2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS) | 978-1-7281-7002-2/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICDCS47774.2020.00073

airFinger: Micro Finger Gesture Recognition via NIR Light Sensing for Smart Devices

Qian Zhang*† Yetong Cao† Huijie Chen† Fan Li† Song Yang† Yu Wang‡ Zheng Yang* Yunhao Liu*
* Tsinghua University, China
† Beijing Institute of Technology, China
‡ Temple University, USA

***Abstract*—Micro finger gesture recognition is an emerging approach to realize more friendly interaction between human and smart devices, especially for small wearable devices, such as smartwatches and virtual reality glasses. This paper proposes airFinger, a novel solution utilizing NIR light sensing to realize both real-time gesture recognition and finger tracking aiming at micro finger gestures. Using a custom NIR-based sensor with novel algorithms to capture subtle finger movements, airFinger enables to detect a rich set of micro finger gestures and track finger movements in terms of scrolling direction, velocity, and displacement. Besides, airFinger is capable of effective noise mitigation, gesture segmentation, and reducing false recognition due to the unintentional actions of users. Extensive experimental results demonstrate that airFinger has robustness against individual diversity, gesture inconsistency, and many other impacts. The overall performance reaches an average accuracy as high as 98.72% over a set of 8 micro finger gestures among $10,000$ gesture samples collected from 10 volunteers.**

***Keywords*-micro finger gesture, light sensor, gesture recognition, interaction**

I. Introduction

Micro finger gestures have gained an impetus for interaction between human and smart devices in recent years. This kind of gesture is performed by subtle finger movements, such as thumb-tip rubbing or drawing against index finger-tip, which allows more effective and rapid manipulation than gestures performed by the whole hand or arm. Additionally, it is good for privacy because users can operate devices without being noticed by others, avoiding fatigue and social awkwardness by using body-scale gestures in public places [1], [2]. Therefore, micro finger gestures are more natural, fast, and unobtrusive, especially for interaction with wearable devices, such as smartwatches and virtual reality glasses.

As micro finger gestures emerge as a promising form for interaction, many studies have exploited to recognize such fine-grained gestures. Google's Soli sensor [3] uses millimeter-wave radar to sense micro finger gestures. However, it has high energy consumption with a high-frequency. Other RF-based methods use Wi-Fi [4]–[9], RFID [10]–[12], or acoustic signals [13]–[17] to detect gestures, which are vulnerable to environmental changes. Camera-based methods utilize depth and infrared cameras [18]–[20] or RGB cameras [21] to enable

This work is supported in part by the NSFC under grant No. 61832010, 61632008, 61672319, 61872081, 61632013, 61772077. Fan Li is the corresponding author.

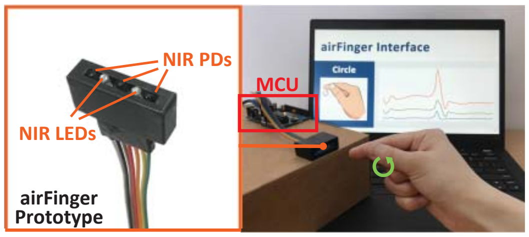


Fig. 1. Recognize micro finger gestures using a prototype of airFinger.

users to interact with smart devices by hand gestures. These methods might present privacy risks and also have relatively high energy and processing consumption. Besides, infrared pyroelectric sensors are leveraged to recognize micro thumb-tip gestures [22]. This approach is adaptable to the recognition of micro finger gestures, but its performance suffers from varied surrounding temperatures. Magnetic sensors [23], [24] can also sense finger-level gestures. However, they need to instrument the fingers with extra sensors.

Recently, an alternative type of methods using Near Infrared (NIR) light sensing may solve the above issues [25]–[28]. It is because NIR emitters/LEDs and receivers/photodetectors (PDs) are highly energy-effective. Additionally, the NIR sensors are small ($3mm$ in diameter of each LED or PD) and easy to be deployed, which benefits to be augmented in existing smart devices, especially for small wearable devices. Thus, this kind of method can achieve low energy consumption and low processing power for reliably micro finger gesture recognition. However, its inability to track finger movements brings limits to its application scopes. A technique capable of micro finger gesture detection and finger tracking would be more desirable, which is relatively challenging and remains unexploited. Therefore, we intend to enable the NIR-based method to achieve both gestures detection and finger tracking for micro finger gestures.

In this paper, we propose airFinger, a technique to reliably detect micro finger gestures and track finger movements in real-time via sensing reflected NIR light from fingers. A typical scenario is shown in Fig. 1, in which a developed custom prototype of airFinger recognizes a circle micro finger gesture performed close to it.

To realize high accurate gesture recognition and real-time finger movements tracking, airFinger faces two key challenges.

The first challenge is to enable airFinger robust to interferences. On one hand, although NIR PDs are sensitive to light changes, which is suitable to capture imperceptible and miniature finger movements, they are also affected by surrounding varied sunlight intensities and other objects moving beside the fingers. To enhance robustness, we design algorithms that enable to automatically mitigate noise, dynamically segment gestures, as well as preserve unique Received Signal Strength (RSS) patterns of micro finger gestures. On the other hand, we observe that people exhibit different RSS patterns for the same gesture (individual diversity) and perform gestures slightly differently from time to time because of different finger positions, towards angles, and moving speeds (gesture inconsistency). To reduce these influence on recognition accuracy, we select 25 kinds of RSS features that are robust against individual diversity and gesture inconsistency via feature importance feedback from a Random Forest (RF)-based classifier, which also saves computing costs and avoids pre-training.

Another challenge of airFinger is to synchronously identify finger moving direction, velocity, and displacement. Although NIR PDs can sense small RSS effects from the micro movements, how to precisely track fingers from the RSS patterns for micro finger gestures is an issue that has not been studied. We develop an energy-effective NIR-based prototype and design a processing-efficient algorithm, which drives airFinger to synchronously track fingers in real-time.

We evaluate the performance of airFinger with $10,000$ gesture samples collected from 10 volunteers. Experimental results show that it can achieve a high average accuracy of up to 98.72% among 8 micro finger gestures.

The main contributions of this paper are summarized in the following:

- We propose airFinger, to our best knowledge, the first work that enables to accurately detect and synchronously track micro finger gestures using NIR light sensing. It has a small size, low cost, and can be deployed with existing smart devices, which provides an alternative new type of interaction between humans and devices.
- We develop a system integrating a custom prototype of NIR-based sensor and energy-effective and power-efficiency algorithms to effectively detect 8 micro finger gestures and track fingers in terms of scrolling direction, velocity, and displacement.
- We conduct extensive experiments with 10 volunteers under different real scenarios. The results show that airFinger reaches an average accuracy as high as 98.72% over $10,000$ gesture samples, which highlights its ability to enable future micro finger gestures interaction with ubiquitous computing applications.

The rest of this paper is organized as follows. Section II surveys related work. Section III describes NIR principles and a gesture set. Section IV presents the design details. Section V shows evaluation results and Section VI discusses future work. Finally, Section VII concludes the paper.

II. Related Work

This section reviews related works on gesture and motion recognition according to the following categories.

NIR light sensing-based methods: According to light propagation and reflection models, many works leverage visible light intensity or shadow to infer gestures and finger positions, such as Okuli [25] and LiGest [29]. Besides, an infrared-based sensor is used to detect 3D hand motion direction [26]. However, these methods are not suitable for micro finger gestures with subtle hand motions. Focusing on finger-level gestures, zSense [28] uses NIR sensing to recognize gestures with low power, low energy, and low cost. Compared to zSense, preserving its advantages of low consumption, our work enables to synchronously track fingers and effectively detect micro finger gestures.

