than finger motions. Fig. 6 shows the distributions of the dynamic levels for one-finger keystroke, two-finger keystroke, three-finger keystroke and hand shift, respectively. As illustrated, three types of keystrokes share similar dynamic levels while hand motions experience much larger values, rendering it an effective metric to distinguish hand and finger motions.

Combining two features together, we distinct the motions with a simple two-step verification. More specifically, once the motion is detected and segmented, the CIR frames are evaluated by thresholding both the peak height of raw CIR phase and dynamic level, and only the motions with both low peak heights and small dynamic levels are considered as finger keystrokes. In our experiments, this conservative decision rule can perfectly filter out interference motions by hands with empirical preset thresholds, but may also cause miss detection of finger keystrokes, which is measured by detection accuracy and evaluated in Section VII.

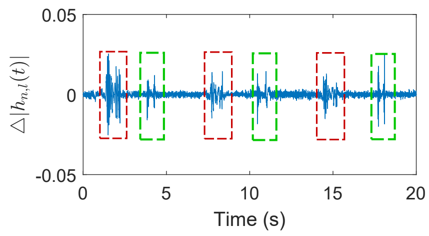
V. Keystroke Localization

A. Adaptive Background Cancellation

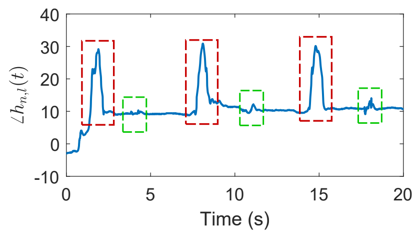
As Fig. 7a shows, the received signals are a mixture of the reflections from all the dynamic and static objects. Hence, we need to eliminate the background reflections and extract only the dynamic components associated with keystrokes.

For each time slot t, the CIR $h_{n,l}(t)$ can be expressed as the combination of the CIR $h_{n,l}(t-1)$ and their differential. From $t-1$ to t, the reflections from the static background are embedded in $h_{n,l}(t-1)$, while the change of CIR consists of the components due to the new dynamic reflection $h^d_{n,l}(t)$ and that due to noise $\varepsilon_{n,l}(t)$. Therefore, we can cancel the impact of background reflections by subtracting the term $h_{n,l}(t-1)$. Assuming $h^d_{n,l}(t)$ does not experience significant change for M successive samples, as illustrated in Fig. 7b, $h^d_{n,l}(t)$ can be estimated as

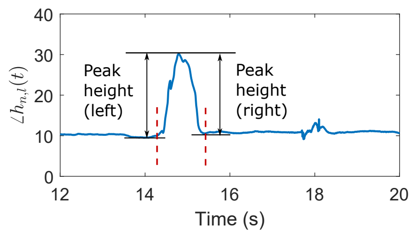
$$\hat{h}^d_{n,l}(t) = h_{n,l}(t) - \frac{1}{M} \sum_{k=1}^{M} h_{n,l}(t-k), \tag{6}$$



(a) Amplitude.



(b) Raw phase.



(c) Peak height of CIR raw phase.

Fig. 5: Features for motion distinction.

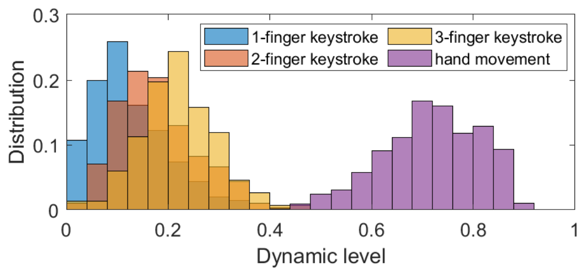
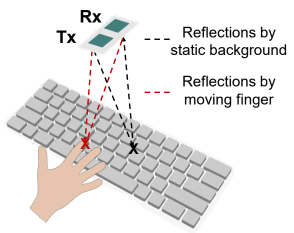
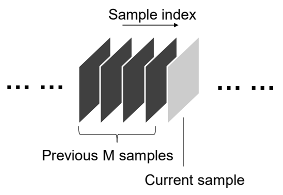


Fig. 6: Dynamic level distribution.


Rx
Tx
Reflections by static background
Reflections by moving finger

(a) Mixed signals from object and background.



(b) Example of adaptive background cancellation.

Fig. 7: Adaptive background cancellation.

where M denotes the number of samples used for background cancellation.

B. *Keystroke Localization*

1) Localization with MUSIC: After extracting the dynamic signals contributed by finger keystrokes, we would like to get the 3-D coordinates of the keystroke locations, which will then translate into actual keys as detailed in the next section. The spatial resolution is greatly limited by the small effective aperture of the receive antenna array. To enhance the spatial resolution and thus accurately localize the keystrokes, *mmKey* performs digital beamforming on the received CIR based on the widely adopted MUSIC algorithm [48]. The basic idea of the MUSIC algorithm is to perform an eigen-decomposition for the covariance matrix of CIR, resulting in a signal subspace orthogonal to a noise subspace. MUSIC is typically used for reconstructing the spatial spectrum of sparse signals, which is in line with the goal of localizing less than 10 keystrokes.

We focus on the targeted $\hat{l}$-th range tap estimated in the previous modules. Assume that there are D reflected signals impinging on the receive antenna array with different azimuths φ and elevations θ in the coordinate system shown in Fig. 1. Then, the CIR $\mathbf{h}$ can be formulated as

$$\mathbf{h} = \begin{bmatrix}\mathbf{s}(\theta_1,\varphi_1),\cdots,\mathbf{s}(\theta_D,\varphi_D)\end{bmatrix}\begin{bmatrix}x_1\\ \vdots \\ x_D\end{bmatrix} + \begin{bmatrix}\epsilon_1\\ \vdots \\ \epsilon_N\end{bmatrix}, \quad (7)$$

where $\mathbf{s}(\theta_i,\varphi_i)$ is the steering vector pointing to (θ_i,φ_i), corresponding to the direction of the i-th reflected signal, *i.e.*, the normalized phase response of the antenna array for a signal coming from the direction (θ_i,φ_i). x_i denotes the complex value of the i-th reflected signal and ϵ_j stands for additive thermal noise by j-th antenna, which is assumed to be a Gaussian random variable with zero mean and independent and identically distributed (I.I.D.) for different receive antennas. A more concise matrix representation of equation (7) can be written accordingly as $\mathbf{h} = S\mathbf{x} + \boldsymbol{\epsilon}$, where S is defined as the steering matrix. Then, the covariance of $\mathbf{h}$ can be evaluated as

$$R_h = \mathbb{E}[\hat{\mathbf{h}}\hat{\mathbf{h}}^H] = S\mathbb{E}[\hat{\mathbf{x}}\hat{\mathbf{x}}^H]S^H + \mathbb{E}[\epsilon\epsilon^H] = R_s + R_\epsilon, \quad (8)$$

where $\hat{\mathbf{h}} = \mathbf{h} - \mathbb{E}[\mathbf{h}]$, and R_s and R_ϵ are the covariance matrices of the signal and noise components, respectively. Then, the eigen-decomposition can be represented as

$$R_h = \begin{bmatrix}U_s & U_\epsilon\end{bmatrix}\begin{bmatrix}\Lambda_s & \\ & \Lambda_\epsilon\end{bmatrix}\begin{bmatrix}U_s\\ U_\epsilon\end{bmatrix}, \quad (9)$$

where U_s is signal space while U_ϵ is noise space. The MUSIC spatial spectrum is expressed as

$$P(\theta,\varphi) = \begin{bmatrix}\mathbf{s}^H(\theta,\varphi)U_\epsilon U_\epsilon^H\mathbf{s}(\theta,\varphi)\end{bmatrix}^{-1}. \quad (10)$$

Fig. 9a shows the pseudo-spectrum of a single keystroke motion. Peaks of the spatial spectrum P indicate the presence of reflected signals due to finger keystrokes, while low values of P indicate the absence of such reflections. Later we will also evaluate other spectrum estimation methods in Section VII, which shows MUSIC performs the best.

