Implementation of Desktop at Fingertip with mmWave Short Range Radar Technology

The Direct Research Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In fulfillment of the requirements for the

Degree of Bachelor of Science

in

Electrical and Computer Engineering & Computer Science

by

Ziheng Li

Aung Khant Min

Zhenyuan Lei

An Yan

Daliang Shi

Date: May 15, 2020

APPROVED:

Professor Kaveh Pahlavan, Professor Erin Solovey

Table of Authorship

Section	Section title	Primary Author(s)	Primary Editor(s)
	Abstract	Ziheng Li	Aung
	Acknowledgement	Ziheng Li	Zhenyuan Lei
1	Introduction	Ziheng Li, Zhenyuan Lei	Ziheng Li, Zhenyuan Lei
1.1	Overview	Ziheng Li	Aung, Zhenyuan Lei
1.2	Motivation	Ziheng Li	Aung, Zhenyuan Lei
1.3	Description	Ziheng Li, Zhenyuan Lei	Aung, Ziheng Li
1.4	Summary of Proposal	Ziheng Li	Aung, Ziheng Li, Zhenyuan Lei
2	Background	Ziheng Li, Zhenyuan Lei	Aung, Ziheng Li
2.1	Radio Frequency Cloud and HCI	Ziheng Li	
2.2	Microgestures in HCI	Aung, Ziheng Li, Zhenyuan Lei	Ziheng Li
2.2.1	Wearable	Ziheng Li, Daliang, Zhenyuan Lei	Ziheng Li
2.2.2	In-air	Ziheng Li, Anyan, Zhenyuan Lei	Ziheng Li
3	Methodology	Ziheng LI, Zhenyuan Lei, Daliang shi, Aung	Ziheng LI, Zhenyuan Lei, Daliang shi, Aung
3.1	Radar Signal Processing	Ziheng Li	Zhenyuan Lei
3.1.1	Points Detection	Ziheng Li	Zhenyuan Lei
3.1.2	Signal Design for Gesture Application	Ziheng Li	Zhenyuan Lei
3.1.3	Hardware: IWR6843ISK	Zhenyuan Lei	Aung
3.1.4	Hardware: IWR6843AOP	Aung, Daliang shi	Ziheng Li
3.1.5	Range-doppler Profile	Ziheng Li	Ziheng Li
3.1.6	Range-azimuth-elevation Profile	Ziheng Li	Ziheng Li
3.1.7	Configuration	Aung, Daliang shi	Ziheng Li
3.2	DFT Gesture Paradigms	Ziheng Li, Aung	Zhenyuan Lei, Aung, An Yan
3.2.1	Indexpen	Ziheng Li	Daliang Shi
3.2.2	ThuMouse	Ziheng Li	Daliang Shi
3.3	Mode Switching Between ThuMouse and IndexPen	Zhenyuan Lei, Aung, Ziheng Li	Aung, Ziheng, Zhenyuan Lei, Anyan, Daliang
3.4	Design Considerations	Aung, Ziheng Li	Aung, Ziheng, Zhenyuan Lei, Anyan, Daliang
3.5	Gesture Sensing	Ziheng Li, An Yan	Zhenyuan Lei, Aung, Ziheng Li
3.5.1	Preprocessing	Ziheng Li	Zhenyuan Lei, Aung
3.5.2	Voxelization	Ziheng Li	Zhenyuan Lei, Aung, An yan
3.5.3	Data augmentation	Ziheng Li	Zhenyuan Lei, An yan, Aung
3.5.4	Experiment Neural Networks	Ziheng Li	Zhenyuan Lei, Ziheng Li, Aung
3.6	DFT Neural Networks, An Yan	Zhenyuan Lei, Aung, Ziheng Li	
3.7	Ground Truth Collecting	Ziheng Li, An Yan	Zhenyuan Lei, Aung, Ziheng Li
3.7.1	YOLO Detection	An yan	Ziheng Li
3.7.2	Leap Motion Detection	An yan	Ziheng Li
4	Evaluation and Results	Aung, Ziheng Li,Zhenyuan Lei,An Yan	
4.1	First Iteration	Ziheng Li, Zhenyuan Lei, An Yan	Aung
4.1.1	Experiment Environment	Ziheng Li, Zhenyuan Lei, An Yan	Aung
4.1.2	Experiment Setup	Ziheng Li, Zhenyuan Lei, An Yan	Aung
4.1.3	Results	Ziheng Li	Ziheng Li
4.2	Second Iteration	Ziheng Li, Aung	Aung, Ziheng, Zhenyuan Lei, Anyan, Daliang
4.2.1	Experiment Environment	Ziheng Li, Aung	Aung, Ziheng, Zhenyuan Lei, Anyan, Daliang
4.2.2	Experiment Setup	Aung	Zhenyuan Lei, An Yan
4.2.3	Results	Aung, Ziheng Li	Ziheng Li
4.3	Third Iteration	Ziheng Li	Ziheng Li
4.3.1	mGestGUI	Ziheng Li	Ziheng Li
5	Conclusion	Aung, Ziheng Li	Ziheng Li
6	Future Work	Aung, Ziheng Li	Ziheng Li

Abstract

Desktop at Fingertip is a proof-of-concept prototype system that approaches the desktop interaction paradigms of cursor control and text input through two-finger in-air micro-gestures. Our system is based on millimeter wave radar sensing, and does not require instrumentation on the user. We present "ThuMouse", a novel interaction paradigm aimed to create a gesture-based and touch-free cursor interaction that accurately tracks the motion of fingers in real-time. ThuMouse allows users to experience the truly mouse-less monitoring of the cursor using frequency-modulated continuous-wave (FMCW) radar. ThuMouse regressively tracks the position of a finger, allowing a finer-grained interaction. The paper features the gesture sensing pipeline we built, with regressive tracking through deep neural networks, data augmentation for robustness, and computer vision as a training base. We also present *IndexPen*, an interaction mode that can successfully identify 29 distinct characters, representing the letters *A-Z*, as well as *Space*, *Backspace*, and *Enter*. We investigate the technical and design considerations of the combined system to enable new in-air, micro, tracking-based interaction and discuss future work that can be enabled with these paradigms.

Summary

This project involves three graduate students working towards a directed research and two undergraudate students working towards a MQP requirement. My role in this group is same as everyone in the group: to collect data, run experiments, write reports, submit to conferences, and deal with software and hardware challenges. In this document, we will mainly be reporting the ThuMouse and IndexPen projects that we have worked on. The document details a general timeline that we will adhere with and make changes along the way, a completed methodology that we utilized to carry out our projects, and a detailed section about the background research that we did before starting the projects. We will also be discussing what we have learned from making these projects happen and how we incorporate the lessons into our future projects.

Acknowledgements

We would like to express our deep gratitude to Meijie Wang, colleagues at the CS department, and lab coworkers, and special thanks to Meijie Wang lent expertise with constructing the experiment environment, with which we tested the working prototype that we conduct studies.Besides, we would like to extend our thanks to our coworkers at the lab who participated in our pilot study, allowing us to accumulate more data to refine our model given limited time constraints.

Contents

List of Figures

List of Tables

1 Introduction

Overview As interaction continuously moves away from desktop to mobile, also with hands-free gadgets such as virtual reality (VR), Augmented Reality (AR) gaining popularity, the demand for efficient and compact interaction methods is ever necessitated. In this decade, radio-frequency (RF) based interfaces have been largely investigated. Many proposed systems leverage channel information from existing infrastructure such as commodity WiFi and Radio-Frequency Identification(RFID) to detect human activities and gestures. Under the umbrella of 5G standards that is quickly taking shape lately, wireless technology is sought to display increased speed, reduced latency, and become more energy-efficient and cost-effective. Among the various new standards, Frequency-Modulated Continuous-Wave (FMCW) variant of mmWave radars captures the spatial and temporal information of objects by transmitting a continuous wave modulated in a specific frequency range. The dynamic profile given by the device, owing to the high signal frequency (usually greater than 60GHz), as well as well-established processing chain, is remarkably precise and capable of achieving sub-millimeter accuracy. Also, the data of mmWave sensors is relatively light-weighted compared to other gesture input methods such as computer-vision based approaches, making mmWave radar more fitted in devising ubiquitous gesture interface for low-power-rated devices such as those for smartwatch and portable AR.

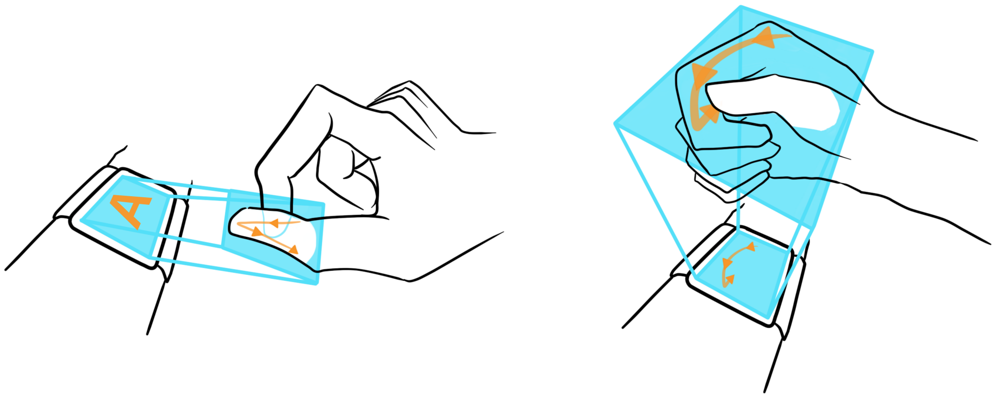


Figure 1: Desktop at Fingertip enables in-air two finger text input (left) and cursor control (right) on devices with limited physical size and no instrumentation required on the hand.

Motivation Just like the touchscreen technology in the early 80s, we believe that the use of mmWave technology as a ubiquitous gesture interface has started to bloom into its respective usage, as evidenced in [1]. The mmWave technology can be applied to many applications such as automotive applications (33), VR headsets, etc. in the form of gesture sensing. It is believed that a viable ubiquitous gesture interface can be established by integrating mmWave technology into everyday electronics such as smartphones, smartwatches, and laptops. For example, the rub motion of the thumb against the index finger can be interpreted as a volume control. In terms of laptops, we believe that this technology can be revolutionary as we would be able to monitor the mouse by tracking the movement of the fingers and allow the users to input text using the gesture sensing applications. All in all, the mmWave technology has the potential to become the new "capacitive touch" that everyone uses on a daily basis.

Contributions The challenges remain in exploring algorithms that enable the capturing of finer micro-gestures, and designing an interface for real-world interaction with it. We propose a mmWave-based cursor interaction system - ThuMouse. It is a mmWave-sensor-based interaction paradigm that can be used as a pointing device. The system tracks the motion of the thumb rubbing on the index finger to actuate the control of a cursor. We build on the mmWave processing chain proposed in (13), and designed FMCW signal suited for micro-gesture sensing. An end-to-end data processing pipeline is built with a robust yet light-weight deep learning regressor and classifier. Experiments were conducted to evaluate the reliability of the system. The thumb tracking achieved a mean square error of $7.55e^{-4}$ mm on the x-y plane and an interface was designed where user may control a the cursor in a graphic user interface (GUI) in real-time.