Camera-based methods: Depth cameras, such as Kinect, have been used to design a hand gesture recognition system [18], [19]. Leap Motions can be combined with Kinect to recognize gestures, which provides more kinds of features to improve recognition performance [20]. Besides, a technique uses only RGB cameras on off-the-shelf mobile devices to recognize in-air gestures [21]. These methods can achieve high recognition accuracy. However, they have potential issues of privacy and high energy and computing cost [30], [31].

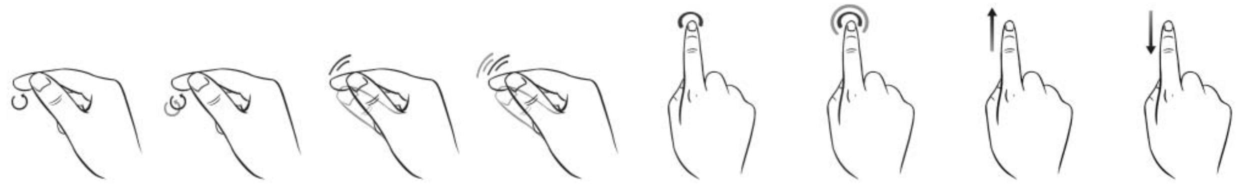
RF-based methods: RF signals (e.g., radar, sound, Wi-Fi, RFID) have also been used for effective finger gesture detecting. Soli [3] uses $60GHz$ radar to achieve sub-millimeter accuracy of 4 gestures. FingerIO [13], Echotrack [14], Strata [15], LLAP [16], and VSkin [17] use inaudible sound to measure the distance between hands and devices or the distance of finger movements. WiGest [5], WiFinger [6], Widar [7], and WiAG [8] reuse existing Wi-Fi signals and extract unique patterns to infer gestures. Besides, RFID technology has also been used to detect gestures by building theoretical models to depict signal changes received from RFID readers [10]–[12]. These RF-based methods utilize existing signals to track or recognize human gestures, however, they might have high cost and be susceptible to ambient disturbances.

Other methods: Magnetometers have been explored for high-precision finger tracking, such as uTrack [23], SynchroWatch [24], and FingerPad [32]. FingerSound [33] leverages an inertial measurement unit to capture thumb movements. Photoplethysmography sensors enable to track 9 finger-level gestures [34] and detect 10 gestures by wearing a smartwatch [35]. These sensors are suitable to infer gestures, but they might not be accurate enough to detect micro finger gestures or they instrument the fingers with extra sensors, which might suffer from the uncomfortable user experience. Thermal infrared signals radiating from fingers enable to capture subtle finger movements for micro gestures [22], however, its accuracy is affected by varied surrounding temperatures.

III. Preliminaries

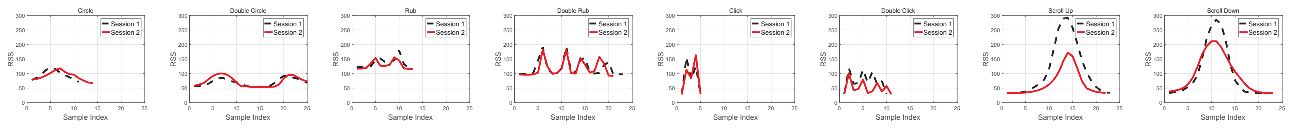
A. NIR Sensing

We leverage harmless NIR, whose wavelength is from $740nm$ to $1400nm$ and invisible to human eyes. It has identical



(a) Circle (b) Double Circle (c) Rub (d) Double Rub (e) Click (f) Double Click (g) Scroll Up (h) Scroll Down

Fig. 2. Gesture set.



(a) Circle (b) Double Circle (c) Rub (d) Double Rub (e) Click (f) Double Click (g) Scroll Up (h) Scroll Down

Fig. 3. Characteristic RSS readings of gestures.

characteristics to visible light, such as light reflection and refraction. If setting one NIR LED and conducting a micro finger gesture in close proximity towards the LED, as human skin can only absorb a tiny amount of NIR [36], most of the emitted NIR will be reflected by the fingers. Then, the highly sensitive NIR PDs enable to capture such subtle light changes caused by the gesture and convert the RSS into electrical signals. Because the fingers of a user move uniquely when performing gestures and thus generate unique patterns in the time-series RSS, we are motivated to use NIR LEDs and PDs to recognize micro finger gestures.

B. Gesture Set

To design a gesture set, we first explore RSS readings using one NIR LED (304IRC-94, $940nm$, $20°$) and one NIR PD (304PT, $700\text{-}1,000nm$, $80°$) located side by side. One right-handed volunteer is asked to perform a large number of candidate gestures. All gestures are performed twice, denoted as two sessions. Then, we observe the RSS readings and select eight common and intuitive micro finger gestures that have their unique RSS patterns in each session and present consistent RSS patterns among the two sessions. The gestures and their corresponding RSS readings are shown in Fig. 2 and Fig. 3, respectively.

According to their recognition requirement, the gesture set includes two types: *Detect-aimed gestures* and *Track-aimed gestures*. *Detect-aimed gestures* contain circle, double circle, rub, and double rub (Fig. 2 (a)-(d)), which are performed by moving the thumb-tip against the tip of the index finger, like drawing a picture. Additionally, to satisfy the need to interact with smart devices using relatively traditional gestures, detect-aimed gestures also contain click and double click (Fig. 2 (e), (f)), resembling clicking/pressing on a touchscreen or double-clicking using a mouse. The gestures of click and double click can be performed in the air or against the hand back. Comparing to detect-aimed gestures that need to be detected, *Track-aimed gestures* also need to be synchronously tracked in terms of finger movements direction, velocity, and displacement. It contains scroll up and scroll down (Fig. 2 (g), (h)), simulating scroll motions, such as sliding up and down to browse information on a webpage.

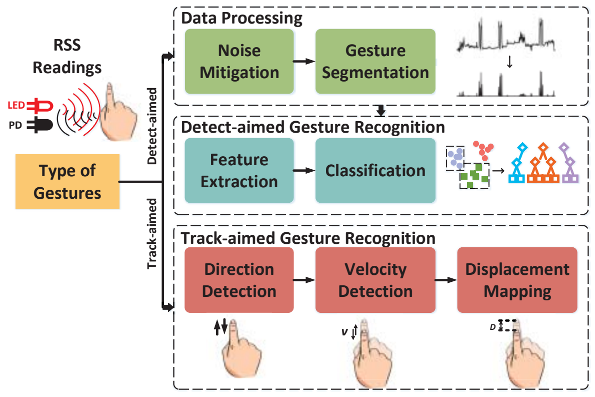


Fig. 4. Overview of airFinger.

IV. Design of airFinger

A. Overview

The proposed airFinger includes three major parts, *Data Processing*, *Detect-aimed Gesture Recognition*, and *Track-aimed Gesture Recognition*, which is shown in Fig. 4. In *Data Processing* (Section IV-B), we design a Square Based Calculation (SBC) algorithm, whose advantages are two-fold. First, it can effectively mitigate noise, such as reflected NIR by surrounding objects, hardware noise from the device itself, and varied sunlight intensities. Second, the results of SBC can help to detect the starting and ending points of a gesture. Additionally, we design a Dynamic Threshold (DT) algorithm to segment gestures, which can automatically and dynamically compute a threshold to find starting and ending points of a gesture under different scenarios.