2) Location Refinement: Although MUSIC algorithm can achieve a high resolution in localizing the sources of motion, it requires a-priori knowledge of the number of sources, which is usually unknown in practice. To handle this problem, we apply a peak selection module before the target localization. A preset number of targets K is fed into the MUSIC algorithm to obtain the initial pseudo-spectrum. K peaks will be extracted from the pseudo-spectrum regardless of the actual number of

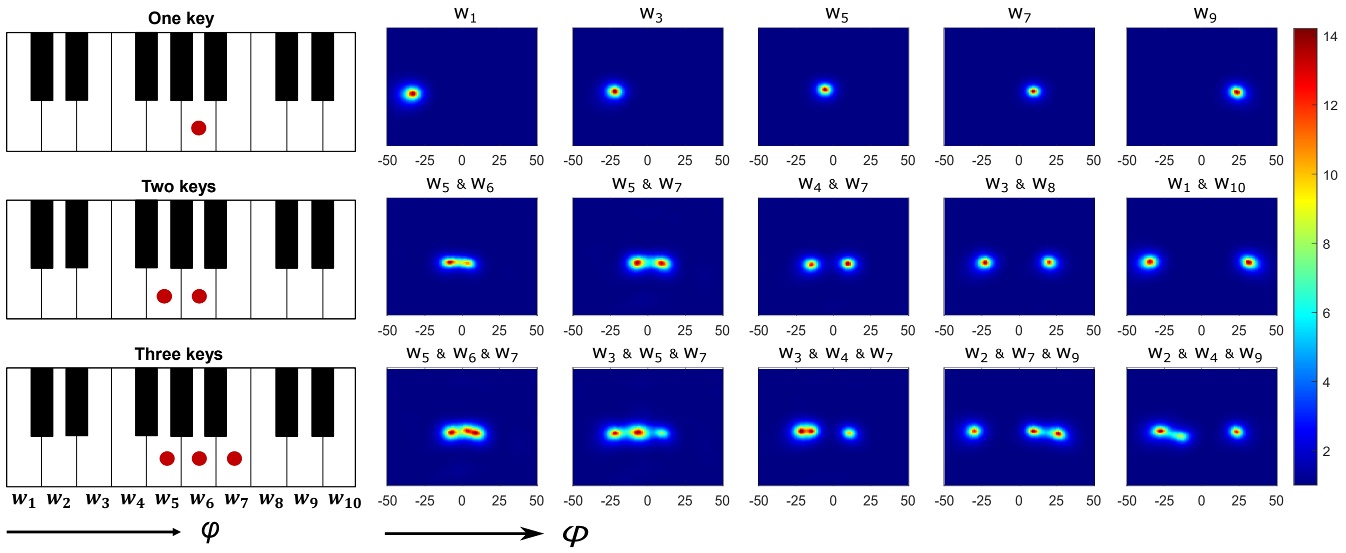
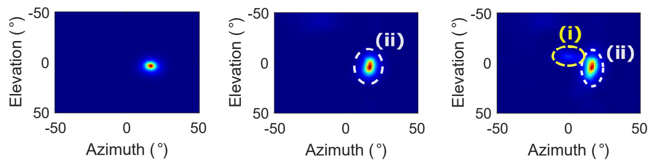


Fig. 8: Illustration of one-key, two-key and three-key keystrokes and spatial spectrum.



(a) K = 1. (b) K = 3. (c) K = 5.

Fig. 9: The MUSIC spectrum for $K = 1$, 3, 5, respectively.

targets present. A peak selection module is then designed to remove the false peaks.

Fig. 9 shows the pseudo-spectrum of a single keystroke motion with different preset K. As K increases, more and more outlier peaks present, including (i) *lower peaks in the background* (marked in yellow circle (i)) and (ii) *higher peaks diffused from the target peak* (marked in white circle (ii)).

To remove the false peaks and determine the number of true targets, we follow two criteria:

- The peaks with heights lower than a preset adaptive threshold th_1 are considered as the noise peaks and will be filtered out. To be generic, th_1 is a proportional function of the height of the highest peak, *i.e.*, $th_1 = c \cdot max(p_1, ..., p_K)$, which is determined in the calibration phase.
- For the neighboring diffused peaks, an agglomerative hierarchical clustering is applied to merge them if the spatial angle distance between these peaks is within a threshold d_{th}. Observing that the peaks tend to expand more in the elevation direction as illustrated in Fig. 9bc due to the signals reflected by the upper parts of fingers, we adopt a relatively smaller weight in the elevation direction. More specifically, the distance between two peaks $(\triangle\theta, \triangle\varphi)$ is weighted with (a, b) respectively where $a < b$ to tolerate more expansion of the peak blurring in θ. d_{th} is an adaptive threshold indicating the size of clusters and is determined by the maximum spatial distance of diffused peaks in the calibration phase.

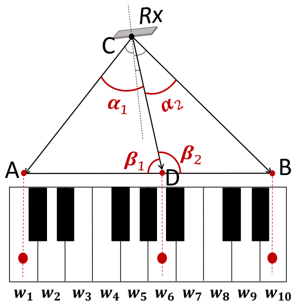
After filtering and clustering the detected peaks, the number of keystrokes is then estimated as the number of clusters, and the highest peak in each cluster is considered as the representative of the cluster, whose estimated location denoted as $(\hat{\theta}, \hat{\varphi})$ will be fed in the keystroke recognition module described in the next section.

VI. Keystroke Recognition

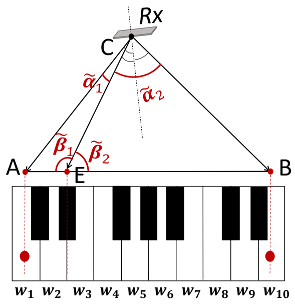
The location of the finger keystroke $(\hat{\theta}, \hat{\varphi})$ estimated by the super-resolution MUSIC algorithm can only reflect the relative position of the keystroke with respect to the Rx. To map the keystroke location onto the keyboard and infer which key is pressed, we need the knowledge of the location of the keyboard relative to the Rx, with which a keystroke at location $(\hat{\theta}, \hat{\varphi})$ can be translated to a specific key. We employ a simply calibration step to obtain such mapping relationships, which only needs to be done once upon the initial setup of a keyboard. As *mmKey* can be compatible with multiple types of keyboards, such as piano keyboard and computer keyboard, we start with the 1-D case using the white keys of a piano keyboard as an example, and extend it to the general 2-D case for the computer keyboard and phone keypad later.