Overall, mmWave sensors presents numerous advantages as a gesture sensor, such as their characteristic environment-independence, compact size, and low cost both computationally and monetarily, making them competent candidate upon which novel gesture scheme can be established. In this regard, prior work includes earlier exploits by Arbabian et al on high-frequency pulse-band radar, and more recently, Texas Instruments mmWave sensor and Project Soli of Google ATAP have contributed to the design of solid-state mmWave FMCW sensors applications around them.

On the other hand, we summarize the limitation of existing system as follows: (1) comparing to capacitive sensing or optical sensors, mmWave radars lacks spatial resolution due to fact that the reflected signals are superimposed; albeit this is offset by the high temporal/velocity resolution and highly sophisticated prediction model, distinguishing similar gestures suffers because the moving parts (i.e. a specific finger) resides in close proximity between each other. (2) Current approaches feed to the machine learning model with the raw analog-to-digital converter (ADC) output with minimal pre-processing. The resulting data profile is usually a range-velocity image of the object in front of the radar. Those data can vary across different platform in their size and resolution, which calls for domain specific predicting models (in contrast, image data from cameras possesses much more generality). (3) Moreover, the high throughput of data taxes the hardware to be able to achieve real-time gesture-recognition; the processing pipeline must be limited in its complexity, where input accuracy must give away for the real-time interaction. (4) The features given by mmWave devices is relatively unique comparing with other sensing technologies, which makes it difficult to adapt existing pre-processing and predicting methods. The lack of distinct and human-readable features (due to low spatial resolution in the raw profile) also poses difficulties in visualizing the data, through which better-performing models can be built and more elegant interfaces such as tracking the motion, can be designed from webcams with "You Only Look Once (YOLO)(an object detection algorithm)(27) as well as LeapMotion as the training base or ground-truth for the radar data. Through this method, the CNN network could be well trained and we could also evaluate its performance specifically.

To train model for the mmWave-sensor-based system, we set up a dual input data collection plan. On the one hand, data is streaming from mmWave radar in the form of detected points (DP). On the other hand, we collect data from webcams and using "You Only Look Once (YOLO)"(an object detection algorithm)(27) as the training base or ground-truth for the radar data. Through this method, the CNN network could be well trained and we could also evaluate its performance specifically.

Our main contributions are : (1) we leverage the sensing ability powered by the signal processing chain from (13); this enables us to detect the spatial position of objects as well as their velocity, making it possible to track the finer gesture. (2) Design the real-time tracking with an end-to-end gesture pipeline using the radar point cloud and applying several data augmentation, enriching the feature and build more robust models. (3) We designed and evaluated 3D Convolutional Long-short-term-memory (LSTM) deep learning model with which the point cloud data is passed to realize the motion tracking and gesture classification(4) We examine micro-gesture designs in the context of desktop interaction and present the concept of *Desktop at Fingertip*, which includes *ThuMouse* and *IndexPen* for cursor control and text input. (5) We investigate the use of millimeter wave radar sensors in detecting an appropriate gesture set, and detail the real-time processing pipeline we have developed. (6) We investigated technical efficacy of *IndexPen* and showed that the proposed approach is able to resolve 29 *IndexPen* characters with high within-user accuracy – 99.7%. (7) We share our collected data and the source code for the mmWave interface we developed and the script that processes the data and provides the evaluation results

2 Background

Building on top of other RF-based interfaces, radar, especially the mmWave variant is given rise by the upcoming 5G standards. As a sensing technology, mmWave FMCW radar possesses high resolution in capturing motions, and to our study specifically, human gestures. In the literature review, we studied (1) prior work that built radar-based interfaces, (2) machine learning algorithm used for gesture detection.

2.1 Radio Frequency Cloud and HCI

With *Desktop at Fingertip*, we aim to capture micro-gesture in-air gestures and control the coordinates in a mobile device accurately and conveniently. To do this, we use millimeter wave radar, which is one type of radio frequency (RF) information that is increasingly available for human-computer interaction applications (19). Radio-frequency (RF) data from existing infrastructure (e.g. WiFi, Radio-frequency identification (RFID)) have been explored to recognize human activities and gestures (24) (41). This is accelerating as 5G standards emerge, with devices possessing increased speed, reduced latency, and increased energy-efficiency and lower cost (9).

As noted by Pahlavan, et al. (19), unmodified global system for mobile communications (GSM) signals have been used to enable tapping, hover and sliding gestures around a mobile device (39). With radio signals and one external sensor hanging on the wall, researchers have demonstrated that gait velocity and stride length can be monitored, enabling health-aware smart homes (11). Indoor WiFi signals have been used to identify motion direction, leading to a contactless dance “exergame” (26) as well as sign language gesture recognition (16). Other work demonstrated that 5GHz WiFi can be used to achieve decimeter localization accuracy of up to four users as well as activity recognition of up to three users doing six different activities

2.2 Micro-gestures in HCI

With the increase in micro-gestures in HCI, there have been numerous types of gestures proposed. To gain insight into end-user preferences, Chan et al. (4) conducted a user elicitation study of single hand micro-gestures, and we refer the reader to this paper for an in-depth exploration. Their study identified four main categories of micro-gestures: *tap*, *swipe*, *draw*, and *circle*. They also found that out of the four, taps and swipes were the most commonly used by end users. *ThuMouse* enables *swipe*, *draw* and *circle*, as it a flexible cursor control method, and the thumb can perform any of these on the index finger. *Draw* was not frequently used by end-users, but is the basis of the *IndexPen* interaction we present. Because we propose an alphabet resembling handwriting, the draw gestures can be easily remembered, which may have been an issue for other *draw* gestures. Due to the fact that the thumb can comfortably reach the other parts of the hand, Chat et al. found that the thumb was used more frequently in general, and was used on all the elicited swipe gestures performed by the participants (4). In both *ThuMouse* and *IndexPen*, we combine the thumb and the index finger.

Different form factors for enabling micro-gestures in the hand have been explored. These can be categorized by whether they take a wearable or an in-air approach.

2.2.1 Wearable

Wearable devices have been proposed for micro-gesture detection of the hand as they can capture physical or physiological data well through the direct contact with the user's skin. *TipText* (36) is a miniature QWERTY keyboard broken into a grid with several letters in each section. By clicking the tip of the index finger with the thumb, users can select a grid section and the specific letter within that section is disambiguated using a language model. Benefitting from the small size, TipText has the potential to be integrated with watches and smartphones. Electromyography, which senses muscle activation during motor movements of the hand and fingers, provides another way to detect gestures by directly decoding the human muscle activities (32; 31). Force sensitive resistors were used in *WristFlex* to successfully recognize subtle finger pinch gestures (7). FingerPad (5) has the most similar interaction properties to *IndexPen*, but uses two nail-mounted devices that detects magnetic field intensities and transforms that to coordinates for a user interface.

2.2.2 In-air

Compared to wearable methods, in-air approaches for gesture detection may be more favorable as they do not require the user to attach any electronics to the body.

This removes some of the device-specific constraints and can present advantages on aspects of sanitation, aesthetic, and utility. For micro-gesture detection, many optical methods are used to extract features of the user. A 3-D time-of-flight (TOF) camera (14) has been used to simplify segmentation between the hand and arm. In using time-of-flight camera, the depth features are included to distinguish certain gestures: gestures which have the same projections, but different alignments. The projections of the hand onto the x and y axis are used as features for the classification. The region of the arm is discarded since it contains no useful information for the classification and due to strong variation between human beings. Yet, the TOF camera has disadvantages due to low resolution. Marin et al. confirmed the effectiveness of using the Kinect camera in human body recognition applications (17). They first extracted the gestures from the acquired depth and color data and then two different types of features were computed from the 3D points corresponding to the hand. They also explored Leap Motion's optical method and reported that it differs from the depth camera which return a complete depth map. The Leap Motion provides a higher level but more limited data description while Kinect provides the full depth map. Even when the data provided by the Leap Motion is not completely reliable (e.g. some fingers might not be detected), the proposed set of features and classification algorithm allows to obtain a good overall accuracy (17).

2.3 Nontraditional real-time interfaces

CV: With the aid of increasingly high processing power and advanced computer algorithms, CV-based real-time gesture interfaces have taken a large percentage of the market share; examples includes Microsoft Kinect (38), Leap Motion (17), Emotiv, and Holo-lens. However, the model accompanying optic sensors is usually complex due to their rich features (large number of pixels and can be highly dependent on the ambient lighting.

Radio Frequency-based: Because of the universal usage of RF signals(e.g., Wi-Fi, RFID, radar signal), analyzing RF propagation to detect the human activity and gesture is becoming a hot spot. Many researches took advantage of existing infrastructure such as RFID (41)(3) to detect the gesture. At the same time, using commodity Wi-Fi devices to actuate whole-home gesture sensing and activity detection is also investigated in (23)(1)(10)(8) (40).

2.3.1 Radar-based interfaces

Radar, as a special variant of RF technology, is built specifically for detecting objects in its coverage. While long-range impulse radar have featured in tracking aircraft and other large-scaled entities, the use of radar in consumer electronics market has remained obscure for decades.

Early work in (37)(2) laid a hardware and signal processing foundation in utilizing mmWave in gesture detection. It is evident from (37) that feasible gesture interface can be implemented with radars that transmits frequency-modulated signal on Extremely High Frequency (i.e. mmWave). More recently, Soli (15) (34) (22) (29) brought it into the context of micro-gesture sensing, wearable, and smart devices. In this setting, gesture interface presents many-faceted possibilities: Soli introduced the idea of virtual tool and proposed the Soli 60-GHz mmWave FMCW radar to track the fine-grained gestures, capable of detecting 4 gestures from a single user(15) . On the other hand, (33) implemented car infotainment interaction with the sensor proposed in (15) , able to distinguish two sets of gestures with each containing three different motions.

In detecting different gestures, traditional machine learning approach such as random forest, Support Vector Machine(SVM). In the paper of Soli (15) and human-car interaction(33),their systems use Random Forest algorithm to automatically detect and select most relevant features for multiple classification of gestures. And in the article of (6), the multiclass-SVM is used to detect human gestures but require vectors of fixed dimension as input. On the other hand, in recent years, Deep learning has displayed great potential in performing through it capability to learn intermediate representation of raw data and been shown to routinely beat traditional machine learning methods aforementioned (34).

On the other hand, (34) investigate the use of deep learning models; a features that a mmWave sensor can yield is the range-velocity map on per-frame bases. Noting the fact this type of feature is essential a 2D image-like array, (34) applied Convolutional Recurrent Neural Network (CRNN), typically used for video classifications and arrived at accuracy of 87% on 11 gestures across multiple subjects.

We summarize the limitations of existing system as follows: (1) comparing to capacitive sensing or optical sensors, mmWave radar lacks spatial resolution due to the fact that the reflected signals are superimposed; albeit this is offset by the high temporal/velocity resolution and highly sophisticated prediction model, distinguishing similar gestures suffers because the moving parts (i.e. a specific finger) resides in close proximity between each other. (2) Current approaches feed to the machine learning model with the raw analog-to-digital converter (ADC) output with minimal pre-processing. The resulting data profile is usually a range-velocity image of the object in front of the radar. Those data can vary across different platform in their size and resolution, which calls for domain specific predicting models (in contrast, image data from cameras possesses much more generality). (3) Moreover, the high throughput of data taxes the hardware to be able to achieve real-time gesture-recognition; the processing pipeline must be limited in its complexity, where input accuracy must give away for the real-time interaction. (4) The features given by mmWave devices is relatively unique comparing with other sensing technologies, which makes it difficult to adapt existing pre-processing and predicting methods. The lack of distinct and human-readable features (due to low spatial resolution in the raw profile) also poses difficulties in visualizing the data, through which better-performing models can be built and more elegant interfaces such as tracking the motion, can be designed.