Detect-aimed Gesture Recognition (Section IV-C) aims to recognize micro finger gestures by extracting unique features. Traditional methods are based on observing and leveraging standard features, such as maximum peak and mean amplitude, which might be poor in robustness. Instead, we use a toolbox to firstly extract a large number of features from the results of *Data Processing*. Then we utilize feature feedback from a random forest classifier to rank features by their contributions to classification. Next, we select the top 25 features from the importance ranking of feedback.

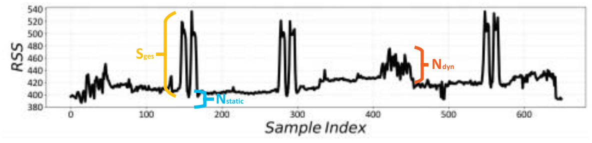
Track-aimed Gesture Recognition (Section IV-D) aims to track fingers synchronously while performing track-aimed gestures. RSS readings caused by track-aimed gestures exhibit unique patterns comparing with other gestures, however, the uniqueness cannot be directly applied to infer scroll direction, velocity, and displacement. To track fingers, we build a custom NIR-based sensor with alternating NIR LEDs and PDs. The sensor has low power consumption, reasonable price, and small size (retailing $0.2 and $3mm$ in diameter of each NIR LED or PD). Additionally, we design a ZEBRA algorithm, which enables airFinger to track finger direction, velocity, and displacement of the scrolling.

Because track-aimed gestures need to be tracked beside detection, which is different from the detect-aimed gestures, so they are handled differently. We design an algorithm to distinguish these two types of gestures (Section IV-E). Additionally, because recognition accuracy of airFinger is affected by unintentional finger movings (non-gestures), such as scratching, we also propose an algorithm to effectively remove such interferences (Section IV-F).

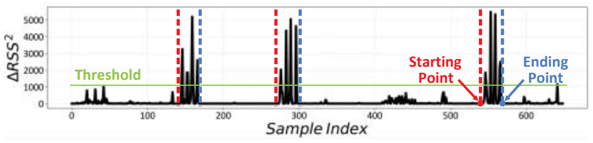
B. Data Processing

1) Noise Mitigation: Technically, RSS readings are only affected by moving fingers of gestures, denoted as S_{ges}. However, besides sudden RSS changes due to hardware, in practice, the other parts of the hand will also reflect the NIR lights, which results in relatively static noise of the RSS readings, denoted as N_{static}. Additionally, the RSS readings are also affected by relatively dynamic noise, denoted as N_{dyn}, which might be from two aspects. First, except the emitted NIR, other NIR sources, such as sunlight, are affected along with the finger movements. Second, surrounding objects will also reflect the NIR to the NIR PDs. So the RSS readings can be formulated as $RSS = S_{ges} + N_{static} + N_{dyn}$. Because of N_{static} and N_{dyn}, although the RSS readings surely capture subtle finger motions, the original RSS is not suitable to be directly used to recognize gestures.

Embracing this issue, firstly, we add a 3D-printed black shield to limit Field-of-View (FoV) of the PDs, which greatly reduces the effect of noise. Secondly, we propose a Square Based Calculation (SBC) algorithm to eliminate the noise. It sets a sliding window of size w to check real-time RSS readings, in which it subtracts values of the current window by the previous and squares the magnitudes (ΔRSS^2). Then N_{static} will be removed. Additionally, as N_{dyn} has a lower magnitude comparing to S_{ges}, N_{dyn} will be relatively mitigated, while S_{ges} will be relatively enhanced. Therefore, the SBC algorithm benefits to mitigate noise and strengthen unique patterns in



(a) Original RSS readings.



(b) RSS readings after SBC and DT algorithms.

Fig. 5. Results of SBC, DT algorithms to mitigate noise and segment gestures.

RSS readings. Besides, it is simple and efficient with $\mathcal{O}(n)$ time complexity.

2) Gesture Segmentation: We observe that RSS values are relatively stable when no gesture is performed and there exist significant changes when a gesture is performed, which helps to segment gestures. After the process of SBC, this observation will be more obvious. Therefore, based on the results of SBC, we could apply a threshold to find the starting and ending points of a gesture. However, a fixed threshold cannot work, because different positions of fingers (*e.g.* distances from fingers to the sensors) will change the range of ΔRSS^2 values. We note that the optimal sensing distance will be discussed in Section V-D. To solve this issue, we design a Dynamic Threshold (DT) algorithm to automatically and dynamically segment gestures. This algorithm is inspired by the problem of background and foreground segmentation in computer vision [37]. We denote a set of m signals of ΔRSS^2 as $S = \{r_1, r_2, ..., r_m\}$. Given a threshold I_{seg}, S can be divided into two classes G and NG representing gestures and non-gestures, respectively:

$$\begin{aligned} G &= \{r_i | r_i > I_{seg}, r_i \in S\}, \\ NG &= \{r_i | r_i <= I_{seg}, r_i \in S\}. \end{aligned} \tag{1}$$

We set ω_0 and ω_1 as probabilities of the two classes separated by I_{seg}. They are computed by:

$$\omega_0 = \frac{|G|}{m}, \quad \omega_1 = \frac{|NG|}{m}. \tag{2}$$

Mean value of each class is denoted as μ_0 and μ_1:

$$\mu_0 = \frac{\sum\limits_{r_i \in G} r_i}{|G|}, \quad \mu_1 = \frac{\sum\limits_{r_i \in NG} r_i}{|NG|}. \tag{3}$$

Then, we iteratively calculate a threshold I_{seg} that maximizes inter-class variance:

$$I_{seg} = \arg\max_{I_{seg}} \omega_0 \omega_1 (\mu_0 - \mu_1)^2. \tag{4}$$

Thus, given an initial threshold, *e.g.*, $I'_{seg} = 10$, along with accumulated RSS readings, we can calibrate and update the

TABLE I
SELECTED FEATURES.

Category	Features
Time Domain	**Standard Deviation**, Variance, Count below/above mean, Last location of maximum, Partial autocorrelation, First location of minimum/maximum, Sample entropy, **Longest strike above/below mean**, **Kurtosis**, AR, Autocorrelation, **Number of peaks**, Quantile, Complexity-invariant distance [39], Mean absolute change, Time reversal asymmetry statistic, Absolute energy, Energy ratio by chunks, Approximate entropy, **Length**, **Linear trend**, Augmented dickey fuller, c3 [40].
Frequency Domain	**Fast Fourier Transform**, **Continuous Wavelet transform**.

$*$ Selected features used to remove other interferences in **bold**.

threshold I_{seg}. Then, to detect a starting point, DT continuously compares ΔRSS^2 with the dynamically computed threshold I_{seg}. If $r_i \in S$ exceeds I_{seg}, it estimates a starting point at time i; if $r_i \in S$ is below I_{seg}, an ending point is detected at time i. Additionally, if two segments are separated by less than time t_e, we cluster them into a single gesture. The results of SBC and DT algorithms are shown in Fig. 5, which demonstrates the effectiveness of our algorithms to mitigate noise and segment gestures.

C. Detect-aimed Gesture Recognition

1) Feature Extraction: Like any machine learning-based application, feature extraction is critical to the success of airFinger. Here, the feature extraction has two issues. First, we find that only using standard features, such as the number of peaks, derivative, or skewness, results in low recognition accuracy. Second, due to individual diversity and gesture inconsistency, ΔRSS^2 values are affected by how a gesture is performed, such as finger moving speed, finger position and pointing direction. Thus, features based on specific RSS values are not appropriate for classification.