1-D Case. To complete the keyboard calibration with the least effort, the user can randomly pick and press three keys. As seen in Fig 10a, assuming key w_1, w_6 and w_{10} are pressed during the calibration and the corresponding estimated azimuths by the MUSIC algorithm are represented as $\hat{\varphi}_1$, $\hat{\varphi}_6$ and $\hat{\varphi}_{10}$, we have $\alpha_1 = \hat{\varphi}_6 - \hat{\varphi}_1$ and $\alpha_2 = \hat{\varphi}_{10} - \hat{\varphi}_6$. According to the law of sines, we have

$$\begin{aligned} &\frac{|AD|}{\sin\alpha_1} = \frac{|AC|}{\sin\beta_1}, \text{ in } \triangle ACD \ ; \\ &\frac{|BD|}{\sin\alpha_2} = \frac{|BC|}{\sin\beta_2}, \text{ in } \triangle BCD \ ; \\ &\sin\beta_1 = \sin\beta_2, \end{aligned} \tag{11}$$



(a) Model for piano keyboard.



(b) 1-D model during detection.

Fig. 10: Geometrical models for keyboard calibration. The user only needs to press three known keys, as indicated by the red dots in (a).

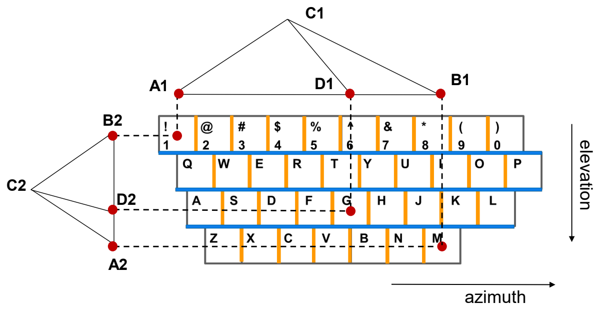


Fig. 11: Generalized 2-D case for keyboard calibration.

where β_1 and β_2 are two unknown angles belonging to two adjacent triangles and forming a straight angle. Denoting the ratio of $|AC|$ to $|BC|$ as η, from equation (11) we have

$$\eta = \frac{|AC|}{|BC|} = \frac{|AD|}{|BD|}\frac{\sin\alpha_2}{\sin\alpha_1}. \tag{12}$$

Since the ratio $\frac{|AD|}{|BD|}$ is already known as $\frac{5}{4}$ in this example. Assuming all of the keystrokes occur at the center of the key, we can derive the value of η. Further, the azimuth boundary of every two adjacent keys can also be derived. For example, as indicated in Fig. 10b, to calculate the boundary between key w_2 and w_3, we apply the law of sines again as

$$\begin{aligned}&\frac{|AE|}{\sin\widetilde{\alpha}_1} = \frac{|AC|}{\sin\widetilde{\beta}_1}, \text{ in } \triangle ACE \text{ ;}\\ &\frac{|BE|}{\sin\widetilde{\alpha}_2} = \frac{|BC|}{\sin\widetilde{\beta}_2}, \text{ in } \triangle BCE \text{ ;}\\ &\sin\widetilde{\beta}_1 = \sin\widetilde{\beta}_2,\end{aligned} \tag{13}$$

where $\widetilde{\alpha}_1$, $\widetilde{\alpha}_2$, $\widetilde{\beta}_1$ and $\widetilde{\beta}_2$ are the angles corresponding to those in Fig. 10a. Then, we get

$$\begin{aligned}&\frac{\sin\widetilde{\alpha}_1}{\sin\widetilde{\alpha}_2} = \frac{|BC|}{|AC|}\frac{|AE|}{|BE|} = \frac{1}{\eta}\frac{|AE|}{|BE|},\\ &\widetilde{\alpha}_1 + \widetilde{\alpha}_2 = \alpha_1 + \alpha_2.\end{aligned} \tag{14}$$

Based on equation (14), the exact values of $\widetilde{\alpha}_1$ and $\widetilde{\alpha}_2$ can be obtained since the ratio $\frac{|AE|}{|BE|}$ is known as $\frac{1}{5}$. Similarly, the boundaries between the other adjacent keys such as (w_1, w_2), ... , (w_9, w_{10}) can be derived. By substracting the absolute azimuth of w_1, the boundary azimuths can be calculated for keystroke recognition.

2-D Case. The geometrical model of the 1-D case can be easily extended to 2-D, where both the elevation and azimuth angles are used for keystroke recognition. As illustrated in Fig. 11, three keys "1", "G" and "M" are pressed for calibration. In the horizontal azimuth direction, we have $\triangle A_1B_1C_1$, from which we can derive all the azimuth boundaries of keys (orange lines), while in the vertical elevation dimension we have $\triangle A_2B_2C_2$ and use it to calculate the elevation boundaries (blue lines). Here C_1 and C_2 denote the same location of the device, viewing from the azimuth and elevation dimensions, respectively.

Given the values of the boundary azimuth and elevation angles of each key, real-time keystrokes can be easily recognized by mapping the estimated keystroke position to the target key on the keyboard with a known layout.

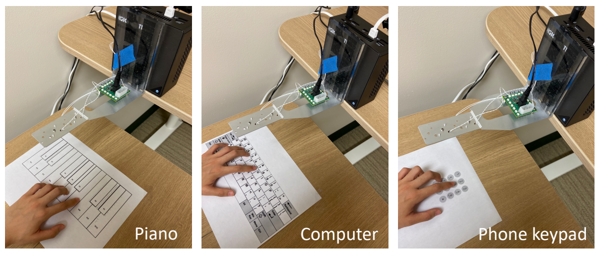


Fig. 12: Setup examples for three difference keyboards.

VII. Experimental Evaluation

We prototype *mmKey* and conduct real-world experiments using a Qualcomm sponsored testbed, which reuses a 802.11ad/ay chipset as a radar-like platform. The default setup is presented in Fig. 12, where the device is put down to cover a flat surface that upholds a printed virtual keyboard. We consider different types of keyboards, including QWERTY computer keyboard, piano keyboard, and smartphone keypad. For each keyboard, we print the layout on a paper, maintaining the same physical size such that users would keep the most familiar typing feeling as on a real keyboard as Fig. 12 shows. Note that with a simple calibration mechanism and the high directionality of mmWave, *mmKey* can easily adapt to any keyboards including user-customized layouts, as long as the geometric arrangement is known. By default, we set the distance between the keyboard and the device to be around 20 cm such that the keyboard will be confidently within the field-of-view (FoV) of our device, which is 100° for both azimuth and elevation directions. The default sampling rate is $f_s = \frac{1}{T_b} = 100$ Hz, where T_b is the burst duration as shown in Fig. 2. The selection of different parameters is further studied in Section VII-D.

We conduct experiments at different locations in both office and home environments, with 10 volunteers involved, including 4 females and 6 males aging from 23 to 32. To obtain the ground truth, we perform the experiments in two different ways: 1) For each keyboard, we generate a random key list covering each key once, and the participants are asked to press the virtual keys by following the list. Each participant will repeat multiple times for each key list. 2) The user is asked to type following a sequence of words/sentences or digits/music scores. Note that participants press the keys in a nearly natural way. Therefore, the keystroke speed is not controlled and it may vary over time for different users. During experiments, we collect CIR series from the testbed when a user is typing and send the data through an Ethernet cable to a computer for processing in MATLAB.