We are jointly inspired by great sensing capability of mmWave sensors on resolving fine-grained finger motions as well as the virtual tools brought up in (15). In this work, we discuss ThuMouse, a gesture system building on top of (13) that aims towards partial desktop environment interaction. Specifically, the system tracks the motion of the thumb to simulate touch-pad or mouse. The signal processing pipeline can be applied to any mmWave devices, and we make available the software processing chain including prepossessing, data augmentation and model-training. The table below shows the literature review we conducted before starting the experiments.

Table 1: Literature Review

System	Problem	Hardware and Features	Machine learning alg
Wisee	(Push,dodge, strike,pull,Drag, Kick,Circle,Punchx2, Bowling)x (LOS,NLOS,TW)/94%	Wi-Fi, Doppler Shift	No ML applied
Soli	Virtual button, virtual slide, Horizontal Swipe, Vertical Swipe/92.10%	Mmwave, fine displacement, total measured energy, measured energy from moving scattering centers, scatter- ing center range, velocity centroid, and many others	No ML applied

Table 1: Literature Review

System	Problem	Hardware and Features	Machine learning alg
Allsee	Flick,Push,Pull, Double Flick, Punch, Lever,Zoom in, Zoom out/94.4% for TV transmission and 97%	TV transmission and RFID	No ML applied, analog only
In-car infotainment control	Low wiggle(91%) Turn over(99%) High grab(95%)— for Driver Swipe(91%) Large circle(97%) Small circle(96%) —for passenger	Mmwave, range, acceleration, energy total, energy moving, velocity, velocity dispersion, spatial dispersion, energy strongest component, movement index.	Random Forest classifier

Table 1: Literature Review

System	Problem	Hardware and Features	Machine learning alg
Interacting with Soli	Pinch Index, Pinch Pinky, Finger Slide, Finger Rub, Slow Swipe, Fast Swipe, Push, Pull, Palm Tilt, Circle, Palm Hold/(87%)	MmWave, range-doppler profile	CNN and RNN
Indoor motion detection	Falling, Running, Walking, Sittting/85%	Wi-Fi, channel state information	Random Forest KNN SVM

Table 1: Literature Review

System	Problem	Hardware and Features	Machine learning alg
Wearable Physiological Monitoring Systems	Standing(90.11%), walking(91.49%), running(89.56%), lying(90.71%), crawling(83.89%), climbing(88.16%), on the stair(86.91%)	Radio frequency channel	SVM
TipText	QWERTY keyboard entry	thumb-tip to tap the tip of the index finger	Spatial model and language model
Soli Music	Musical interaction	Mmwave, acceleration, fine displacement and energy total	same with soli

Table Notes,
extra
information

3 Methodology

In this section, we discuss the system pipeline with which the preliminary studies is carried out. Namely, we explain the detection principle of mmWave sensors, the experimenting environment and hardware, the evaluated gesture schemes, and algorithm used to process the data and actuate the interactions. It is worth noting that content of this section are up to change based on future development.

3.1 Radar Signal Processing

We go over the RF front-end, its various stages of signal processing, and explains why mmWave FMCW radars are suitable for dynamic gesture recognition. It also covers how the radar detects objects as reflecting points and presents the hardware used for evaluation.

3.1.1 Points Detection

The mmWave FMCW radar detects an object through the principle of reflection. During each sampling period, the transmitter (Tx) sends a chirp, which is a signal with its frequently changing over time. The reflected signal from objects in the field of view (FoV) of the radar is picked up by the receiver antennas (Rx). The signal used for further analysis is given by mixing the Tx and Rx signals in the following way:

$$X_{Tx} = sin[\omega_{Tx} \times t + \phi_{Tx}] \tag{1}$$

$$X_{Rx} = sin[\omega_{Rx} \times t + \phi_{Rx}] \tag{2}$$

$$X_{mixed} = sin[(\omega_{Tx} - \omega_{Rx}) \times t + (\phi_{TX} - \phi_{RX})] \tag{3}$$

From the above equations, we can calculate the intermediate frequency (IF) signal X_{mixed} or X_{IF} by combining the TX signal and RX signal.the IF signals only valid during the chirping time (i.e. when the Tx signal is present). The Fast Fourier Transform (FFT) is performed on the IF signal, the frequency peaks then corresponds to the distance between the radar and the reflecting object. This FFT is known as one-dimension FFT (1DFFT).

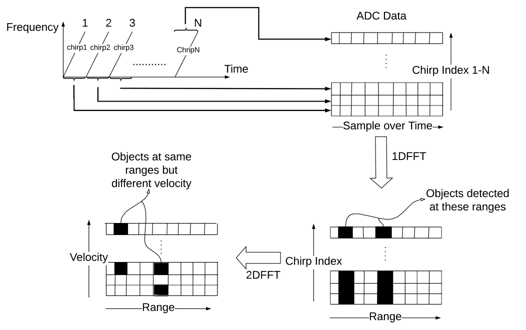


Figure 2: the sensor extracts the analog to digital converter (ADC) data from the instantaneous frequency (IF) signals and put in ADC data bins. Applying row-wise FFT (on the data of each receiver), the range profile is obtained where a further FFT is performed column-wise over multiple bins (chirps) to get the velocity of the detected points at certain ranges(13)

On top of the first FFT over a single chirp, which resolves the range, the radar emits a series of chirps in quick succession as depicted in Figure 1. Through a second FFT (2DFFT) over the IF signal for those multiple chirps, the radial velocity (the velocity of the object relative to the sensor) of objects can be obtained. The detecting power of mmWave radar, when resolving the radial range and velocity, is defined by two resolutions: range resolution and velocity resolution. Range and velocity resolution is given by the following equations:

$$Range_{res} = \frac{C}{2 \times F_B} \tag{4}$$

where C is the speed of light(3×10^8) and F_B is the frequency band with which the chirp sweeps. For 4GHz of chirp band, it gives $3.75cm$ range resolution.

$$Velocity_{res} = \frac{\lambda}{2T_f} \tag{5}$$

where λ is the wavelength of the chirp starting frequency and T_f is frame time which equals to the number of chirps times the duration of a single chirp. If radial distance between two reflecting point is less than the range resolution, the radar would identify the two as a single detected point, and likewise for velocity. Note that because the velocity is calculated after determining the range, closely places objects (pairwise distance no greater than the range resolution) can still be distinguished as different objects if the difference in their velocity is larger than the V_{res}.

This is the key to gesture detection with FMCW radars, as the fingers usually resides within $Range_{res}$ but during dynamic gestures, fingers or parts of hand typically move at different speeds in relatively to each other. This creates a dynamic profile where lays a fertile feature space for micro-gesture study.

So far we are able to detect objects with their radial range and velocity, now with multiple chirps being received by the Rx, the angle of a reflecting point is entailed by the phase change of the IF signal across different Rx. Moreover, with antennas arranged both horizontally and vertically, the sensor is able to resolve both azimuth and inclination.

The angle resolution equation is as follows:

$$\theta_{res} = \frac{\lambda}{N \times d \times cos(\theta)} \tag{6}$$

, where d is the spacing between Rx antennas, and θ is the angle of objects. The equation signifies that θ_{res} is non-linear. As the function sin(theta) is the most sensitive when theta is around zero degrees, the sensor gives the best angle discrimination when the object is directly in front of the radar (i.e. $\theta_{object} = 0$). As theta increases and approaches 90 degrees, the angle estimation accuracy degrades as the sin(theta) value does not change much.

With the radial range, angles including azimuth inclination, and velocity, the points are essentially velocity heat-map in 3D polar coordinates. Thus, the dynamic profile that the sensor yields is a list size-4 vectors $(r, \theta, \Phi, doppler)$, each representing a detected point, where r is the radial range between the point and the original (where the radar is situated), θ is the inclination or vertical angle, and ϕ is the azimuth or the horizontal angle.

To summarize, the reflecting points must satisfy the following pairwise conditions to be seen by the radar: (1) if the radial distance (relative to the radar) between to reflecting surface is greater than R_{res} (2) if (1) is not satisfied, then difference in radial velocity of the two points must be greater than V_{res} (3) if (2) is not satisfied, the angle between the two points must be greater than the θ_{res}.

In dynamic gesture capturing, the motion is considered to be more crucial compared to static hand shapes. Meanwhile, in order to eliminate other static environmental noise such as the arm, floor and walls, Clutter Removal is applied at the end of second FFT which calculates the velocity(13). It also abates the computation load as the static points are removed from all the radar frames. By subtracting the mean from the S_{2DFFT}, most immobile objects are removed from the samples.

$$S_{2DFFT} = S - mean(S_{2DFFT}) \tag{7}$$

where S_{2DFFT} is the bin formed by 2DFFT and S are each sample in S_{2DFFT}.

To be able to further manipulate the point cloud data such as transformations, we perform a Polar to Cartesian Coordinates conversion as the last step in the point processing pipeline.

$$x = r \times sin\theta \times cos\phi y = r \times sin\theta \times sin\phi z = r \times cos\theta \tag{8}$$

where the x, y and z are the Cartesian coordinates of the detected point. Their unit are in meters.

3.1.2 Signal Design for Gesture Application

The Understanding of the above resolution equations (4)(5)(6) is crucial when devising suitable signal shape for the gesture recognition. It defines several key points in using mmWave as a gesture interface (1) To detect fine-grained gestures, range resolution needs to be reasonably small so that the structure of the hand will be distilled in the feature space. (2) dynamic gesture recognition requires a high degree of temporal (velocity) resolution(15). (3) Moreover, to reduce ambiguity on the angle of arrival (AoA), the gesture-performing area should be aligned with the central axis perpendicular to the antenna module.

However, it is out of the scope of this paper to compare how different hyper-parameter affect the precision of tracking. Without a in-depth analysis of tuning signal configuration and with consideration on hardware limits, we apply

$$F_B = 4GHz \qquad F_{start} = 60GHz F_{end} = 64GHz \qquad T_f = 20mesc \tag{9}$$

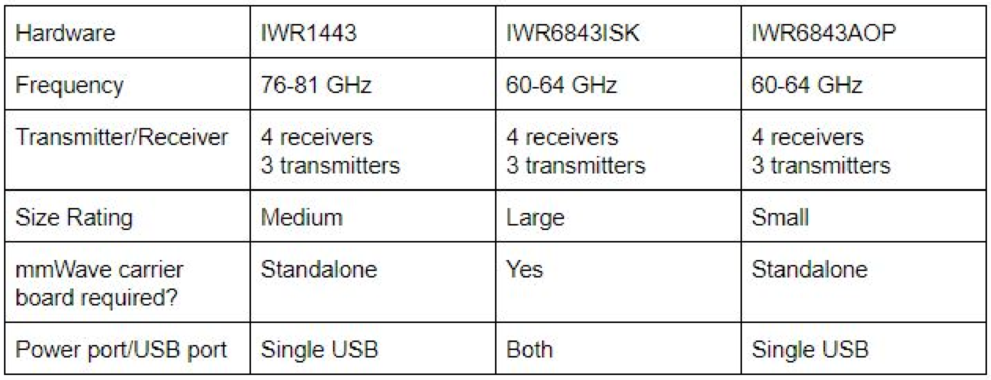
with 30msec inter-frame delay for processing, which gives a frame rate of 20FPS, Range$_{res}$=3.75cm, Velocity$_{res}$=0.12m/s A extension of this work can be made on studying the how the parameters affects the detecting precision of mmWave sensors.