To obtain distinguishing and consistency features, firstly we use a toolbox tsfresh [38] to automatically extract a large number of candidate features. If using all these features with a limited amount of training data, the problem of over-fitting might happen. Thus, we want to select a set of features from the candidate features. We apply a Random Forest (RF)-based classifier to rank these features by their importance feedback. Next, we combine signal observation and feature importance to select 25 kinds of features from both time and frequency domain, which are listed in Table I. These features are most relevant to micro finger gestures and independent on RSS effects from noise interferences, individual diversity, and gesture inconsistency, which derives a high recognition accuracy. Therefore, we reduce the amount of data needed for training and improve classification accuracy.

2) Classification: Recording RSS readings when users are performing gestures, airFinger recognizes the gestures in real-time using the selected features above. We apply an RF-based classifier to recognize micro finger gestures because several works have shown that RF can perform well to classify large amounts of data regarding accuracy, robustness, and scalability [22]. Besides, comparing to Hidden Markov Models (HMM), Dynamic Time Warping (DTW), and Convolutional Neural Networks (CNN), RF has lower computational expense, which is more suitable for real-time gesture recognition on wearable smart devices [3]. Moreover, we compare its recognition accuracy with other light-weighted classifiers including Logistic Regression (LR), Decision Trees (DT), and Bernoulli Naive Bayes (BNB). We find that the RF-based classifier has the best performance, which is described in Section V-E.

D. Track-aimed Gesture Recognition

To track a scrolling motion, we need to know its direction, velocity, and displacement while a user is scrolling by finger to operate a screen. To realize this, finger localization is an intuitive method. Although Infrared Radiation (IR) proximity sensors have been widely used to measure distances, such as lidar or sensor unit GP2Y0A21YK0F of SHARP, they cannot measure micro finger gestures, because the measurement scope is larger than $10cm$ [41].

To solve this issue, firstly we design a special NIR-based sensor consisting of NIR LEDs and PDs altering located close to each other and side by side. For example, as shown in Fig. 6, two NIR LEDs (L_1, L_2) and three NIR PDs (P_1, P_2, P_3) are in the interval distributions. The FoV of NIR LEDs and PDs are $20°$ and $80°$, respectively. Their irradiation scopes and sensing scopes are denoted as IL_1, IL_2, and SP_1, SP_2, SP_3, respectively. Then, we design a ZEBRA algorithm to track finger locations.

1) Direction: When a finger is posited in IL_1, it reflects NIR to P_1 and P_2. Thus, it will increase the signal values of P_1 and P_2. Similarly, when the finger is posited in IL_2, it reflects NIR to P_2 and P_3 and increases signal values of P_2 and P_3. Within a small FoV, we assume that P_1 will not receive the reflected NIR from IL_2, and vice versa. Thus, if P_1 has earlier signal ascending than P_3, the gesture is regarded as scroll up. Otherwise, it is a scroll down. Two typical waves of scroll up and scroll down are shown in Fig. 7. Note that, to determine the orders of signal ascending of PDs, we use the SBC algorithm (Section IV-B) to find ascending points and ending points of gestures.

However, sometimes, users do not scroll completely between P_1 and P_3. They might scroll up only passing P_1. Under this condition, airFinger will only detect a signal ascending point of P_1 during the entire gesture duration, which also infers that the signal ascending point of P_1 happens before P_3. So, we still regard this gesture as scroll up. Similarly, if users scroll down only passing P_3, which results in only detecting a signal ascending point of P_3 during the entire gesture duration, the gesture will be regarded as scroll down.

Therefore, we can identify scroll direction based on the different orders of signal ascending points in real-time, without waiting for the end of this gesture.

2) Velocity & Displacement: To track scroll velocity, as the physical distance between P_1 and P_3 is fixed, the velocity

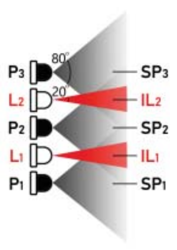
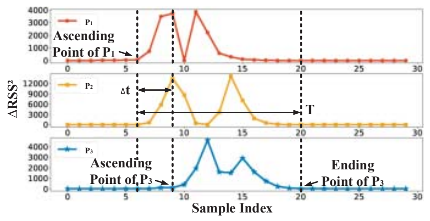
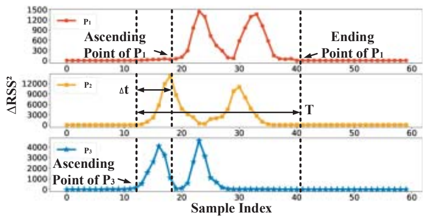


Fig. 6. Principle of a NIR-based sensor.



(a) Scrolling from P_1 to P_3, signal ascending of P_1 occurs before P_3.



(b) Scrolling from P_3 to P_1, signal ascending of P_3 occurs before P_1.

Fig. 7. Signals of track-aimed gestures. Δt denotes time difference between signal ascending of P_1 and P_3. T denotes the total duration time of a track-aimed gestures.

is proportional to the time difference Δt between signal ascending of P_1 and P_3, which is denoted as $v(\Delta t)$. We also assume that every user scrolls at a constant velocity. Then scroll displacement D_t during time t of a track-aimed gesture is mapped as:

$$D_t = \alpha \cdot v(\Delta t) \cdot \min\{t, T\}, \tag{5}$$

where T is the total duration time of a track-aimed gesture, and α is scroll direction ($\alpha = 1$ denotes scroll up and $\alpha = -1$ denotes scroll down).

Therefore, we can track scroll velocity and displacement using the designed special sensor and we note that the displacement can map to different scales according to different application demands.

3) Summary: We list the three conditions of ZEBRA to recognize direction, velocity, and displacement in terms of scroll up in the following. The recognition for scroll down is vice versa.

- When a user scrolls just through IL_1, we use the orders of signal ascending points to determine α. Since there are no signal changes in P_3, Δt is incalculable. We assign a velocity v' according to experience. Then $D_t = \alpha \cdot v' \cdot \min\{t, T\}$.
- When a user scrolls through IL_1 and IL_2, we use the orders of signal ascending points to determine α. Then we get $v(\Delta t)$ from Δt and T from the total scroll time. Then $D_t = \alpha \cdot v(\Delta t) \cdot \min\{t, T\}$.
- When a user scrolls through IL_1 and IL_2 and passes the sensing space, α is determined by the orders of signal ascending points. $v(\Delta t)$ is determined by Δt. Since the finger is out of sensing space, we use T to compute the whole scroll duration. Then $D_t = \alpha \cdot v(\Delta t) \cdot \min\{t, T\}$.

The detailed rules in ZEBRA are described in Alg. 1. When only finding an ascending point of P_1, we set this gesture as scroll up. The velocity $v(\Delta t)$ during time t is set according to experience v' and the displacement is $D_t = \alpha \cdot v' \cdot \min\{t, T\}$ (lines 2-7). When finding an ascending point of P_1 before P_3, we still set this gesture as scroll up. The velocity $v(\Delta t)$ during time t is proportional to Δt and the displacement is $D_t = \alpha \cdot v(\Delta t) \cdot \min\{t, T\}$ (lines 8-13). Similarly, when only finding an ascending point of P_3, we set this gesture as scroll down. The velocity $v(\Delta t)$ during time t is set according to experience v' and the displacement is $D_t = \alpha \cdot v' \cdot \min\{t, T\}$ (lines 14-19). When finding an ascending point of P_3 before P_1, we still set this gesture as scroll down. The velocity $v(\Delta t)$ during time t is proportional to Δt and the displacement is $D_t = \alpha \cdot v(\Delta t) \cdot \min\{t, T\}$ (lines 20-25).

Overall, ZEBRA enables to simulate track-aimed gestures in terms of direction, velocity, and displacement in real-time with low computation and low energy costs.