We mainly use three metrics for evaluation. We use *detection accuracy* (DA) and *recognition accuracy* (RA) to quantify how *mmKey* correctly detects the keystrokes and how it localizes and recognizes them, respectively. Based on DA and RA, we calculate the overall accuracy (OA) as $OA = DA \times RA$. DA and RA are defined as follows:

$$\begin{aligned} DA &= \frac{\#\ of\ detected\ keystrokes}{\#\ of\ total\ keystrokes}; \\ RA &= \frac{\#\ of\ recognized\ keystrokes}{\#\ of\ detected\ keystrokes}. \end{aligned} \tag{15}$$

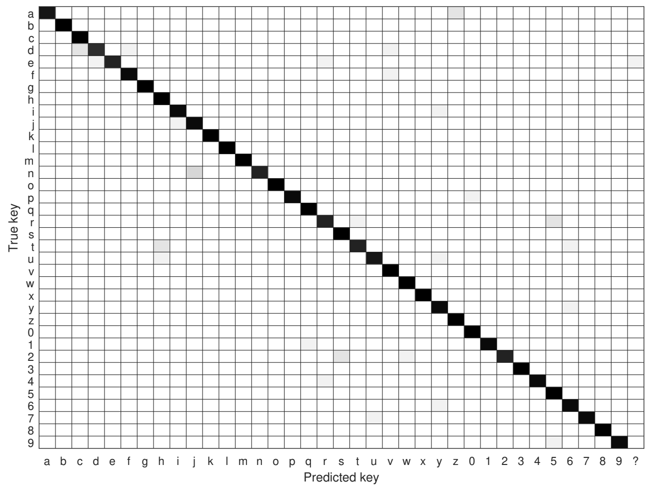
Below, we first evaluate the overall performance of three different types of keyboards and then report the parameter study in Section VII-D.

A. Virtual Computer Keyboard

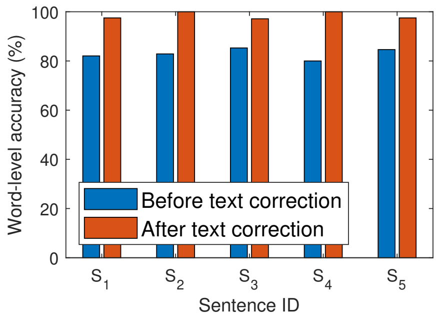
1) Performance on Individual Keys: We first investigate the performance of *mmKey* for a virtual computer keyboard. The printed standard alphanumeric keyboard has a common QWERTY-based layout with the distance between neighboring keys 19 mm. We involve the keys of letter and digit in the experiments and select “1”, “G” and “M” as the landmark keys for calibration. The OA confusion matrix for recognizing 36 keys (26 letters plus 10 digits) on a virtual computer keyboard is shown in Fig. 13a. As we can see, *mmKey* achieves remarkable keystroke recognition with an average OA of 95.42% for a computer keyboard. As illustrated, there exist some detection errors. Some samples of certain keys are recognized as the neighboring keys, especially the one below the real key. This is because there exist reflections from the knuckles leading to the estimation error in the elevation direction. In real applications when users are typing typical texts, we believe these errors can be easily recovered by the mature spell check techniques, as demonstrated in the next section.

2) Word Recovery: We further explore the capabilities of *mmKey* on recovering the input sentences and evaluate the accuracy in word level. Adopting the same methods in WiKey [18] to collect sentences samples, we ask the user to type each of the following sentences 5 times on the printed computer keyboard: S_1 = “the quick brown fox jumps over the lazy dog”, S_2 = “nobody knew why the candles blew out”, S_3 = “the autumn leaves look like golden snow”, S_4 = “nothing is as profound as the imagination”, and S_5 = “my small pet mouse escaped from his cage”.

We first run *mmKey* on the CIR data and obtain the direct outputs, i.e., a sequence of recognized keys. Then we



(a) Confusion matrix for 36 keys on computer keyboard.



(b) Word recovery accuracy in sentence input testing.

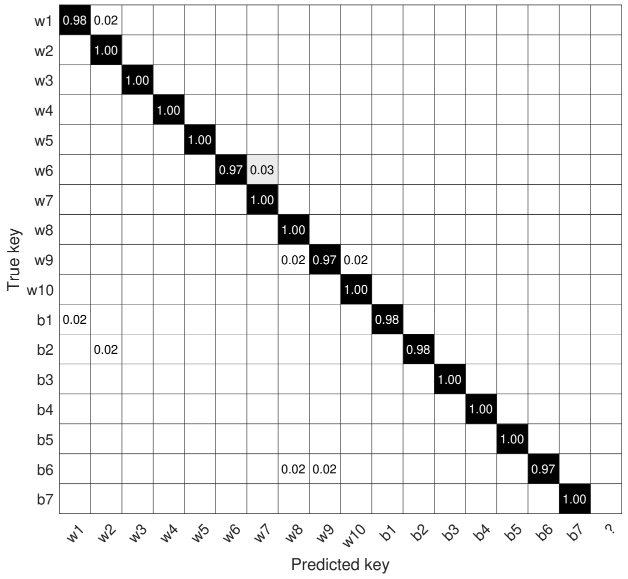
Fig. 13: Performance on virtual computer keyboard.

feed the outputs into Grammarly[1] for correction, which is a popular commercial English writing tool. Here we calculate the word-level accuracy (WA) by $WA = \frac{\#\ of\ correct\ words}{\#\ of\ total\ words}$ and illustrate the results in Fig. 13b. It is as expected the WA on the direct outputs of *mmKey*, which is about 80%, is not as high as its OA since a single mis-recognized letter will lead to a wrongly recognized word. With the help of spell check/text correction, the word-level mis-recognition can be easily corrected with a considerable accuracy greater than 97%. With the high accuracy, *mmKey* could promise a ubiquitous virtual keyboard for mobile and portable usage everywhere in practice.

B. Virtual Piano Keyboard

In this part, we report the overall performance of *mmKey* in the recognition of keystroke on a piano keyboard. To adapt to the system capacity of limited FoV, we employ a segment of piano keyboard consisting of ten white keys and seven black keys in between. We denote the white keys and black keys from left to right as w_1, w_2, ..., w_{10} and b_1, b_2, ..., b_7, respectively. Three keys w_1, b_4 and w_{10} are used for calibration. Note that as piano only have two rows of keys, we use the weighted average of elevation angles of w_1, b_4, w_{10} as the elevation boundary, *i.e.*, $\frac{1}{2}(\frac{\theta_{w_1}+\theta_{w_{10}}}{2}+\theta_{b_4})$. Different from

[1]https://www.grammarly.com


(a) Confusion matrix for single-keystroke.