3.1.3 Hardware: IWR6843ISK

Our system is based on the IWR6843 designed by Texas Instrument, an integrated single-chip based on the FMCW technology and the frequency is modulated between 60 to 64 GHz. Benefit by the on-board digital processing unit and hardware accelerator aiming at quickly computing FFT and resolve log-magnitude operations, it can detect, at a relatively high frame rate (20 FPS), the spatial coordinates (range angle) and velocity of objects. *IWR6843ISK* is an antenna module that falls under the line of the *IWR6843* mmWave sensor device. It has a long range on-board antenna with 108° azimuth field of view (FoV) and 44° inclination FoV. *MMWAVEICBOOST* provides a platform for the rapid evaluation.

3.1.4 Hardware: IWR6843AOP

Our system is based on the Texas Instrument's IWR6843AoP (antenna on package), an integrated compact mmWave radar device. It operates between 60 to 64 GHz thus giving a best range resolution of around $3.75cm$ as stated above, well-suited for micro-gesture detection applications. Benefit by the on-board digital processing unit (DPU) aiming at quickly compute FFT and resolve log-magnitude operations. The radar duty cycle includes (1) data capturing phase, when the radar continuous sends chirp signals and receiving them (2) processing phase during which the above FFT calculations is carried out with the DPU, and (3) transmitting phase: where the result of the detection is relayed through a serial interfaces the the host computer. Due to the limited processing power of the hardware, we need to make some trade offs between the frame rate, range resolution, and velocity resolution.



Hardware	IWR1443	IWR6843ISK	IWR6843AOP
Frequency	76-81 GHz	60-64 GHz	60-64 GHz
Transmitter/Receiver	4 receivers 3 transmitters	4 receivers 3 transmitters	4 receivers 3 transmitters
Size Rating	Medium	Large	Small
mmWave carrier board required?	Standalone	Yes	Standalone
Power port/USB port	Single USB	Both	Single USB

Figure 3: The hardware comparison table for TI mmWave devices

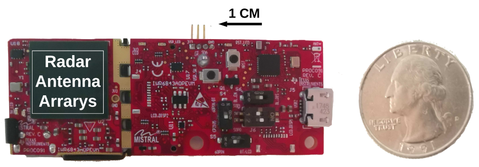


Figure 4: The physical appearance of the IWR6843 Antenna-on-Package radar device, and its comparative size

By the end of each duty cycle, the host will have received the detection information captured during the last cycle. This information is defined as a frame that includes two dynamic profiles, that of **range-doppler** and **range-azimuth-elevation**.

3.1.5 Range-doppler Profile

The range-doppler profile is obtained at the end of the 2D FFT, where the range and velocity information of the objects are extracted. The profile is a two-dimensional matrix with columns being the velocity and rows being the range. At each point in the matrix, the value is the amplitude calculated through the 2D FFT and represents how strong the reflecting signal is at a specific range and moving at some velocity. It can help to visualize the features as heatmaps. Figure 5 shows how motions are reflected in the range-doppler profile.

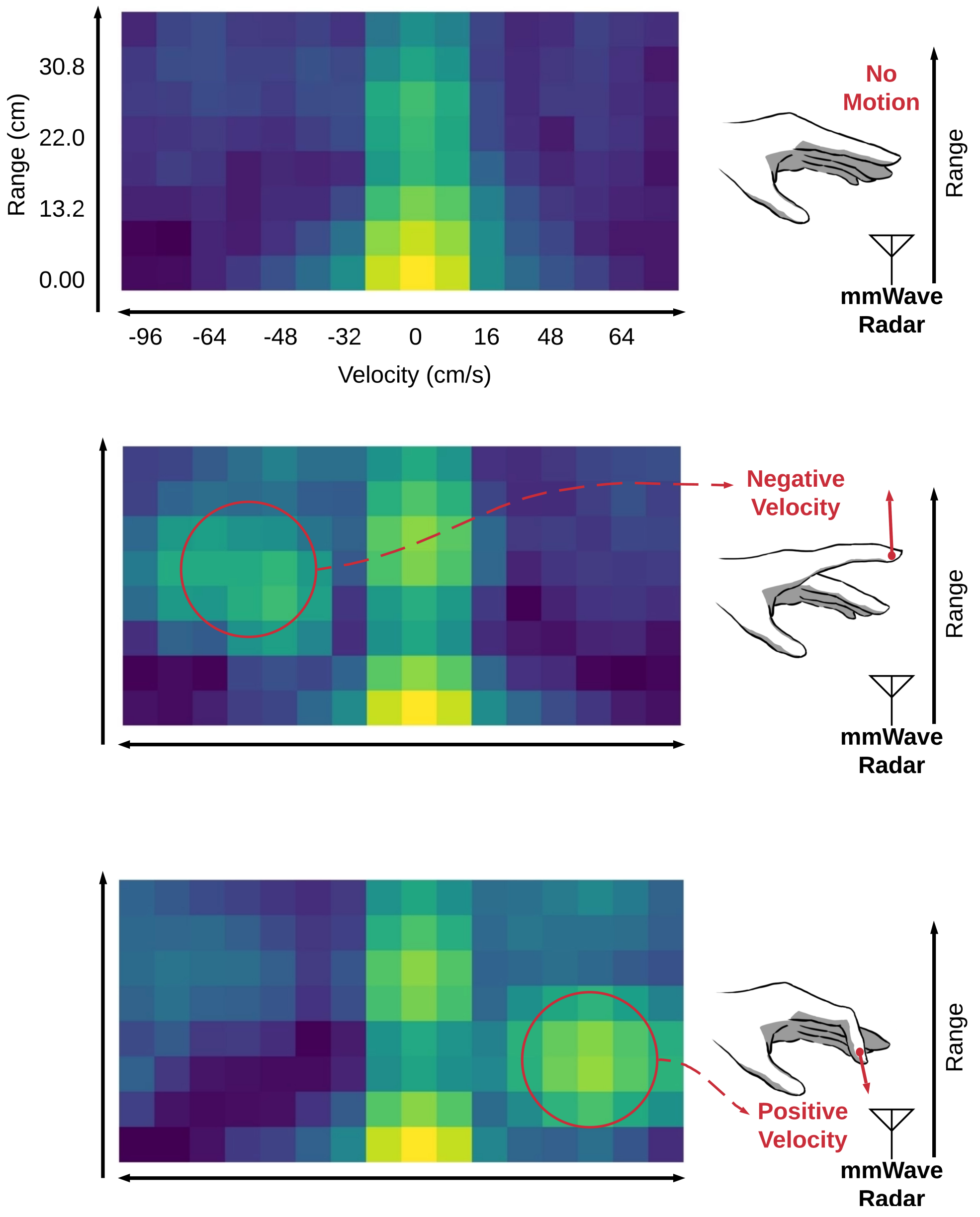


Figure 5: The range-doppler profiles show the relative power for objects of (1) a specific distance from the radar;(2) moving at a certain radial velocity relative to the radar

The number of rows corresponds to the number of range bins and we used eight, making the radar able to see objects up $r_{max} = r_{res} \times num_r ange_b ins = 35.2cm$. The number of columns is the that of the velocity bins, which we set to 16. It gives a maximum detectable velocity $v_{max} = v_{res} \times num_v elocity = 2.56m/s$. These parameters define the gesture performing range for the system. The numbers of bins are constrained by the data processing time and the transmitting speed of the serial interface. With the trade-off between the frame rate. We used these parameters to ensure the best resolutions at 30 FPS.

3.1.6 Range-azimuth-elevation Profile

Range-azimuth profile (Figure 6) is a dynamic feature set given by the radar. Obtained at the end of the signal processing pipeline, this profile represents angle of arrival (AoA) for the detected objects. Both the azimuth and the elevation combine into a matrix where the rows are the angle (azimuth and elevation) between the objects and the radar. The columns are the same as in the range-doppler profile; it shows how far away the objects are from the radar.

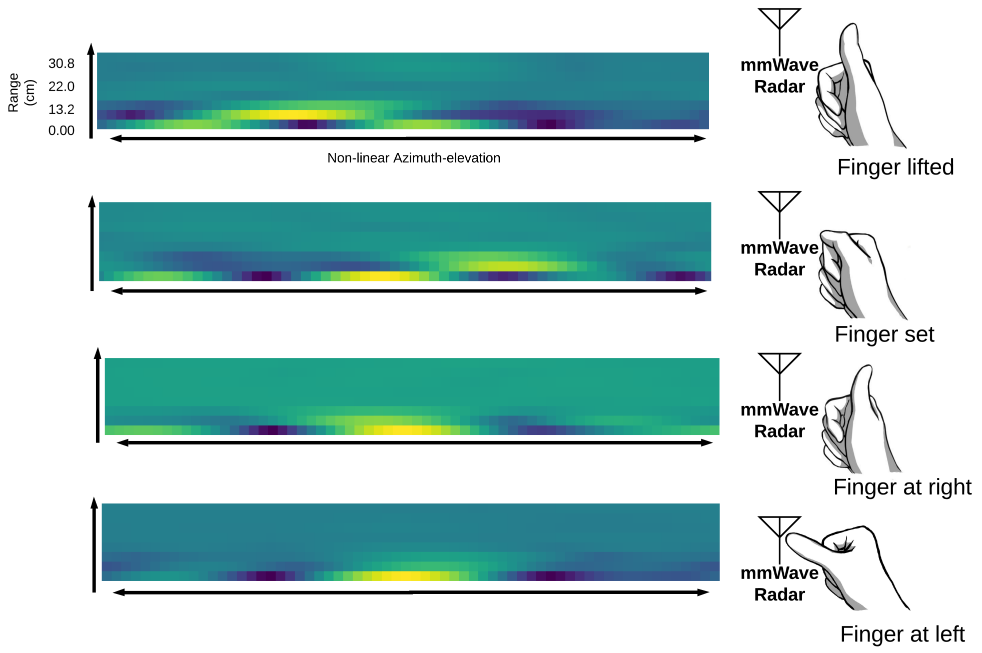


Figure 6: The range-azimuth-elevation profiles gives the relative power for objects of (1) a specific distance from the radar; (2) at certain azimuth and elevation angle relative to the radar. The azimuth-elevation axis is non-linear as the angle resolution degrades as the objects moving away from the central axis of the sensor (Equation 6)

In the DFT system, we collect frames from the mmWave sensor, with each frame consisted of a 8×16 range-doppler profile and a 8×64 range-azimuth-elevation profile. Each frame lasts $33ms$, giving roughly 30 frames per second (FPS). The sequenced frames is then used as the deep learning features in the next step of the pipeline to give the final *ThuMouse* tracking and *IndexPen* classification.

3.1.7 Configuration

We want the maximum possible specifications we can get out of the radar module but these specifications come at a cost of the CPU load. Whenever the CPU load goes over 100 percent, the board freezes and hangs onto the last state it was in and we need to restart the board. In order to account for CPU load as well as the sudden surges of detected points, we are only using 20 frames per second as for frame rate, 0.039 meter for our range resolution, and 1.1 meters per second for the velocity resolution. In order to detect movements, we use "Remove Static Clutter". It removes all the detected points of a stationary object in front of the radar.