ALGORITHM 1: ZEBRA Algorithm.

```
Input: RSS signals $S$.
Output: scroll direction $\alpha$, velocity $v(\Delta t)$ and scroll
        displacement $D_t$ during time $t$.
1  Find signal ascending point using SBC algorithm;
2  if only $P_1$ has an ascending point, $P_3$ has no
     ascending point;
3   then
4      detect a scroll up gesture: $\alpha$=1;
5      set velocity according to experience $v'$:
         $v(\Delta t) = v'$;
6      $D_t = \alpha \cdot v' \cdot \min\{t, T\}$;
7  end
8  if $P_1$ has an ascending point before $P_3$;
9   then
10     detect a scroll up gesture: $\alpha$=1;
11     velocity is proportional to $\Delta t$: $v(\Delta t) = \Delta t$;
12     $D_t = \alpha \cdot v(\Delta t) \cdot \min\{t, T\}$;
13 end
14 if only $P_3$ has an ascending point, $P_1$ has no
     ascending point;
15  then
16     detect a scroll down gesture: $\alpha$=-1;
17     set velocity according to experience $v'$:
         $v(\Delta t) = v'$;
18     $D_t = \alpha \cdot v' \cdot \min\{t, T\}$;
19 end
20 if $P_3$ has an ascending point before $P_1$;
21  then
22     detect a scroll down gesture: $\alpha$=-1;
23     velocity is proportional to $\Delta t$: $v(\Delta t) = \Delta t$;
24     $D_t = \alpha \cdot v(\Delta t) \cdot \min\{t, T\}$;
25 end
```

E. Distinguish Detect-aimed and Track-aimed Gestures

As detect-aimed gestures and track-aimed gestures have different recognition algorithms, we need to distinguish them at the beginning of gesture performing. We observe that when performing a detect-aimed gesture, signal ascending points from all PDs almost occur simultaneously. However, when performing a track-aimed gesture, signal ascending points from all PDs occur in orders. So we set that when the time difference between signal ascending points is less than a threshold I_g, we regard the current gesture as a detect-aimed gesture. Otherwise, it is regarded as a track-aimed gesture. This algorithm is light-weighted in computing cost and response time, which can effectively distinguish gestures at the beginning of gesture performing. The performance of this algorithm is shown in Section V-I.

F. Remove Other Interferences

RSS readings are affected by many other interferences, such as a user unintentional moving fingers rather than performing a gesture or sudden RSS changes due to hardware noise. Because the interferences also cause significant changes in RSS readings like what a gesture does, they can be falsely segmented as a detect-aimed gesture. To remove such interferences, we design a machine learning-based model to identify gestures and non-gestures. Similarly, we also utilize tsfresh [38] to extract features firstly, then we select 9 kinds of features that are ranked as the most effective and relevant features by RF (listed in Table I in bold). We also compare the performance of RF with LR, DT, and BNB. The recognition accuracy of RF is the highest. So we apply an RF-based classifier to identify gestures and non-gestures. Note that since these features to remove the interferences are also extracted to recognize detect-aimed gestures, they can be saved and reused later. Thus, this algorithm can reduce inevitable interferences without extra consumption burden. The performance of this method is shown in Section V-J.

V. Evaluation

A. Implementation and Experiment Setup

We build a prototype of airFinger that contains two NIR LEDs (304IRC-94, 940nm, 20°), three NIR PDs (304PT, 700-1,000nm, 80°), alternatively located close to each other, and a 3D-printed black shield, which is shown in Fig. 1. The LEDs and PDs are fixed to face the same side and their diameters are both 3mm. In the experiments, we use amplifiers and a Micro Controller Unit (MCU) Arduino UNO to measure RSS readings of the NIR PDs at 100Hz. The total power consumed by the PDs and LEDs is highly efficient, 24mW excluding the consumption of microcontroller. The window size w and parameter t_e for clustering gestures are set to 10ms and 100ms, respectively. The threshold I_g used to distinguish detect-aimed gestures and track-aimed gestures is set to 30ms. These settings are learned from the collected samples in the data collection in the following.

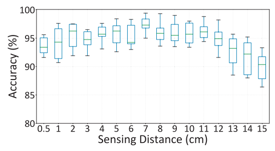


Fig. 8. Accuracy of different sensing distances between user fingers and the sensor.

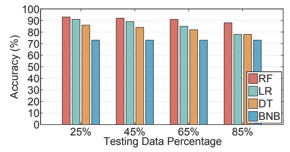
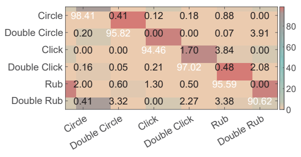
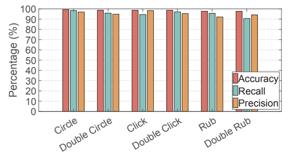


Fig. 9. Accuracy comparison between four classifiers with different percentage of testing data.



(a) Confusion matrix.



(b) Accuracy, recall, and precision.

Fig. 10. Overall performance of detect-aimed gestures among 10 volunteers.

B. Data Collection

We have 10 volunteers (4 males and 6 females), aging from 20 to 49 ($avg = 25.7$). Everyone is healthy and right-handed. They all attend an orientation that teaches how to perform the designed eight gestures shown in Fig. 2. Note that, volunteers perform gestures according to their habits, without given any instructions (*e.g.* magnitude or duration).

The data collection process has 5 sessions. During each session, every volunteer is asked to sit in a comfortable posture and perform gestures in close proximity to the prototype. Each gesture is repeated 25 times. Between sessions, every volunteer takes a five minutes break such as standing up and relaxing arms. The process lasts about 1-2 hours for each volunteer. Additionally, as an incentive to participate in the experiment, we have paid all volunteers by monetary compensation. We totally collect 10,000 samples (10 people $*$ 8 gestures $*$ 5 sessions $*$ 25 times) with manually gesture labels.

C. Performance Metrics

Confusion Matrix: Each row and each column represents the ground truth and the predicted result, respectively. The i^{th}-row and j^{th}-column entry of the matrix is defined as the ratio of the total number of samples that are classified as the j^{th} gesture while actually are the i^{th} gesture divide to the total number of samples that are the i^{th} gesture.

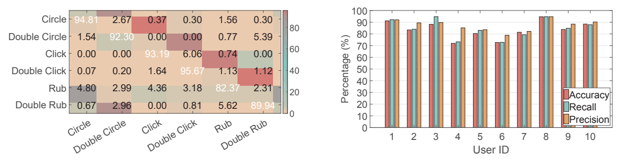
Accuracy: The ratio of the total number of samples that are correctly classified divide to the total number of classified samples made by the classifier.

Recall: The ratio of the samples that are correctly recognized as label g among all the samples with label g.

Precision: The ratio of the samples that are correctly recognized as label g among all the samples that are recognized as label g.

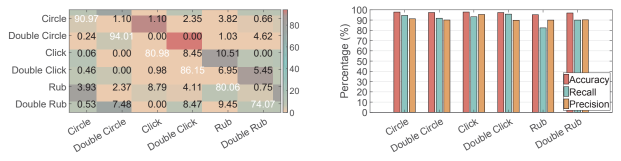
D. Study of Sensing Distance

We study the optimal sensing distance from users' fingers to the prototype of airFinger. Three volunteers are asked to



(a) Confusion matrix. (b) Accuracy, recall, and precision.

Fig. 11. Impact of individual diversity of detect-aimed gestures among different people.



(a) Confusion matrix. (b) Accuracy, recall, and precision.