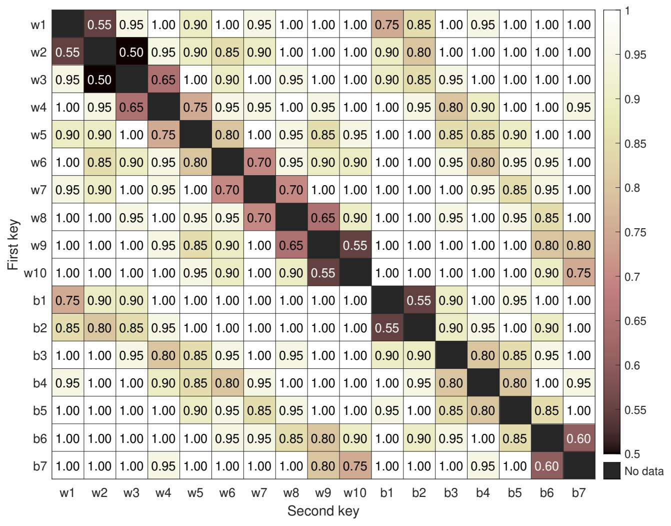
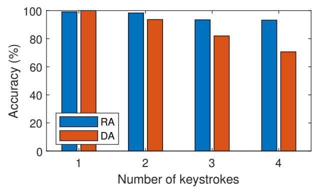

(b) Accuracy when pressing two concurrent keys.

Fig. 14: Performance of virtual piano keyboard.


(a) Accuracy vs number of keystrokes.

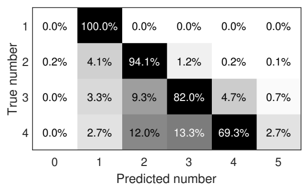
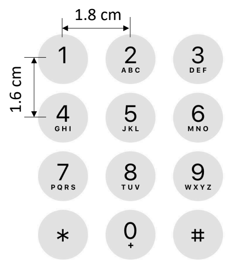

(b) Confusion matrix for detection.

Fig. 15: Multi-keystroke accuracy.


(a) Phone keypad layout.

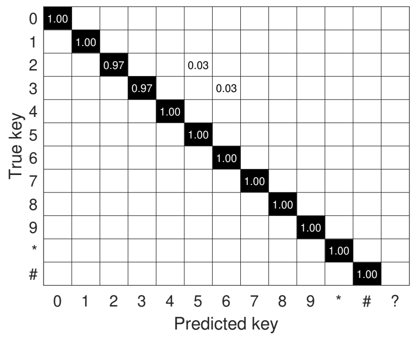

(b) Confusion matrix for phone keypad.

Fig. 16: Performance on virtual phone keypad.

computer keyboards, users may press multiple keys concurrently when playing the piano. Thus we conduct experiments with two scenarios: *single keystroke* and *simultaneous multiple keystrokes*.

1) Single-Key Keystroke: For single-key keystroke case, the experimenter is asked to press each key 60 times. The pressing order follows a random sequence generated from MATLAB as stated before.

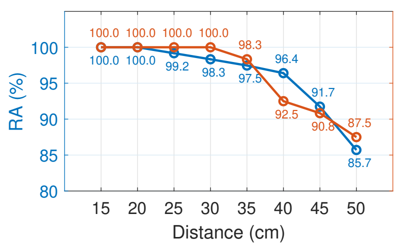
Accounting for both the white keys and black keys, Fig. 14a shows the confusion matrix for single-key keystrokes, where "?" means the miss of detection. As seen, *mmKey* achieves a high OA of 99.12%.

2) Simultaneous Multiple Keystrokes: Users may need to press multiple keys simultaneously to play the piano. We first look at the two-key case. The experimenter is asked to press two keys at the same time, and each combination of keys is repeated 20 times. The instances of two-key keystroke spectrum can be found in Fig. 8 while the OA for all combinations can be seen in Fig. 14b. As shown, *mmKey* recognizes the keystrokes accurately when two keys are located far enough. However, when two pressed keys get closer especially for the adjacent keys, the accuracy may decrease due to the co-located fingers. The overall accuracy for double-key keystroke recognition is 92.54% for all cases, and the accuracy becomes 96.93% for non-adjacent keys. Also, observing the accuracy along the diagonal, we can find that the OA decreases near the edge of the keyboard due to the effects of inter-finger blockage at the edge locations.

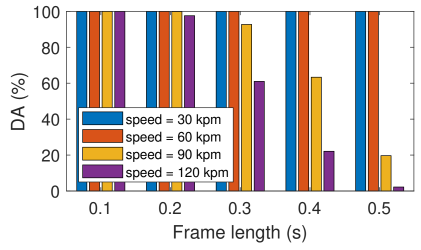
Now we extend to the cases of > 2 concurrent keys by investigating the OA with respect to the number of concurrent keystrokes. The results show that the OA decreases to 76.67% when there are three keys being pressed and further decreases to 65.94% for four keys. We further examine the detection accuracy, as illustrated in Fig. 15b. As seen, more keystrokes will lead to more miss detection due to the blockage between multiple fingers, but do not affect much the recognition accuracy as illustrated by Fig. 15a. Once a keystroke is detected, *mmKey* can recognize it accurately.

C. Virtual Phone Keypad

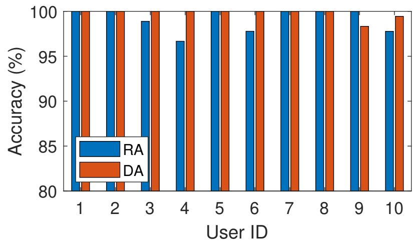
We also test *mmKey* against a virtual phone keypad. Phone keypad is a widely-used keyboard in daily life due to the proliferation of mobile devices. As shown in Fig. 16a, the applied keypad has 12 keys including digits 0 to 9 and two special characters "*" and "#". The keypad is printed on a paper with the same size as the real phone keypad with an inter-key separation of 1.8 cm in horizontal direction and 1.6 cm in vertical dimension. In the keyboard calibration phase,



(a) Impact of range.

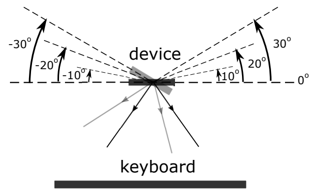


(b) Impact of typing speed and W.

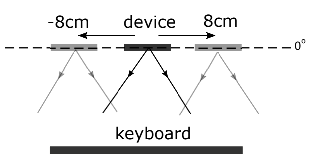


(c) Impact of user diversity.

Fig. 17: Impact of factors on *mmKey* performance: (a) range, (b) speed and window length, (c) sampling rate and (d) user heterogeneity.



(a) Rotation of device.



(b) Translation of device.

Fig. 18: Setup of various placements with distortions (front view).

key “1”, “8” and “#” are pressed to estimate the boundaries. In the real-time recognition phase, the user is asked to press different keys following the random-order list. There are total 40 samples for each key. The confusion matrix of OA is shown in Fig. 16b. As we can see, although the keys are close to each other (< 2 cm), the keystrokes can be recognized accurately and reliably with an OA 98.33%.

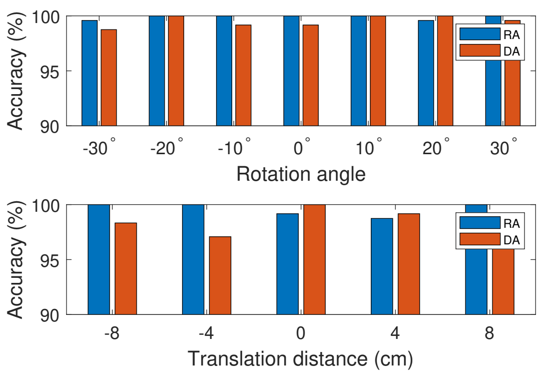


Fig. 19: Performance for various placements with distortions.