3.2 DFT Gesture Paradigms

In our study, we evaluated two gesture schemes: IndexPen and ThuMouse: gestures for text input and cursor-like interaction respectively.

3.2.1 IndexPen

IndexPen are performed by drawing characters with the tip of the index finger against the distal.(picture)Turning the hand 90° allow natural transition between ThuMouse and IndexPen. To minimize the effort for detecting without the loss of writing comfortable, we use the partial of the Palm Pilot alphabet, our alphabet for evaluation includes 4 letters (E, H, L, O) and one special characters: backspace. The evaluation set is extracted from the alphabet so that the users can input

’HELLO’. The order of strokes used for IndexPen is the Palm Pilot alphabet (28). This is to avoid unnecessary noise and to retain data purity.

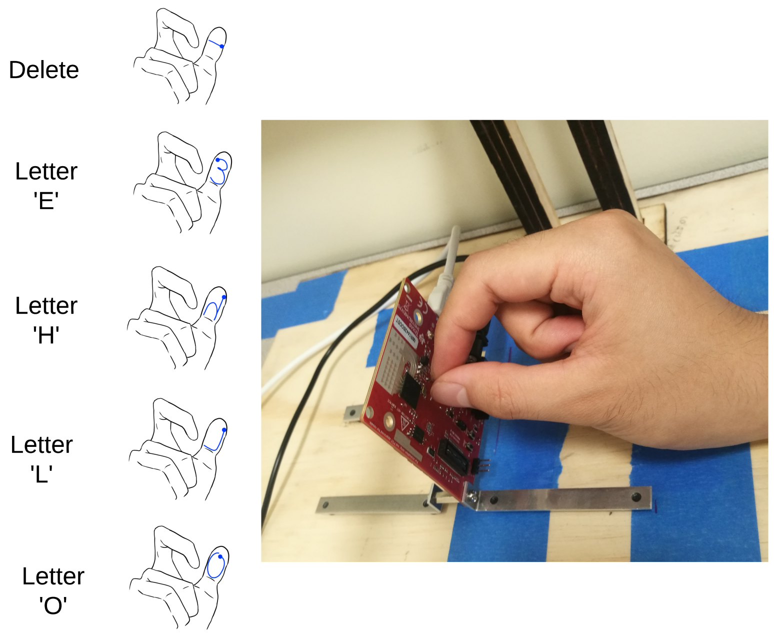


Figure 7: Left: the alphabet where the index finger of the right hand writes on the the face of the thumb. Right: the IndexPen gesture is performed with the thumb pointing at the center of the antenna module. This ensures best angle resolution as discussed in section 3.1.1

3.2.2 ThuMouse

ThuMouse is the application we build around the capability of locating the position of the thumb-tip. We define ThuMouse gesture follows: user may move his or her thumb against the planar surface of the index finger. The temporal displacement of the thumb is reflected in cursor movement; the gesture is similar to ’Thumb Rub’ in(34)(33), but ThuMouse actually resolves the position of the thumb on the rubbing surface, allowing for finer-grained controls. In other words, the thumb tip is playing the role of mouse and surface defined by the index finger is emulating the mouse pad, which is natural, soft and unobtrusive. This gesture can maximum guarantee the the index finger surface have ample space where the thumb tip can navigate.

In addition to resolving the x-y coordinate of the thumb-tip that emulates the movement of a mouse, we introduce the z axis (perpendicular to the thumbnail) such that if the thumb leaves surface of the index finger, the tracking would freeze just as a touch-pad will cease the cursor movement when it lost contact with the controlling finger. Moreover, the system interprets click from successive up-and-down motion in a short time interval.

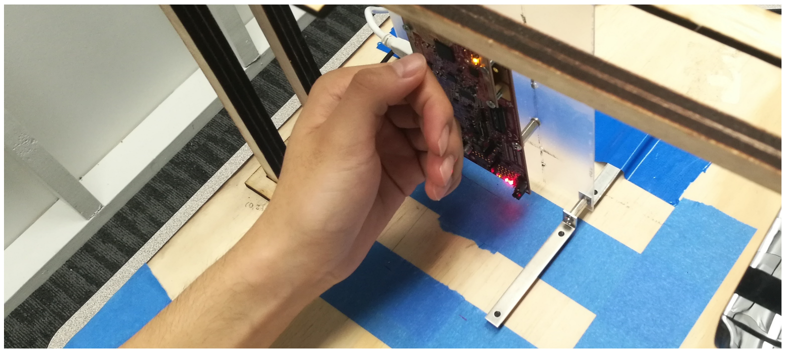


Figure 8: Like the IndexPen, the ThuMouse gesture is performed with the thumb of the acting hand pointing at the center of the antennas to achieve best angle resolution as per section 3.1.1

IndexPen enables text entry as gestures performed by the index finger writing on the the face of the thumb. To minimize additional training that the user has to be subjected to, we aimed to create an input mode that reflects how a person naturally writes. We took inspiration from the Palm Pilot alphabet (28). This writing table was developed in the early days of handwriting recognition, and therefore the strokes are designed for the ease of identification. However, the PalmPilot alphabet was designed for writing pads and the features used are the pixel values. With *IndexPen*, the features used are based on the dynamic profiles produced by mmWave radars (i.e., range, velocity, angle).

Based on these design considerations, we created the *IndexPen* gestural alphabet (Figure 10). It includes 29 fingertip writing gestures, representing the 26 letters and 3 utility keys. The gesture mimics real-world writing strokes and is designed to afford a natural way of writing. While striving for natural gestures, we also take into account whether such gestures are distinguishable through the mmWave radar features. As will be described below in the Radar Signal Processing section, the radar detects the range, angle and velocity of the objects. Letters with similar starting points, traces, and end points are likely to be confused. Based on these considerations, the letters such as A, F, K and T are simplified into one-stroke, while letters including O and V are given embellishments to make them more distinguishable from C and V respectively (Figure 10).

In addition to the 26 English letters, we add three utility characters to the table, including *space*, *delete* and *enter*. These three characters were designed with emphasis on the ease of performing the gestures and the likeness to real writing experiences. The *space* gesture is a slide towards the natural writing directionality, whereas the *delete* gesture is the reverse of the *space*. *Enter* is a forward swap intended to mimic 'pressing enter to send the message' action.

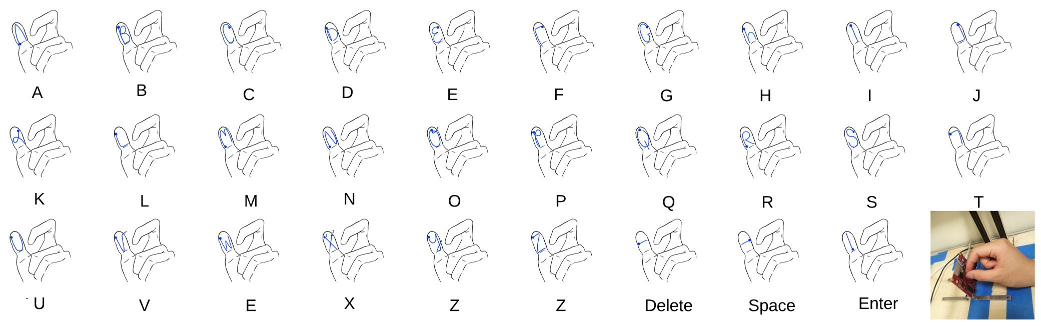


Figure 10: The IndexPen Gestural Alphabet. The basic text-entry interface has 26 letters, delete, space and enter. The illustration's gesture is with the left hand so that the characters are not 'flipped'. The snapshot at the bottom right shows a user ready to perform the IndexPen gesture.

3.3 Mode Switching Between ThuMouse and IndexPen

The user should be able to seamlessly switch between the two modes of interaction. The hand position for *ThuMouse* and *IndexPen* is different, and so differential decoding can enable automatic mode switching between the two modes (Figure **??**). To reach this level of context-awareness, future work will further evaluate the robustness of this approach.

3.4 Design Considerations

In envisioning the desktop input interface at the tip of the fingers, we examine three system design aspects that help to afford comfortable, and easy-to-learn controls. The three factors are accuracy, smoothness and learnability.

IndexPen

We define the following design considerations in evaluating the performance of IndexPen.

Accuracy The accuracy of IndexPen can be assessed in two-folds. Being a symbol classification problem, it's accuracy is simply to see if the detected gestural character is same as the output word, for a single user. We call this within-user accuracy. On the other hand, IndexPen set contains as many as 29 symbols, and some characters similar to each other. Additionally, as the gestures simulates actual writing, the users may transfer his or her personal writing style into the context. Therefore, another way assess accuracy would be how closely do we need to follow the pen pilot strokes for our system to register as the correct letter. As a result, to design an accurate text-entry system, the system would need to output the correct letter without requiring too much effort from the user. This is defined as the cross-user accuracy.

In mutliclass classification problem like ours, there will always be false-positives and false-negtaives in the system. False-positives for our system would be the system detecting characters while the user is not actively writing anything. False-negatives would be when the system fails to detect the writing even though the user is actively performing the gestures required for the characters. Second fold of evaluating accuracy: * false-positives, the system detects a writing but the user actually did not write anything * false-negative, the system fails to detect the writing when the user, in fact, performed the gestures

Responsiveness A dependable text-entry system would have a smooth transitions between each characters. The importance of segmentation and out of distribution came into play when designing a real-time text-entry system. It is also important that the system registers and processes the user's input in a swift fashion so that the interaction flows effortlessly for the user. The system should not cause any inconvenience by making the user wait for each character to register before writing the next letter.

This is particularly important as the classification is being continuously performed over streams of data. The responsiveness would measure the time between when the user considers the gesture to be completed, to when the written character is detected.

Input Speed One of the most important aspects of designing a system is making it time efficient. The user should not have to wait after performing each character. The lack of speed could potentially cause the user would rather prefer to type on another device. We could measure the input speed by measuring the Words Per Minute from typing. We also need to make a design so that there are no False Positives as it would greatly reduce the input speed. The user would have to constantly delete the characters instead of inputting the characters. The standard QWERTY keyboard has an average of 40 words per minute and TipText (36) has an average of 11.9 words per minute. As a result,

How fast can the user type in characters with IndexPen

Learnability For this text-entry system to be learnable, the design should be intuitive for the users. We would also need to look at the system's consistency and naturalness. For consistency, we mean that the same input strokes performed by the user should output the same letter. If the user keeps performing the same stroke and our system outputs different letter, it would affect the learnability of the system. For naturalness, the user should be able to follow a line of instruction such as "Please write the letter on your thumb using your index finger" and perform the actions instantly without having to output extra effort to learn the system. Although the output is the same, everyone have different strokes when it comes to letter writing. As a result, for the naturalness, a reliable system would not care whether the strokes are different and map the final output.

ThuMouse

We list out the following design considerations for measuring ThuMouse's performance as a cursor device.

Accuracy To achieve satisfactory interaction experience, the processing pipeline must be able to map the finger movement to that of the cursor with high accuracy. Consider a traditional computer mice, it provides a diligent experience for navigating through the digital world. This particular interaction can be interpreted as that a user can, with physical interaction with the mouse, (1) move the cursor to where the user wants it to be (2) adjust the speed of the cursor based on needs (3) the cursor starts to relocate and stops when the user starts and stops moving.