Fig. 12. Impact of gesture inconsistency of detect-aimed gestures among different sessions.

conduct the designed eight gestures in close proximity to the prototype at different distances from $0.5cm$ to $12cm$ with an increment of $0.5cm$. We find that the optimal sensing distance is about $0.5cm$ to $6cm$, in which the accuracy of airFinger achieves above 90%. We note that this sensing distance can reduce noise disturbance and provides an unobtrusive and natural distance for users to interact with smart devices by micro finger gestures [32].

E. *Performance of RF-based Classifier*

To show the effectiveness of the RF-based classifier, we compare its accuracy with three other commonly used classifiers including LR, DT, and BNB. All these classifiers use default parameters. We use all the collected gesture samples and vary the percentage of testing data. The comparison results are shown in Fig. 9.

We can see that with the increasing percentage of testing data, all accuracies of the classifiers slightly decrease, while the RF-based classifier has the highest accuracy. When the testing data is 25% of the whole data set, the RF-based classifier reaches the highest accuracy. Although LR also performs not bad, its computing time is much longer than that of RF. Therefore, the RF-based classifier outperforms all the other tested classifiers and has high efficiency for the classification of multi-dimensional features.

F. *Detect-aimed Gesture Evaluation*

1) Overall Performance: To show the overall performance of airFinger to recognize six detect-aimed gestures among 10 volunteers, we conduct a five cross-validation (leave-one-out-cross-validation) over all collected samples. The overall results are calculated by averaging the results from all combinations of training and testing data. The average accuracy is 98.44%, which indicates that airFinger enables to correctly recognize the detect-aimed gestures.

The confusion matrix of the overall performance is shown in Fig. 10 (a). We can see that averagely above 90% of gestures are correctly recognized. Circle shows the best recognition results, 98.41% of them correctly identified. Additionally, as shown in Fig. 10 (b), all gestures can be recognized well with the lowest average recall and precision of 90.65% and 92.13% (both above 90%), respectively. Thus airFinger has a promising overall performance to recognize the detect-aimed gestures among 10 volunteers.

2) Impact of Individual Diversity: To prove the robustness of airFinger against individual diversity, we use samples of nine users as training data and samples of the remaining one user as testing data. The results are calculated by averaging the results from all ten combinations of training and testing data. The average accuracy is 83.61%, which indicates that airFinger has robustness among different people.

Fig. 11 (a) shows the confusion matrix of six detect-aimed gestures. Double click gestures have the best performance, 95.67% of them correctly recognized. Fig. 11 (b) shows the accuracy, recall, and precision of 10 volunteers and 80% of them can reach an accuracy above 80%. The average precision and recall of the 10 volunteers are 84.69% and 87.44%, respectively. While volunteer 4 and 6 have relatively low accuracies due to false detection of click and rub. It might because these gestures are not highly user-friendly for them, but their other gestures still achieve high accuracies. So airFinger is resilient against individual diversity. Additionally, these results demonstrate that we can pre-train the classifier and then people can directly work with airFinger without user-specific calibration.

3) Impact of Gesture Inconsistency: To prove the robustness of airFinger against individual inconsistency, we use 4 sessions of each user as training data and the remaining one session as testing data. The results of all combinations are averaged. The average accuracy of all the detect-aimed gestures is 97.07%, which indicates gestures from different sessions can still be correctly recognized.

In Fig. 12 (a), the confusion matrix is shown for recognition of the detect-aimed gestures. Double circle has the best performance, 94.01% of them correctly recognized. In Fig. 12 (b), all gestures can reach accuracies above 95%. The average recall and precision are 91.28% and 91.11%, respectively. Rub and double rub have relatively low recall and precision. The reason is that some gestures of double rub of volunteer 3 are recognized as gestures of rub. It might because he/she performs slowly so that gestures of double rub are segmented into two gestures of rub. Overall, airFinger is resilient against gesture inconsistency and a pre-trained classifier enables users to conduct gestures without pre-setup before each use.

G. *Track-aimed Gesture Evaluation*

We test the recognition results of scroll directions. The average accuracy of scroll up is 99.88% and the average accuracy of scroll down is 99.26%. Thus airFinger performs well to recognize directions of the track-aimed gestures.

Additionally, we implement a real-time interface of gesture tracking on a tablet and set $v' = 80mm/s$ according to experience. The interface displays some news with words and

TABLE II
PERFORMANCE SUMMARY.

Category		Accuracy
Detect-aimed Gestures	Circle	99.26%
	Double circle	98.72%
	Click	98.65%
	Double click	98.68%
	Rub	97.69%
	Double rub	97.62%
	Average accuracy = 98.44%	
Track-aimed Gestures	Scroll up direction	99.88%
	Scroll down direction	99.26%
	Average accuracy = 99.57%	
Rate of scroll velocity & displacement		2.6/3.0
Summary average accuracy = 98.72%		

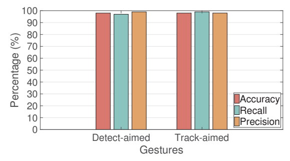


Fig. 13. Performance of distinguishing two kinds of gestures.

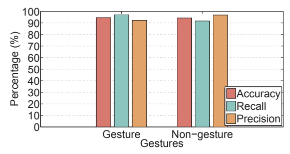
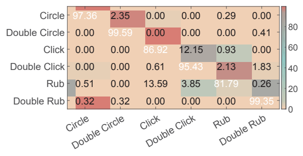
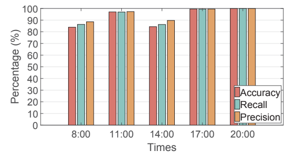


Fig. 14. Impact of unintentional motions.



(a) Confusion matrix.



(b) Accuracy, recall, and precision.

Fig. 15. Impact of environmental light changes of detect-aimed gestures.

pictures. When a volunteer is watching the news by the track-aimed gestures, we use a bar mapping to the track-aimed gestures along with their direction, velocity, and displacement. All volunteers are asked to rate the interface by a score of 1 to 3, which represents noticeable un-matched scrolling, standard, and fluent matched scrolling, respectively. The average score is 2.6 and 90% of users do not feel un-matching scrolling when using our interface, which confirms the performance of our technique. Thus, airFinger reaches the goal to synchronously recognize the track-aimed gestures in terms of direction, velocity, and displacement.

H. Performance Summary

The above experiment has proved airFinger's ability to recognize the detect-aimed gestures and track-aimed gestures. As shown in Table II, for the six detect-aimed gestures, the average accuracy of all samples is 98.44%. For the two track-aimed gestures, the average accuracy of scroll direction is 99.57%. Thus, the accuracy of our system for all eight gestures is 98.72%. Overall, airFinger is robust to different environments and when faced with different users, it does not need to be re-trained to achieve good performance.

I. Performance of Distinguishing Gestures

We evaluate the designed algorithm to distinguish the detect-aimed gestures and track-aimed gestures with all the collected samples. The results are shown in Fig. 13. The accuracy, recall, and precision of our algorithm are all above 98%. The results demonstrate that the two kinds of gestures can be correctly distinguished, which provides a sound foundation to recognize the designed micro finger gestures.

J. Other Impacts

1) Unintentional Motions: We also test if airFinger is resilient to unintentional motions. When performing gestures, users may unintentionally move hand or fingers occurring non-gestures. To mimic the unintentional motions, six volunteers are asked to perform gestures and non-gestures over two sessions. During each session, every volunteer is asked to perform 25 gestures and 25 non-gestures like scratching, extending, or reposition hands and fingers. Between sessions, volunteers are asked to take a break. The volunteers totally perform 300 unintentional motions and 300 designed gestures over the two sessions. We label the performed gestures and compare them with the recognition results. We conduct a three cross-validation (leave-one-out-cross-validation) over all these samples and average the results from all combinations of training and testing data.