D. Parameter Study

In this section, we benchmark *mmKey* by studying the impacts of different parameters. Without loss of generality, we evaluate with the piano keyboard unless otherwise specified.

1) Impact of Range: The performance of *mmKey* may vary over the distance from the target to the device. To investigate the impact, we perform experiments with eight different keyboard-radio distances, ranging from 15 cm to 50 cm with an increment of 5 cm. As depicted in Fig. 17a, the increase of distance leads to a degradation in both DA and RA as expected. This is because both the spatial resolution and the reflection strengths decrease over range. In other words, the keys become relatively narrower from the view of the radar and the reflected signals become weaker at larger distances, which lead to detection and recognition errors.

2) Impact of Keystroke Speed and Window Length: Keystroke speed is another important property that may raise users' concern. Since the motion detection and segmentation are applied with the assumption that finger motions do not overlap with each other, there are two factors that may impact the DA of *mmKey*: *keystroke speed* and *length of sliding window* W. We benchmark these two factors by performing experiments with different typing speeds, quantified by keystroke per minute (kpm). The DA of different combinations of speed and W are presented in Fig. 17b. As seen, *mmKey* achieves consistently high accuracy for 30 kpm and 60 kpm, regardless of the W values. For faster typing speeds, the detection rate remains high if a short window is used, but it decreases quickly with larger W since two adjacent keystrokes are easily mistaken as a merged single keystroke. For performance evaluation, the default W is set to be 0.1s, which can handle various typing speeds.

3) Impact of User Heterogeneity: In this part, we study the impact of user heterogeneity. During the experiments, We enroll 10 participants labeled as “User 1” to “User 10”. We use a piano keyboard for evaluation without loss of generality. Among all the 10 participants, only one (User 9) is familiar with the piano keyboard and good at playing the piano, and two of them (User 2 and 3) have some basic knowledge of the piano. Also, during the data collection, there is no restriction on using one or two hands. Two of them (User 1 and 9) put both of their hands on the keyboard and type different keys using different hands while others use one hand. Therefore, these 10 users provide a reasonable level of diversity in terms of different typing behaviors. We calibrate the keyboard once at the beginning, and then apply it to all the participants. The results in Fig. 17c show a near 100% accuracy for both keystroke detection and recognition, implying that *mmKey* can support diverse users with only one-time calibration pre-processing.

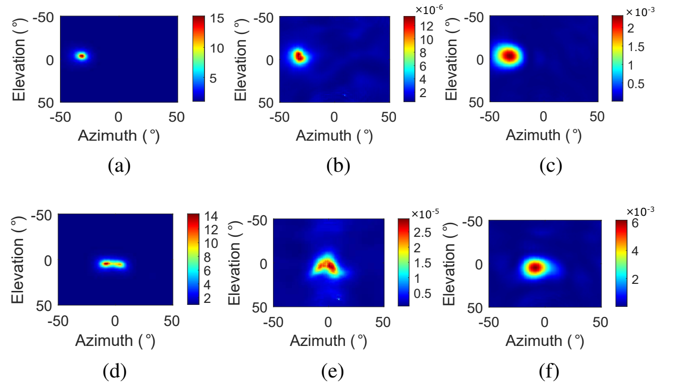


Fig. 20: Examples of obtained spectrum by three different spatial spectrum estimators: MUSIC, MVDR and CBF. (a)-(c): Single-key case by MUSIC, MVDR and CBF, respectively. (d)-(f): Double-key case by MUSIC, MVDR and CBF, respectively.

4) Impact of Device Placement: As the typical experimental setup presented in Fig. 12, the device is put "parallel" to the keyboard and the Rx locates at almost the middle of the keyboard, which looks symmetric. However, in practice, the placement of device is not always perfect, but with potential distortions due to rotation and translation. Therefore, we study the impacts by placing the device with different orientations and offsets, as detailed below.

Rotation. In this experiment, the device is not perfectly parallel to the keyboard but deviates with some angle as Fig. 18a presents. Here we test deviation angles ranging from -30° to 30° with a step of 10°, where "-" represents the clockwise rotation and "+" is anticlockwise from the front view of users. The results are shown in Fig. 19, which shows remarkable robustness to the orientation distortions with consistently high DA and RA retained.

Translation. For the translation case, the device is not aligned with the center of the keyboard but is moved by a certain distance. As Fig. 18b illustrates, we shift the device from the center of the keyboard by different distances. Specifically, we tested at -8 cm, -4 cm, 0 cm, 4 cm, 8 cm, where "-" and "+" represent the left and right directions from the front view, respectively. As depicted in Fig. 19, the results show no significant difference among the tested distances in both DA and RA, indicating that *mmKey* can adapt to various translation placements.

5) Comparing Spatial Spectrum Estimators: We employ MUSIC to achieve super resolution in space on our device. In this section, we compare *mmKey* with existing beamforming techniques, including conventional beamforming (CBF) and well-known minimum variance distortion response (MVDR) beamforming (a.k.a. Capon beamforming).

Fig. 20a-c are the spectrum of pressing a single key w_1 generated by the three spatial spectrum estimators. As we can visually compare, for single-finger keystroke case, the MUSIC algorithm can produce the finest spectrum due to its super-resolution property. For the double-finger keystroke case, MUSIC also performs much better than the other two.

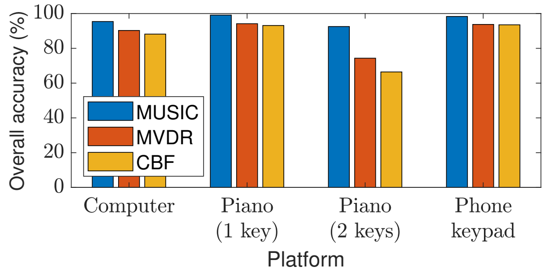


Fig. 21: Overall accuracy on different platforms using three spatial spectrum estimators: MUSIC, MVDR and CBF.

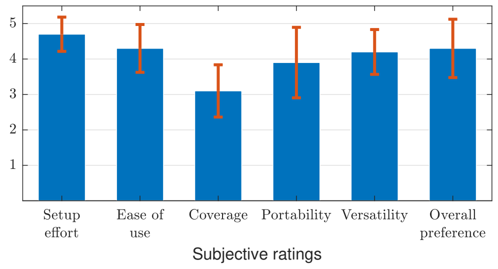


Fig. 22: Means and standard deviations of the subjective ratings, all on 5-level scales where 5 is the most positive rating.

Fig. 20d-f illustrate the spectrum of keystrokes of two adjacent keys w_5 and w_6. As we can see, only MUSIC algorithm can detect two neighboring sources while it becomes hard to distinguish the multiple keystrokes by MVDR or CBF. MVDR is slightly better than CBF in this case, but is not as focusing as MUSIC.