The performance is boiled down to three types of accuracy: location, velocity, and synchronization. When using a cursor input device, the location accuracy is how precise the device can translate the its physical displacement to that of cursor. During the process of movement, the velocity accuracy can be measured by comparing the rate of change in displacements between the physical and the virtual; the discrepancy between the first derivatives of the displacements tells if the cursor is moving at the speed that the user intends. The synchronization deals specifically with the situation when the user has just commenced or ceased a movement. We can view it as that the two entities involved: physical device and the cursor can, at any time, be in one of the two states : moving or still. If state of one entity changes, how much time have elapsed till the other entity followed that change. In other words, synchronization accuracy measures the responsiveness for the mapping of states.

Smoothness A successful tracking device should be free of any perceptible glitches. That is, user may not experiences moments when the cursor stops for a time then jump to another location. The tracking pipeline needs to issue displacement in a succession of fixed and short time intervals. On the other hand, the device should allow user to navigate the digital environment easily with error-proofing interactions. In the case of a computer mice, if there is a target needs to be clicked, most of us are able to trace our way towards the target with a straight line, thank to the nature mapping between mice movement and that of the cursor.

To the design of new tracking devices, being smooth implies that the device must be capable of continuously detecting and mapping user's input with high enough frames per second (FPS). In addition, tracking is different from classifying gestures because of the flexibility it affords. In actuating this flexibility, the designer should careful with the way that the user gives inputs. The input method (i.e., gesture) should not allow to much space for making errors and the system needs to have a fairly deterministic mapping between the actions in the physical and the virtual world.

Learnability To build an intuitive design, the learning curve of the system must be low. A learning curve is a hypothetical graph of how much time and effort required for a person to invest before our tool become productive to them. Having a low learning curve would allow the users to spend time using the system as opposed to having the users learn the system for hours on end. The two most important factors for developing a easy-to-use system are consistency and naturalness of the input techniques. To achieve consistency and naturalness, a good system would map the movement of the cursor to whichever direction the user moves their thumb. If the user moves to the left, the cursor will follow the same path and same direction. Affordance looks at the perceived and actual properties of a thing that determines how the thing is to be operated. An ideal system would make have the same perceived and actual affordances and would have no surprise functionality. With the natural mapping as well as the consistency of the method, we hope to achieve a learnable and intuitive system on which the future cursor based input systems could depend upon.

To understand competence of the proposed system, we device quantitative measures for each of the above design considerations. The data is collected in a two-stage user study and the result is presented in section **??**.

3.5 Gesture Sensing

The software gesture pipeline starts with detected points, which represent the objects front the radar. The points are extracted by the pipeline explained in the last section and from there, our system extracts finer features of the dynamic gestures. In this section, we cover the pre-processing steps in which the observed points are transformed into deep learning features. We also discuss the deep learning model that analyze the pre-processed feature and how the gesture schemes are actuated. Besides, we discuss two different methods – YOLO and Leap Motion as our input labels.

3.5.1 Preprocessing

The preprocessing step is to transform the detected points into suitable machine learning features of which a neural networks can study. The points are first clustered and filtered, by doing so algorithms after this point focus only on the hand that is performing the gesture. The filtered points are then rasterized in a 3D voxel space, forming a 3D feature array,

The pre-processing is performed every time the radar resolves a point cloud data collection through the detection procedure explained in the last section; we call this data collection a radar frame. A frame at time t is consisted of n detected points, defined as $n \times 4$ matrix; each row is the Cartesian coordinates and Doppler (velocity) of the detected points. The number n may vary across frames depending on how the resolvability conditions from section 3.1.1 are satisfied. We can refer to the matrix as the point array as each row denotes a detected point.

$$p_t = \begin{bmatrix} x_0 & y_0 & z_0 & doppler_0 \\ x_1 & y_1 & z_1 & doppler_1 \\ ... & & & \\ ... & & & \\ x_n & y_n & z_n & doppler_n \end{bmatrix} \tag{10}$$

Clustering: Similar to the static clutter removal at signal-level by equation (7), ThuMouse further removes dynamic noise in point-level using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The algorithm identifies high density areas and expose out-lier; in ThuMouse settings, we define that there must be at least 3 points to form a cluster and two points need to be at most 20cm apart to be considered as in the same cluster. These parameters are picked based empirical observations in our experiments and shown to perform well in determining the noises and the 'core of gesture'. For readers interested in the DBSCAN algorithm, we refer this paper (30) by Sander et al. It explains the detail of the algorithm. We define the point array after applying DBSCAN as $P_{filtered}$.

We define the gesture performing area as being within the radial range of R_{bound} meters relative to the radar. R_{bound} is depending on the gesture scheme implemented with the pipeline. For mobile usage, it is reasonable to set R_{bound} at 0.25m. We create a bounding volume: $x, y, z \in [-R_{bound}, R_{bound}]$ around the point cloud to filter out any points that lie outside the specified range of the radar. To prepare for Voxelization (the next step in pre-process), the spatial coordinates in $P_{filtered}$ needs to be within the range of 0-1. Therefore, the xyz value are the min-max normalized in the bounding volume with:

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}, y_i = \frac{y_i - y_{min}}{y_{max} - y_{min}}, z_i = \frac{z_i - z_{min}}{z_{max} - z_{min}} \tag{11}$$

where the minimal and maximum values are given by the bounding volume:

$$x_{min} = y_{min} = z_{min} = -R_{bound} x_{min} = y_{min} = z_{min} = R_{bound} 3.1.1 \tag{12}$$

3.5.2 Voxelization

As the last step in pre-processing, we rasterize $P_{t_{filtered}}$ into a $(25 \times 25 \times 25)$ voxel, defined as $Voxel_t$. This procedure is necessary as the subsequent convolutional feature extractor only take inputs with fixed dimensions, and the points array is not acceptable because of the variable number of rows. On the other hand, convolutional Neural Network (CNN) has shown great success in extracting and representing point cloud data if put in voxel form (18). Because feature $P_{filtered}$ at this stage is essentially in point-cloud format, we take advantage of the convolutional network structure by voxelizing the detected points.

The unit for the three axis are in meters, with Pt being Min-Max normalized between 0 and 1, the grid of the voxel has the resolution of 1 centimeter. Additionally, the forth column of $P_{filtered}$ is the velocity (doppler) of each observed points. Different from typical point-cloud voxelization methods (20), this velocity information is treated as the heat, or color of each voxel. Overlapping points are added to the voxels incrementally; i.e., locations where there are more objects moving would become a hotter spot. The resulting volume can be interpreted as a 3D velocity heat map, or 3D graph with a single color channel: velocity. The characteristic of this features is effectively the same as that of a regular 2D image: the distribution of the hot spot is non-linear.

3.5.3 Data augmentation

Data augmentation methods helps to increase the amount of relevant training data without physically collecting more. Moreover, it is shown that CNN can benefit from data augmentation to become more robust (21). In this study, we borrow data augmentation methods that was brought up in (25) and (20) to . Namely, a 3-fold data augmentation is applied on the detected points to form new samples; the labels (ground truth) for augmented is the same as its none-augmented counterparts. The three augmentation techniques being performed are: translation, scale, and rotation, and they are applied to $P_{filtered}$ before the voxalization. **Translation** changes the spacial coordinates of the detected points. It is for simulating an added and small Gaussian noise to the data. In other words, each points is translated by a small displacement; the amount of displacing is defined by the noise distribution has a mean of 0 meter and a standard deviation of 0.02 meter. The numbers are derived from empirical observation for our study case for the noise can not be too large to twist the original data-set in a substantial way nor can it be so small that its augmentation value is barely noticeable.

Scale linearly alter the coordinates if the points along a the x, y, and z axis. The factor used for scaling is the same normal distribution used in translation. Scale is effectively the same as Translation for it also changes the location of the each points in $P_{filtered}$, but instead of introducing noise, it is meant for simulate person with differently shaped hands than those of the participants.

Rotation is to cover the case where participants may perform the gesture at varying tilted angle. Although the subjects are instructed to perform the gesture in a certain relative position to the radar, the angle of the hand is not strictly imposed as per real-life scenarios. The rotation is applied along the three Cartesian axis, and the amount of rotation is follows the same distribution as in the translation and scale.

3.5.4 ThuMouse Neural Networks

Dynamic gestures have temporal, as well as spatial characteristics. Therefore, we decided that the network model the dynamic profile of a gesture needs to contain (1) convolutional layers that extract the non-linear features of each radar frame, (2) LSTM cells that retain the features from the frames in a time regressive manner, (3) dense layers as output that are adjustable based on given gesture scheme. Moreover, in order to run the gesture system in real-time, the network should also be lightweight and low latency for smoother user experience.

To meet the above requirements, we design the following neural network model as we shown in Fig. 8.

Algorithm 1: Data augmentation algorithm applied to Detected Points

```
Result: an augmented sample $P_{aug}$ from $P_{filtered}$
if $translate$ then
    for $pinP_{filtered}$ do
        $translate(p, amount = Gaussian(\mu = 0.0, \sigma = 0.02))$
    endfor
endif
if $scale$ then
    ScaleX(P$_{filtered}, factor = Gaussian(\mu = 0.0, \sigma = 0.02)$)
    ScaleY(P$_{filtered}, factor = Gaussian(\mu = 0.0, \sigma = 0.02)$)
    ScaleZ(P$_{filtered}, factor = Gaussian(\mu = 0.0, \sigma = 0.02)$)
endif
if $rotate$ then
    RotateX(P$_{filtered}, factor = Gaussian(\mu = 0.0, \sigma = 0.02)$)
    RotateY(P$_{filtered}, factor = Gaussian(\mu = 0.0, \sigma = 0.02)$)
    RotateZ(P$_{filtered}, factor = Gaussian(\mu = 0.0, \sigma = 0.02)$)
endif
```

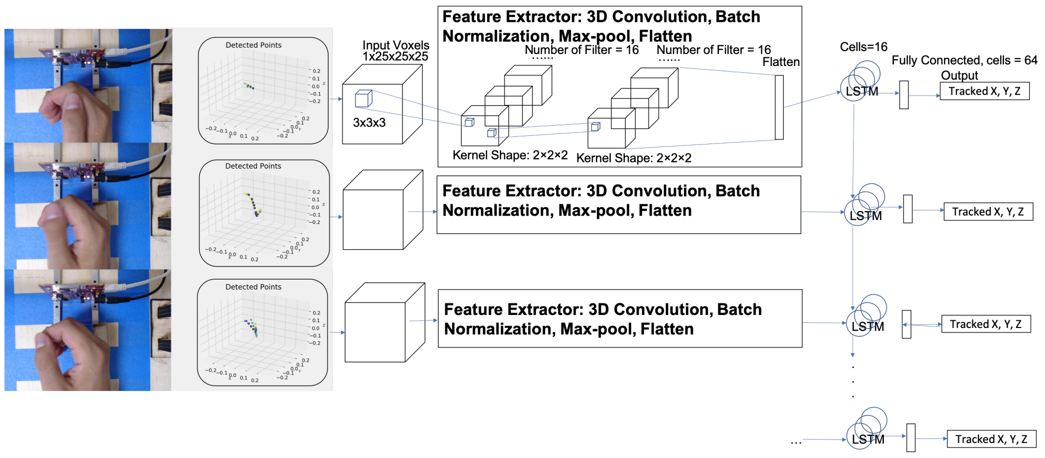


Figure 13: ThuMouse CRNN Architecture: the input layer reads the voxalized detected points from the mmWave sensor; 3D convolution is then performed onthe 3D volumes to produce the feature map which is feed into LSTM layers. LSTM cells propagate information into fully connected layers and outputs thex, y and z as tracked position of the

In contrast to work using the Range-Doppler profile as input (34), ThuMouse's network model takes in the voxalized detected points, which include the x, y, z, and velocity. The detected points can represent the tendency of motion better which allows a shallower and lighter model to be implemented without loss in performance.