It is encouraging to find that airFinger can achieve an average accuracy of 94.83%, which is shown in Fig. 14. The average recall and precision are 94.83% and 94.88%, respectively. The results demonstrate that the designed algorithm has high effectiveness to remove unintentional motions.

2) Environmental NIR Changes: To evaluate the impact of environmental NIR changes, we conduct experiments from 8 to 20 o'clock every 3 hours in one day, which represent differently environmental NIR conditions. Two volunteers are asked to perform all the designed gestures under the 5 different times. Each gesture is performed 25 times.

As shown in Fig. 15 (a), above 80% of gestures are correctly recognized and even over 99% of double circle and double rub are correctly recognized. Fig. 15 (b) shows the accuracy, recall, and precision under the 6 different times. The average accuracy is 92.97%. The average recall and precision are 93.8% and 95.02%, respectively. Thus, we show that airFinger is resilient against environmental NIR changes.

3) Dominant Hand Influence: We validate the impact of using the non-dominant hand on the performance of airFinger. We have six right-handed volunteers to take part in two sessions. During each session, they perform all the designed eight gestures with their left hand (non-domain hand) 20 times. We conduct a three cross-validation (leave-one-out-cross-validation) over all these samples and average the results from all combinations of training and testing data. The prototype is also oriented accordingly.

As shown in Fig. 16, the average accuracy of gestures performed by non-dominant hand is over 95%, only slightly lower than that performed by the dominant hand. The average

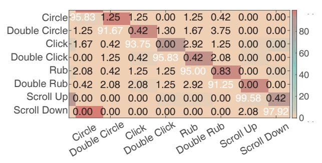


Fig. 16. Impact of dominant hand.

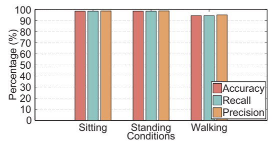


Fig. 17. Performance of a demo within a wristband.

recall and precision are 95.10% and 95.13%, respectively. Thus, we show that using the non-dominant hand does not affect the performance of airFinger.

4) Other Human Interferences: We test if airFinger is resilient to human interferences. When a volunteer is performing gestures, another volunteer is asked to move around, such as passing by or waving arms. We find that other human moving around does not affect the accuracy of airFinger. We analyze the reasons in two aspects. First, the designed NIR-based sensor is optimized to be sensitive to the range from $0.5cm$ to $6cm$ proximity to itself. The effect of other human moving around is out of the sensing range of the sensor, which cannot affect the performance of airFinger. Second, we have designed SBC and TD algorithms to mitigate noise, which is beneficial to reduce the effects of human interferences.

Additionally, we also ask a volunteer to use an IR remote control around when another volunteer is performing gestures. When it is directly pointed to the sensors, the remote control will cause recognition errors, however, this kind of situation is rare in practice. So commonly non-directly-pointed operations will not have an impact on airFinger.

K. Demo within a Wristband

To evaluate if airFinger can perform well under real wristband scenarios, we augment our prototype with a wristband and implement a Bluetooth module to send signals to a laptop. The band can be worn on wrists and recognize micro finger gestures in proximity to it. We test it with six right-handed volunteers and all of them wear it on their left hand and conduct gestures by the right hand in three common usage conditions including sitting, standing, and walking. Under each condition, every volunteer repeatedly performs all the designed gestures 25 times. The built wristband and experiment scenarios are shown in Fig. 18.

The results are shown in Fig. 17. The averaged accuracy, recall, and precision are 97.17%, 97.17%, and 97.46%, respectively. Thus our technology ensures great performance for practical usage on wristbands while sitting, standing, and walking. It can build robust interaction between users and their wrist-worn smart devices via micro finger gestures.

VI. Discussion and Future Work

Outdoors Situation: As the sunlight contains a large amount of NIR, the PDs of airFinger might be up into the saturation region under the high intensity of sunlight outdoors. To solve this issue, we plan to optimize hardware design to be workable under different light intensities via frequency modulation, high sample rate, and adjustable amplifiers.

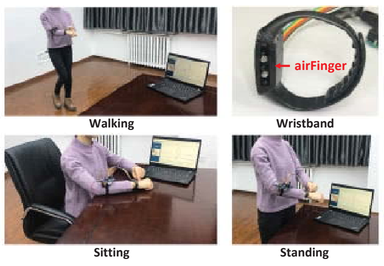


Fig. 18. Demo of a wristband. One volunteer performs gestures while sitting, standing, and walking.

Gesture Set: We would like to build a sensor with more number of LEDs and PDs along with other posited distributions to construct a multi-dimensional sensing area and improve input resolution, which enables to expand the gesture set so as to realize more applications. Additionally, it is an interesting option to enable user-self-defined gestures. Users might be willing to define customized gestures on their own. Like personalized icons, customized gestures can provide more space for users to interact with their smart devices and somehow preserve both personality and privacy.

Energy and Storage: Although NIR LEDs and PDs are cost-effective, we could optimize hardware design and recognition algorithms to further reduce power-consuming and storage-cost. Additionally, it can combine with a Bluetooth or Wi-Fi module to pass data for processing to a cloud server or an edge computing terminal, which will extend its computing ability and reduce consumption burden on local devices.

VII. Conclusion

We design airFinger to recognize micro finger gestures. Our technology exploits NIR light sensing to capture subtle finger movements to identify gestures and track finger movements in real-time. We build a prototype of airFinger and conduct extensive experiments with 10 volunteers under different scenarios. Its overall accuracy achieves 98.72% among 8 micro finger gestures and it has resilience against individual diversity, gesture inconsistency, and many other impacts.

Our proposed solution, airFinger, made the first step to develop an approach based on NIR light sensing to detect micro finger gestures and track fingers with energy and processing efficiency. As its target is micro finger gestures, airFinger can expand working space for wearable devices and avoid blocking the line of sight when using a tiny screen device. Additionally, it has a small size, low cost, and can be applied to different applications for interaction with ubiquitous smart devices.

References

[1] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," in *Industrial Technology and Vocational Education*, vol. 12, no. 2, 2012, p. 1437.

[2] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, and P. Olivier, "Digits: Freehand 3D interactions anywhere using a wrist-worn gloveless sensor," in *ACM symposium on User interface software and technology, UIST*, 2012.

[3] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, "Soli: Ubiquitous gesture sensing with millimeter wave radar," in *ACM Transactions on Graphics, TOG*, vol. 35, no. 4, 2016, p. 142.

[4] Z. Yang, Z. Zhou, and Y. Liu, "From RSSI to CSI: Indoor localization via channel response," in *ACM Computing Surveys*, vol. 46, no. 2, 2013.

[5] H. Abdelnasser, M. Youssef, and K. A. Harras, "WiGest: A ubiquitous WiFi-based gesture recognition system," in *IEEE International Conference on Computer Communications, INFOCOM*, 2015, pp. 1472–1480.

[6] S. Tan and J. Yang, "WiFinger: Leveraging commodity WiFi for fine-grained finger gesture recognition," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc*, 2016, pp. 201–210.

[7] Y. Zheng, Y. Zhang, K. Qian, G. Zhang, Y. Liu, C. Wu, and Z. Yang, "Zero-effort cross-domain gesture recognition with Wi-Fi," in *International Conference on Mobile Systems, Applications, and Services, MobiSys*, 2019, pp. 313–325.

[8] A. Virmani and M. Shahzad, "Position and orientation agnostic gesture recognition using WiFi," in *International Conference on Mobile Systems, Applications, and Services, MobiSys*, 2017, pp. 252–264.

[9] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: Wireless indoor localization with little human intervention," in *ACM International Conference on Mobile Computing and Networking, MobiCom*, 2012, pp. 269–280.