To quantitatively compare their performance, we apply three estimators on the same dataset in different scenarios, respectively. Fig. 21 shows the OA for each estimator under each setting. It can be observed that for all the different keyboards, *mmKey* based on MUSIC achieves the highest accuracy. The OA drops with MVDR and CBF due to their limited resolution, especially for the double-key case, where the MUSIC-based approach outperforms MVDR and CBF significantly with an over 90% OA.

E. Subjective Evaluation with User Study

Finally, we carry out a user study on all the participants for their feedback on user experience. We collect subjective measurements from the ten volunteers through an online questionnaire. The users are asked about the perceived experience of using *mmKey* as an input tool, including the setup complexity, ease of use, coverage, portability and so on. The responses from real users are summarized in Fig. 22. All responses are scaled from 1 to 5 where 5 is the most positive rating. As we see, *mmKey* is rated positively with the average responses to most questions greater than 4.0. We note that each volunteer is also asked to give an overall rating on how he/she likes *mmKey*. On average, ten volunteers give a positively 4.0+ average rating and two of them express their willingness to experience it in real applications.

VIII. Limitations and Discussions

Typing Speed. The average typing speed on a physical computer keyboard is about 37∼40 words per minute, which translates to about 185∼200 characters (keys) per minute. *mmKey* supports a reasonably good speed of 120 kpm, as people generally type more slowly on a virtual printed keyboard due to the lack of keypress feedback. However, the performance of *mmKey* may deteriorate for fast typing (>120 kpm), during which the finger motion and hand motion overlap with each other. It is worthwhile to study the segmentation of the CIR time series and explore new features in order to support reliable recognition of high-speed keystrokes.

Detection Range. While *mmKey* can support accurate keystroke recognition ($> 90\%$) at a range up to 45 cm, which is large enough to cover a computer keyboard given the common FoV of mmWave radios, the detection range still needs to be improved so as to support the implementation of a full piano keyboard. As the device-keyboard distance increases, the reflected signals become weaker and the keys become relatively smaller from the view of the radar, which makes the keystroke detection and recognition harder. We keep it as future work to investigate the antenna diversity for a better resolution and thus larger ranges.

Keyboard Calibration. *mmKey* needs minimal calibration (i.e., only three keystrokes) to associate the key locations relative to the device with the actual keys. Except for that, *mmKey* does not need any other training, making it deployable anywhere for a ubiquitous virtual keyboard. The calibration only needs to be done once for a specific setup. However, it is recommended not to change the relative location between the device and the printed keyboard; Otherwise, a re-calibration would be needed to associate the new mapping.

Cost and Device Readiness. *mmKey* is implemented on a mmWave platform sponsored by Qualcomm. *mmKey* itself does not introduce any extra hardware cost. The Qualcomm platform does need some modifications and is admittedly bulky for its current form. Nevertheless, the platform only uses a single commodity 60GHz WiFi chipset with an additional antenna array and thus would be fairly low-cost and tiny once mass production. And we sincerely hope Qualcomm would publicly release the testbed soon. In the meanwhile, we plan to extend *mmKey*, as a software solution, to other mmWave platforms such as TI mmWave radars.

Portability. Although current *mmKey* prototype still requires additional hardware and is not as portable as wearables and multi-functional keyboards, it enables a virtual keyboard by reusing a mmWave device wherever it is already available. With the miniaturization of antennas and chips, it is expected that the mmWave hardware will become lighter, portable, cheaper and energy-saving as a tiny chipset that will be widely available on home routers, smartphones [49], [50] and vehicles [51]. Then *mmKey* could immediately enable virtual typing around those devices in the integrated IoT system. Although wearables and hand-equipped sensors allow typing anywhere, they could not easily input information to the targeted IoT devices in the above scenarios since the wearables are unlikely connected to the IoT devices.

Potential Applications. As the first virtual keyboard using a single mmWave radio, the core contribution of *mmKey* is the processing pipeline that enables accurate localization of micro motion, which can also enable in-air finger tracking/gesture recognition and similar interactive applications.

IX. Conclusion

This paper presents *mmKey*, the first universal virtual keyboard system using a single mmWave device. *mmKey* achieves accurate multi-finger keystroke detection and recognition and supports various keyboard layouts. It employs a novel pipeline of signal processing to detect, segment, and recognize keystrokes, without requiring any training. We evaluate the performance of *mmKey* on various keyboards, including virtual piano keyboard, virtual phone keypad and virtual computer keyboard. The results demonstrate an overall accuracy over 95% for single-key case on different keyboard layouts and a recognition accuracy over 90% for multi-key scenario, which translates to a word recognition accuracy above 97%.

References

[1] M. Funk, A. Sahami, N. Henze, and A. Schmidt, “Using a touch-sensitive wristband for text entry on smart watches,” in *CHI ’14 Extended Abstracts on Human Factors in Computing Systems*, p. 2305–2310, ACM, 2014.

[2] C.-M. Wu, C.-W. Hsu, T.-K. Lee, and S. Smith, “A virtual reality keyboard with realistic haptic feedback in a fully immersive virtual environment,” *Virtual Reality*, vol. 21, no. 1, pp. 19–29, 2017.

[3] W. Chen, Y. Lian, L. Wang, R. Ruby, W. Hu, and K. Wu, “Virtual keyboard for wearable wristbands,” in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pp. 1–2, 2017.

[4] Y. Zhao, C. Lian, X. Zhang, X. Sha, G. Shi, and W. J. Li, “Wireless iot motion-recognition rings and a paper keyboard,” *IEEE Access*, vol. 7, pp. 44514–44524, 2019.

[5] Y. Yin, Q. Li, L. Xie, S. Yi, E. Novak, and S. Lu, “Camk: A camera-based keyboard for small mobile devices,” in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.

[6] H. Ji, J. Chen, Q. Lin, and A. Li, “A local fingertips movement and fingertips clustering based virtual keyboard adopting a camera,” in *Proceedings of the 2018 International Conference on Computing and Pattern Recognition*, pp. 61–67, 2018.

[7] T. Murase, A. Moteki, G. Suzuki, T. Nakai, N. Hara, and T. Matsuda, “Gesture keyboard with a machine learning requiring only one camera,” in *Proceedings of the 3rd Augmented Human International Conference*, pp. 1–2, 2012.

[8] X. Su, Y. Zhang, Q. Zhao, and L. Gao, “Virtual keyboard: A human-computer interaction device based on laser and image processing,” in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 321–325, IEEE, 2015.

[9] L. Zhuang, F. Zhou, and J. D. Tygar, “Keyboard acoustic emanations revisited,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 1, pp. 1–26, 2009.

[10] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, “Context-free attacks using keyboard acoustic emanations,” in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pp. 453–464, 2014.

[11] J. Wang, K. Zhao, X. Zhang, and C. Peng, “Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization,” in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pp. 14–27, 2014.

[12] B. Obermaier, G. R. Muller, and G. Pfurtscheller, “” virtual keyboard” controlled by spontaneous eeg activity,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 11, no. 4, pp. 422–426, 2003.

[13] R. Scherer, G. Muller, C. Neuper, B. Graimann, and G. Pfurtscheller, "An asynchronously controlled eeg-based virtual keyboard: improvement of the spelling rate," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 979–984, 2004.

[14] U. Orhan, K. E. Hild, D. Erdogmus, B. Roark, B. Oken, and M. Fried-Oken, "Rsvp keyboard: An eeg based typing interface," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 645–648, IEEE, 2012.