The input of the network is the voxalized points aforementioned with the shape $(25 * 25 * 25 * 1)$. The convolutional layers act as feature extractors to initially interpret the spatial features of a radar frame; it includes 3D convolutional, batch normalization, max-pooling layers, and concluded with flatten layers. To avoid the common problem of 'dead neurons' in convolutional layers, we use the $leaky_{Relu}$ activation function for layers.

The condensed features then go through one layer of LSTM that regressively looks back to the previous 20 timesteps, (corresponding with the number of frames the system receives per second). The model culminates with a fully connected layer where it gives the tracked position of the thumb tip in its spatial coordinates (x, y and z).

In order to decrease the over-fitting effect from trained neurons, we need to make the system more applicable to generalized situations, such as in the case of a new user. To do so, we randomly drop nodes trained in the system. With it, we reduce the output dependency from certain features. We apply the common practice of dropout here and the rate being applied to both the LSTM and fully connected layers is 0.5.

3.6 DFT Neural Networks

With the sensor-signal processing pipeline explained above, the physical information of the objects detected by the radar is represented in range-doppler, and range-azimuth-elevation profiles. As the data streams in, the sequence of the two profiles forms a dynamic profile that characterized by the gesture performed.

In this section, we cover the multi-input multiple-output (MIMO) deep learning model that we constructed. The model extracts high-level features from the two types of profiles aforementioned and output both the IndexPen classification and ThuMouse tracking. We design the network structure with the following considerations.

It has been shown that Convolutional Neural Networks (CNN) are well-suited for distilling two-dimensional features such as images. CNN captures the spatial feature of the image-like profiles and is able to extract non-linear, and high-level features such as whole finger motion, the relative location of specific hand parts, and so on.

Meanwhile, the input contains two profiles and the model should first merge them in some way. In solving similar problems, past studies have added the channel dimension and put the multiple images as different channels(35). For this system, it is also important to note that the relevant features for gesture detection in range-doppler and range-azimuth-elevation can differ greatly as the former reflect the radical velocity, and the range-azimuth-elevation is the spatial information. This fact led us to use two different CNNs for the two profiles, and they do not depend on each other.

The extracted feature maps from the two profiles is flattened and concatenated before being sent to the time distributed layers. As the features are coming in as sequences, time has to be taken into account. That is, the model needs to look at data in a time interval to make predictions. As traditional neural networks suffer from the exploding/vanishing gradient problem when dealing with sequenced data, we use long-short-term-memory (LSTM) layers to solve the problem. LSTM has the capability of gating the information, giving the model another level of flexibility to decide what temporal features are important for the gesture detection.

Indeed, prior work (34) showed that Convolutional Recurrent Neural Networks (CRNN) with LSTM cells perform well in resolving the range-doppler profile in gesture classification. We extend this idea to that of the range-azimuth-elevation profiles as second input for the network. After extracting the spatial and temporal features, the system take to the standard Fully Connected (FC) layers. We make a copy of the extracted features and put them through two different FC layers that outputs the *ThuMouse* tracking and *IndexPen* classification predictions, respectively.

It is important to note that the design uses the same spatial and temporal layers for two gesture applications. The output does not diverge into *IndexPen* and *ThuMouse* until the second-to-last layer. It was considered to use two entirely separate models, as having a model fine-tuned to a specific gesture application is likely to yield higher accuracy. However, synchronization of *ThuMouse* and *IndexPen* is crucial as they are integrated parts under *DFT*. Additionally, because of the similarity between the two gestures, the detecting model may duplicate the feature extractors for each, taking advantage of transferred learning.

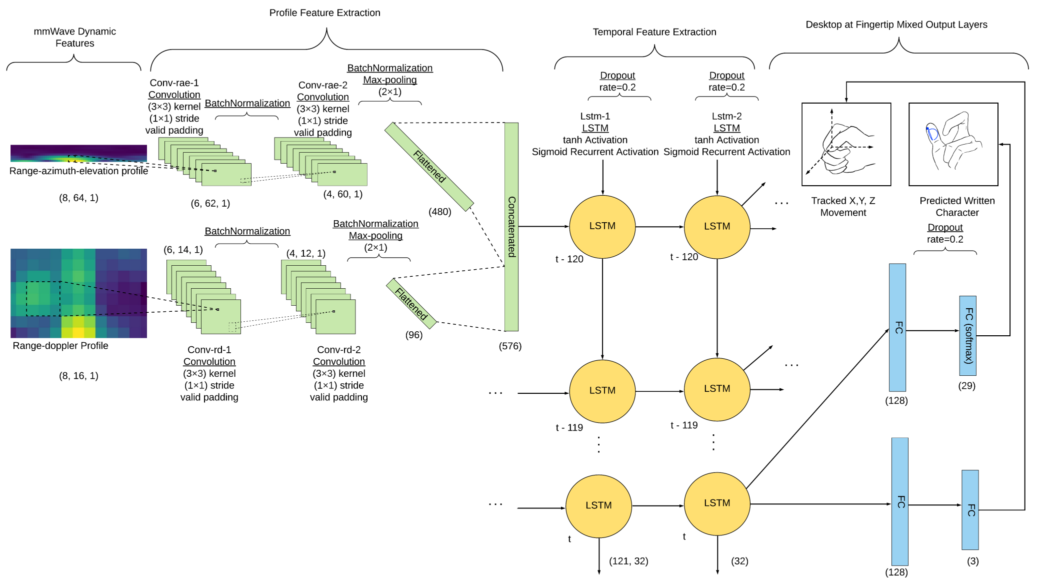

mmWave Dynamic Features
Profile Feature Extraction
Temporal Feature Extraction
Desktop at Fingertip Mixed Output Layers
Conv-rae-1 Convolution (3×3) kernel (1×1) stride valid padding
BatchNormalization
Conv-rae-2 Convolution (3×3) kernel (1×1) stride valid padding
BatchNormalization Max-pooling (2×1)
Range-azimuth-elevation profile
(8, 64, 1)
(6, 62, 1)
(4, 60, 1)
Flattened
(480)
Concatenated
(576)
Range-doppler Profile
(8, 16, 1)
(6, 14, 1)
(4, 12, 1)
Flattened
(96)
Conv-rd-1 Convolution (3×3) kernel (1×1) stride valid padding
Conv-rd-2 Convolution (3×3) kernel (1×1) stride valid padding
Dropout rate=0.2
Lstm-1 LSTM tanh Activation Sigmoid Recurrent Activation
Lstm-2 LSTM tanh Activation Sigmoid Recurrent Activation
LSTM
t - 120
t - 119
t
(121, 32)
(32)
Tracked X,Y, Z Movement
Predicted Written Character
FC
FC (softmax)
(128)
(29)
(3)

Figure 14: DFT CRNN Architecture: the two inputs include the profiles for range-doppler and range-azimuth profiles. 2D convolution is then performed on the each of the profile feature map which is feed into LSTM layers. LSTM cells further extract the temporal information and send copies of the their output to the ThuMouse and IndexPen FC layers which give the final prediction of the gesture detected.

Computer Vision as Training Base Inspired by Huang et al.(12), we use a double input apart from the mmWave sensor that we are experimenting with. However, our design of the gesture disallows the presence of another device in its performing area. We evaluated a number of alternative ground-truth observers such as an accelerometer mounted on the thumb. But any wearable device will introduce bias in the signal shape received by the radar device.

Our solution for this problem is to use webcams and the radar as dual inputs in recording the thumb's movement. Compared with many other options, a camera has the advantage of being a physically unintrusive observing device and the recent advancement in video-based convolutional neural networks promises high accuracy. It can help us evaluate the radar's tracking performance by analyzing the outputs from each method. Even though there exists time discrepancy between the camera's and radar's system, the time-stamps of radar are fully included in camera's time-stamps. Therefore, we choose the camera's tracking as ground truth reference for radar's tracking.

3.7 Ground Truth Collecting

Ground-truth in our system refers to the position of the thumb. We want the true position of the thumb so that we can train our network. In order to achieve the true position, we use webcams with YOLO algorithm to yield the ground-truth.

3.7.1 YOLO Detection

We have two iterations for this project. During the first iteration, we used YOLO for ground truth labelling. To get the location information of thump tip in each camera frames, we use the YOLO(27). YOLO is CV-based object tracking algorithm that gives the "bounding box" of the detected objects. We have to manally label the ground truth with this approach. In the second iteration, we use the interactive LeapMouse that auto-labels the ground truth for us. We can evaluate the performance of the ThuMouse tracking using the ground truth systems.

The method is as follows: a YOLO model that identifies the position of the fingertip is pre-trained with 750 images from 3 participants from the research group (each select 250 images) with 300 epochs, nearly 20k steps. We observed that the model thus trained performs reasonably well in noting where the finger tip is from camera frames.

During a data collection session for ThuMouse, we record the radar frames along with the two cameras. Then we use the trained YOLO model to process recorded video frames and obtain the location information of the fingertip. Because the frames from both radar and camera are recorded simultaneously, we can take the location information of finger-tip from the cameras as the ground truth for the radar's estimation. It is possible that the radar's timestamp does not match the frames of the camera due to the fact that the radar operates at 20 FPS and camera at 30 FPS. Since the goal is to get the location of the finger for every radar frames, if the timestamp of a radar frame lays in between the time of two camera captures, we linearly interpolate the positions given by the two photos to get the location of the fingertip at the time when that radar frame is recorded.

The Yolo model is trained with 750 images from 3 users (each select 250 images) with 300 epochs, nearly 20k steps. We use this model to predict on recorded video frames and note the location of the fingertip as given by the CV model.