[10] S. Pradhan, E. Chai, K. Sundaresan, L. Qiu, M. A. Khojastepour, and S. Rangarajan, "RIO: A pervasive RFID-based touch gesture interface," in *ACM International Conference on Mobile Computing and Networking, MobiCom*, 2017, pp. 261–274.

[11] Y. Zou, J. Xiao, J. Han, K. Wu, Y. Li, and L. M. Ni, "GRfid: A device-free RFID-based gesture recognition system," in *IEEE Transactions on Mobile Computing, TMC*, vol. 16, no. 2, 2017, pp. 381–393.

[12] C. Wang, J. Liu, Y. Chen, H. Liu, L. Xie, W. Wang, B. He, and S. Lu, "Multi-touch in the air: Device-free finger tracking and gesture recognition via COTS RFID," in *IEEE International Conference on Computer Communications, INFOCOM*, 2018, pp. 1–9.

[13] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "FingerIO: using active sonar for fine-grained finger tracking," in *SIGCHI Conference on Human Factors in Computing Systems, CHI*, 2016, pp. 1515–1525.

[14] H. Chen, F. Li, and Y. Wang, "EchoTrack: Acoustic device-free hand tracking on smart phones," in *IEEE International Conference on Computer Communications, INFOCOM*, 2017, pp. 1–9.

[15] S. Yun, Y.-C. Chen, H. Zheng, L. Qiu, and W. Mao, "Strata: Fine-grained acoustic-based device-free tracking," in *International Conference on Mobile Systems, Applications, and Services, MobiSys*, 2017, pp. 15–28.

[16] W. Wang, A. X. Liu, and K. Sun, "Device-Free gesture tracking using acoustic signals," in *ACM International Conference on Mobile Computing and Networking, MobiCom*, 2016, pp. 82–94.

[17] K. Sun, T. Zhao, W. Wang, and L. Xie, "VSkin: Sensing touch gestures on surfaces of mobile devices using acoustic signals," in *ACM International Conference on Mobile Computing and Networking, MobiCom*, 2018, pp. 591–605.

[18] Z. Ren, J. Yuan, and Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," in *ACM International Conference on Multimedia, MM*, 2011, pp. 1093–1096.

[19] C. Wang, Z. Liu, and S. Chan, "Superpixel-based hand gesture recognition with kinect depth camera," in *IEEE Transactions on Multimedia*, 2015, pp. 29–39.

[20] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *IEEE International Conference on Image Processing, ICIP*, 2015, pp. 1565–1569.

[21] J. Song, G. Soros, F. Pece, S. R. Fanello, S. Izadi, C. Keskin, and O. Hilliges, "In-air gestures around unmodified mobile devices," in *ACM Symposium on User Interface Software and Technology, UIST*, 2014, pp. 319–329.

[22] J. Gong, Y. Zhang, X. Zhou, and X.-D. Yang, "Pyro: Thumb-tip gesture recognition using pyroelectric infrared sensing," in *ACM Symposium on User Interface Software and Technology, UIST*, 2017, pp. 553–563.

[23] K. Chen, K. Lyons, S. White, and S. N. Patel, "uTrack: 3D input using two magnetic sensors," in *ACM symposium on User Interface Software and Technology, UIST*, 2013, pp. 237–244.

[24] G. Reyes, J. Wu, N. Juneja, M. Goldshtein, W. K. Edwards, G. D. Abowd, and T. Starner, "SynchroWatch: One-handed synchronous smartwatch gestures using correlation and magnetic sensing," in *ACM International Conference on Ubiquitous Computing, UbiComp*, vol. 1, no. 4, 2017.

[25] C. Zhang, J. Tabor, J. Zhang, and X. Zhang, "Extending mobile interaction through near-field visible light sensing," in *ACM International Conference on Mobile Computing and Networking, MobiCom*, 2015, pp. 345–357.

[26] J. S. Kim, S. J. Yun, D. J. Seol, H. J. Park, and Y. S. Kim, "An IR proximity-based 3D motion gesture sensor for low-power portable applications," in *IEEE Sensors Journal*, vol. 15, no. 12, 2015, pp. 7009–7016.

[27] Y. Li, T. Li, R. A. Patel, X.-D. Yang, and X. Zhou, "Self-powered gesture recognition with ambient light," in *ACM symposium on User Interface Software and Technology, UIST*, 2018, pp. 595–608.

[28] A. Withana, R. Peiris, N. Samarasekara, and S. Nanayakkara, "zSense: Enabling shallow depth gesture recognition for greater input expressivity on smart wearables," in *ACM Conference on Human Factors in Computing Systems, CHI*, 2017, pp. 3661–3670.

[29] R. H. Venkatnarayan and M. Shahzad, "Gesture recognition using ambient light," in *ACM International Conference on Ubiquitous Computing, UbiComp*, vol. 2, no. 1, 2018.

[30] D. Sbirlea, M. G. Burke, S. Guarnieri, M. Pistoia, and V. Sarkar, "Automatic detection of inter-application permission leaks in android applications," in *IBM Journal of Research and Development*, vol. 57, no. 6, 2013, pp. 10:1–10:12.

[31] Z. Weinberg, E. Y. Chen, P. R. Jayaraman, and C. Jackson, "I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks," in *IEEE Symposium on Security and Privacy, S&P*, 2011, pp. 147–161.

[32] L. Chan, R. Liang, M. Tsai, K. Cheng, C. Su, M. Y. Chen, W. Cheng, and B. Chen, "Fingerpad: Private and subtle interaction using fingertips," in *ACM Symposium on User Interface Software and Technology, UIST*, 2013, pp. 255–260.

[33] C. Zhang, A. Waghmare, P. Kundra, Y. Pu, S. Gilliland, T. Ploetz, T. E. Starner, O. T. Inan, and G. D. Abowd, "FingerSound: Recognizing unistroke thumb gestures using a ring," in *ACM International Conference on Ubiquitous Computing, UbiComp*, vol. 1, no. 3, 2017.

[34] T. Zhao, J. Liu, Y. Wang, H. Liu, and Y. Chen, "PPG-based finger-level gesture recognition leveraging wearables," in *IEEE International Conference on Computer Communications, INFOCOM*, 2018, pp. 1–9.

[35] Y. Zhang, T. Gu, C. Luo, V. Kostakos, and A. Seneviratne, "Findroidhr: Smartwatch gesture input with optical heartrate monitor," in *ACM International Conference on Ubiquitous Computing, UbiComp*, vol. 2, no. 1, 2018.

[36] I. V. Meglinski and S. J. Matcher, "Quantitative assessment of skin layers absorption and skin reflectance spectra simulation in the visible and near-infrared spectral regions," in *Physiological Measurement*, vol. 23, no. 4, 2002, pp. 741–753.

[37] N. Otsu, "A threshold selection method from gray-level histograms," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, 1979, pp. 62–66.

[38] "Toolkit," 2016, http://tsfresh.readthedocs.io/en/latest/.

[39] G. E. Batista, E. J. Keogh, O. M. Tataw, and V. M. Souza, "CID: An efficient complexity-invariant distance for time series," in *Data Mining and Knowledge Discovery*, vol. 28, no. 3, 2014, pp. 634–669.

[40] T. Schreiber and A. Schmitz, "Discrimination power of measures for nonlinearity in a time series," in *Physical Review E Statistical Physics Plasmas Fluids and Related Interdisciplinary Topics*, vol. 55, no. 5, 1997, pp. 5443–5447.

[41] "Sharp distance measuring sensor unit," 2006, http://www.farnell.com /datasheets/1657845.pdf.