[15] B. Wang, Q. Xu, C. Chen, F. Zhang, and K. J. R. Liu, "The promise of radio analytics: A future paradigm of wireless positioning, tracking, and sensing," *IEEE Signal Processing Magazine*, vol. 35, no. 3, pp. 59–80, 2018.

[16] K. J. R. Liu and B. Wang, *Wireless AI: Wireless Sensing, Positioning, IoT, and Communications*. Cambridge University Press, 2019.

[17] B. Chen, V. Yenamandra, and K. Srinivasan, "Tracking keystrokes using wireless signals," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 31–44, 2015.

[18] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using wifi signals," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 90–102, 2015.

[19] Y. Meng, J. Li, H. Zhu, X. Liang, Y. Liu, and N. Ruan, "Revealing your mobile password via wifi signals: Attacks and countermeasures," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 432–449, 2019.

[20] *Qualcomm 802.11ad 60GHz WiFi*, 2019.

[21] *Talon AD7200 Multi-Band Wi-Fi Router*, 2019.

[22] P. Marquardt, A. Verma, H. Carter, and P. Traynor, "(sp) iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 551–562, 2011.

[23] J. Liu, Y. Chen, and M. Gruteser, "Vibkeyboard: virtual keyboard leveraging physical vibration," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pp. 507–508, 2016.

[24] K. Ling, Y. Liu, K. Sun, W. Wang, L. Xie, and Q. Gu, "Spidermon: Towards using cell towers as illuminating sources for keystroke monitoring," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 666–675, IEEE, 2020.

[25] J. Zhang, X. Zheng, Z. Tang, T. Xing, X. Chen, D. Fang, R. Li, X. Gong, and F. Chen, "Privacy leakage in mobile sensing: Your unlock passwords can be leaked through wireless hotspot functionality," *Mobile Information Systems*, vol. 2016, 2016.

[26] G. Bui, B. Morago, T. Le, K. Karsch, Z. Lu, and Y. Duan, "Integrating videos with lidar scans for virtual reality," in *2016 IEEE Virtual Reality (VR)*, pp. 161–162, 2016.

[27] V. Tam and L. Li, "Integrating the kinect camera, gesture recognition and mobile devices for interactive discussion," in *Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) 2012*, pp. H4C–11–H4C–13, 2012.

[28] D. D. F. Jr. and N. C. Currie, "Microwave and millimeter-wave systems for wall penetration," in *Targets and Backgrounds: Characterization and Representation IV* (W. R. Watkins and D. Clement, eds.), vol. 3375, pp. 269 – 279, International Society for Optics and Photonics, SPIE, 1998.

[29] F. Zhang, C. Wu, B. Wang, and K. J. R. Liu, "mmeye: Super-resolution millimeter wave imaging," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[30] F. Wang, F. Zhang, C. Wu, B. Wang, and K. J. R. Liu, "Vimo: Multiperson vital sign monitoring using commodity millimeter-wave radio," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1294–1307, 2021.

[31] H. Abdelnasser, M. Youssef, and K. A. Harras, "Wigest: A ubiquitous wifi-based gesture recognition system," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1472–1480, IEEE, 2015.

[32] H. Li, W. Yang, J. Wang, Y. Xu, and L. Huang, "Wifinger: talk to your smart devices with finger-grained gesture," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 250–261, 2016.

[33] D. Wu, R. Gao, Y. Zeng, J. Liu, L. Wang, T. Gu, and D. Zhang, "Fingerdraw: Sub-wavelength level finger motion tracking with wifi signals," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–27, 2020.

[34] L. Sun, S. Sen, D. Koutsonikolas, and K.-H. Kim, "Widraw: Enabling hands-free drawing in the air on commodity wifi devices," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 77–89, 2015.

[35] N. Yu, W. Wang, A. X. Liu, and L. Kong, "Qgesture: Quantifying gesture distance and direction with wifi signals," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–23, 2018.

[36] Y. Ma, G. Zhou, S. Wang, H. Zhao, and W. Jung, "Signfi: Sign language recognition using wifi," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–21, 2018.

[37] J. Shang and J. Wu, "A robust sign language recognition system with multiple wi-fi devices," in *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*, pp. 19–24, 2017.

[38] F. Zhang, C. Wu, B. Wang, H.-Q. Lai, Y. Han, and K. J. R. Liu, "Widetect: Robust motion detection with a statistical electromagnetic model," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 3, pp. 1–24, 2019.

[39] F. Zhang, C. Wu, B. Wang, M. Wu, D. Bugos, H. Zhang, and K. R. Liu, "Smars: sleep monitoring via ambient radio signals," *IEEE Transactions on Mobile Computing*, 2019.

[40] H. Abdelnasser, K. A. Harras, and M. Youssef, "Ubibreathe: A ubiquitous non-invasive wifi-based breathing estimator," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 277–286, 2015.

[41] C. Wu, F. Zhang, B. Wang, and K. J. R. Liu, "mmtrack: Passive multi-person localization using commodity millimeter wave radio," in *IEEE International Conference on Computer Communications*, April 27-30 2020.

[42] P. S. Santhalingam, A. A. Hosain, D. Zhang, P. Pathak, H. Rangwala, and R. Kushalnagar, "mmasl: Environment-independent asl gesture recognition using 60 ghz millimeter-wave signals," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–30, 2020.

[43] Y. Zhu, Y. Zhu, B. Y. Zhao, and H. Zheng, "Reusing 60ghz radios for mobile radar imaging," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 103–116, 2015.

[44] Z. Meng, S. Fu, J. Yan, H. Liang, A. Zhou, S. Zhu, H. Ma, J. Liu, and N. Yang, "Gait recognition for co-existing multiple people using millimeter wave sensing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 849–856, 2020.

[45] C. Xu, Z. Li, H. Zhang, A. S. Rathore, H. Li, C. Song, K. Wang, and W. Xu, "Waveear: Exploring a mmwave-based noise-resistant speech sensing for voice-user interface," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 14–26, 2019.

[46] S. D. Regani, C. Wu, B. Wang, M. Wu, and K. J. R. Liu, "mmwrite: Passive handwriting tracking using a single millimeter wave radio," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[47] C. Wu, F. Zhang, B. Wang, and K. J. R. Liu, "msense: Towards mobile material sensing with a single millimeter-wave radio," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 3, pp. 1–20, 2020.

[48] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE transactions on antennas and propagation*, vol. 34, no. 3, pp. 276–280, 1986.

[49] *Soli Radar-Based Perception and Interaction in Pixel 4*, 2020. https://ai.googleblog.com/2020/03/soli-radar-based-perception-and.html.

[50] *Apple U1 TMKA75 Ultra Wideband (UWB) Chip Analysis*, 2020. https://www.techinsights.com/blog/apple-u1-tmka75-ultra-wideband-uwb-chip-analysis.

[51] *Vayyar Imaging Announces First 60GHz In-car Automotive Grade Radar-on-chip*, 2020. https://www.prnewswire.com/news-releases/vayyar-imaging-announces-first-60ghz-in-car-automotive-grade-radar-on-chip-300983523.html.