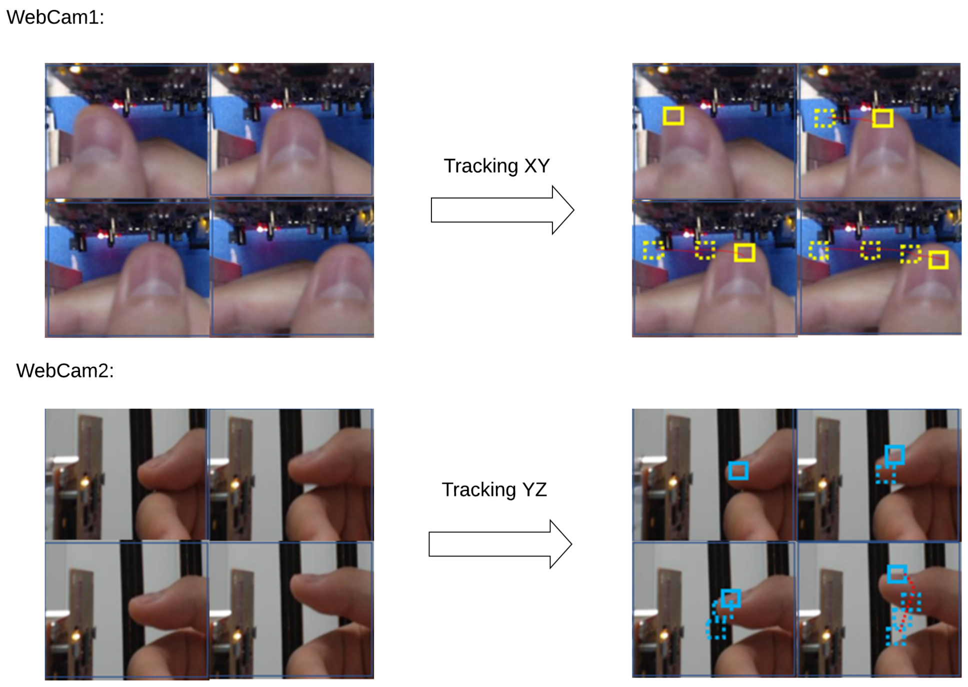


Figure 15: YOLO Sample: YOLO algorithm is used to predict thumb tip's location with the top and side cameras. The tracked box produced forms the ground-truth for evaluating the radar tracking performance.

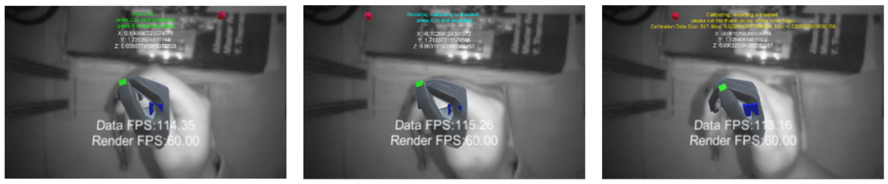
By comparing the YOLO's tracking of the fingertip and the radar's, we can test and evaluate the performance of the trained CRNN model.

3.7.2 Leap Motion Detection

Previously, we used YOLO to detect object and there were some disadvantages that comes with it. First, the YOLO algorithm relies on the camera's performance, sometimes the camera will lose focus if rubbing too fast. It becomes hard to predict accurate results as ground truth system is too blurred to train network.

Secondly, the Camera's position will highly influence the results of (x,y,z). Due to the 2 cameras we used to predict x-y and y-z, little movement of the whole system will cause the the output of x,y,z inaccurate enough and thus making the performance of network bad. So, now we introduce a new device named Leap Motion. It detects user's hand and simulate a 3D output shown in desktop.

There are two major benefits comparing Leap Motion with YOLO. Because Leap Motion could simulate one 3D model with one infrared camera while YOLO requires for 2 cameras to give (x,y,z), using Leap Motion could reduce the effect of cam's movement and match the corresponding (x,y,z). Secondly, Leap Motion could simulate the hand model in real-time quicker then using YOLO. So it would help us to use Leap Motion's prediction in (x,y,z) as the baseline of the thumb tricking and compare the performance of the prediction from radar's data.



```
Operating...
press C to start calibration
press R to start recording
X: 0.100680723074079
Y: 1.72539248317748
Z: 0.00697748968377709
Data FPS:114.35
Render FPS:60.00
```

```
Recording, calibrating is disabled.
press R to end recording.
X: -0.10204124301572
Y: 1.71307311579586
Z: 0.00317163996584257
Data FPS:115.26
Render FPS:60.00
```

```
Calibrating, recording is disabled.
please rub the thumb on top of the index finger.
Calibration Data Size: 547; Max: 0.0225543977896894; Min: -0.022632881990158
X: -0.0615766308456659
Y: 1.72940614615627
Z: 0.00022539128258587
Data FPS:113.16
Render FPS:60.00
```

Figure 16: Real time Leap Motion model in Unity. The Green box represent the central of thumb tip. Besides, we set the reference point at the end of index finger because it has less movement than other point and much easier to detect. (a)This is the Operating interface for data collection, user would test to show their hand(right hand) under the leap Motion and adjust their movement or position for later collecting; (b)This is the Recording interface, once the collection is started, the output label of x,y,z will be present on the screen and be recorded later; (c)3rd figure here represent the calibrating function, users would be asked to keep rubbing and the system would record user's max z-value and later be regarded as the threshold for telling clicking or not.

3.8 Second Iteration Neural Networks

With the sensor-signal processing pipeline explained above, the physical information of the objects detected by the radar is represented in range-doppler, and range-azimuth-elevation profiles. As the data streams in, the sequence of the two profiles forms a dynamic profile that characterized by the gesture performed.

In this section, we cover the multi-input multiple-output (MIMO) deep learning model that we constructed. The model extracts high-level features from the two types of profiles aforementioned and output both the IndexPen classification and ThuMouse tracking. We design the network structure with the following considerations.

It has been shown that Convolutional Neural Networks (CNN) are well-suited for distilling two-dimensional features such as images. CNN captures the spatial feature of the image-like profiles and is able to extract non-linear, and high-level features such as whole finger motion, the relative location of specific hand parts, and so on.

Meanwhile, the input contains two profiles and the model should first merge them in some way. In solving similar problems, past studies have added the channel dimension and put the multiple images as different channels(35). For this system, it is also important to note that the relevant features for gesture detection in range-doppler and range-azimuth-elevation can differ greatly as the former reflect the radical velocity, and the range-azimuth-elevation is the spatial information. This fact led us to use two different CNNs for the two profiles, and they do not depend on each other.

The extracted feature maps from the two profiles is flattened and concatenated before being sent to the time distributed layers. As the features are coming in as sequences, time has to be taken into account. That is, the model needs to look at data in a time interval to make predictions. As traditional neural networks suffer from the exploding/vanishing gradient problem when dealing with sequenced data, we use long-short-term-memory (LSTM) layers to solve the problem. LSTM has the capability of gating the information, giving the model another level of flexibility to decide what temporal features are important for the gesture detection.

Indeed, prior work (34) showed that Convolutional Recurrent Neural Networks (CRNN) with LSTM cells perform well in resolving the range-doppler profile in gesture classification. We extend this idea to that of the range-azimuth-elevation profiles as second input for the network. After extracting the spatial and temporal features, the system take to the standard Fully Connected (FC) layers. We make a copy of the extracted features and put them through two different FC layers that outputs the *ThuMouse* tracking and *IndexPen* classification predictions, respectively.

It is important to note that the design uses the same spatial and temporal layers for two gesture applications. The output does not diverge into *IndexPen* and *ThuMouse* until the second-to-last layer. It was considered to use two entirely separate models, as having a model fine-tuned to a specific gesture application is likely to yield higher accuracy. However, synchronization of *ThuMouse* and *IndexPen* is crucial as they are integrated parts under *DFT*. Additionally, because of the similarity between the two gestures, the detecting model may duplicate the feature extractors for each, taking advantage of transferred learning.

4 Evaluation and Results

For the evaluation and results, we have three iterations at which we ran the project. At this point, we are currently at the third iteration. The first iteration of Thumouse is when started to shape the idea by actually conducting experiments and the second iteration is the refining of the idea. The third iteration is a stage for developing our own GUIs and conduct user-studies.

4.1 First Iteration

Initial evaluation on the proposed system is conducted and we hereby present the preliminary results and findings. Here we discuss how the two proposed gesture schemes perform. We present a quantitative evaluation where we analyze the tracking effectiveness of ThuMouse and classification accuracy of IndexPen by showing the efficacy of the system on the validation set.

4.1.1 Experiment Environment

IndexPen: In recording the data for IndexPen, the subject would write each character during a fixed interval; at 20 FPS, the interval is set at 4 seconds as we find this is a typical time to perform one writing with IndexPen. Each character is repeated twice in succession, which generate writing 20 samples per loop. The process is repeated 3 times to produce a group of 60 samples; we define this as a session. Each subject is asked to perform two sessions in one sitting and we collect 10 sessions per participant. The number of samples thus produced is 30000 (10 characters$\times$60 repetition per loop$\times$10 session per subject$\times$5 subject). Since the order of writing characters from the alphabet is pre-determined, each gesture sample is label accordingly.

ThuMouse tracking dual input: Assigning the true x, y, and z location (ground-truth) to the radar frames is more challenging to carry out. To achieve frame-level tracking, the absolute coordinates of the thumb tip needs to be obtained as the ground truth with fits to the model's output layer.

We are inspired by Huang et al.(12) to use a double input apart from the mmWave sensor that we are experimenting with. However, the method of (12) does not directly apply here; our design of the gesture disallow the presence of another device in its performing area. The team evaluated a number of alternative ground-truth observers such as an accelerometer mounted on the thumb. But any wearable devices will introduce bias in the signal shape received by the radar device.

Our solution for this problem is to use both a Webcam and the radar as dual inputs in recording thumb's movement. Comparing with many other options, a camera has the advantage of being non-physically-intrusive observing device and the recent advancement in video-based convolutional neural networks promises high accuracy. It can helps us evaluate radar's tracking performance by analyzing the outputs from each method. Even though there exists time discrepancy between the camera's and radar's system, the time-stamps of radar are fully included in camera's time-stamps. Therefore, we choose the camera's tracking as ground truth reference for radar's tracking.

To get the location information of thump tip in each camera frames, we use the YOLO(27). YOLO is CV-based object tracking algorithm that gives the "bounding box" of the detected objects. With it, we can evaluate the performance of the ThuMouse tracking architecture.

ThuMouse needs to able to resolve the how much the cursor should move at each radar frame, given that the radar is capturing detected points at 20 FPS, manually labeling the data is out of the question, nor could we do the same approach as we did for IndexPen because there is no pre-defined gesture set.

The method is as follows: a YOLO model that identifies the position of the fingertip is pre-trained with 750 images from 3 participants from the research group (each select 250 images) with 300 epochs, nearly 20k steps. We observed that the model thus trained performs reasonably well in noting where the finger tip is from camera frames.

During a data collection session for ThuMouse, we record the radar frames along with the two cameras. Then we use the trained YOLO model to process recorded video frames and obtain the location information of the fingertip. Because the frames from both radar and camera are recorded simultaneously, we can take the location information of finger-tip from the cam's as the ground truth for the radar's estimation. It is possible that the radar's timestamp does not match the frames of the camera due to the fact that the radar is at 20 FPS and camera at 30. Since the goal to get the location of the finger for every radar frames, if the timestamp of a radar frame lays in between the time of two camera captures, we linearly interpolates the positions given by the two photos to get the location of the fingertip at the time when that radar frame is recorded.

According to section 3.1.1, the mmWave sensor only detects objects that are moving in relative to the sensor itself. We decided it is more feasible for the system resolve the displacement ($delta_x$, $delta_y$, $delta_z$ for displacement along the three axis) of the fingertip between frames instead of giving the absolute position.

By comparing the cam's real-time variation of fingertip and the radar's predicted variation, we can test the performance of the trained CRNN model.

Layer Type	Filters	Size/Stride	Output
Input Image	/	608*608	/
Convolutional	32	3*3	224*224
Maxpool	/	2*2/2	112*112
Convolutional	64	3*3	112*112
Maxpool	/	2*2/2	56*56
Convolutional	128	3*3	56*56
Convolutional	64	1*1	56*56
......			

=

The time-stamps of radar can be extracted from recorded files and each time-stamp could find between two frames of the camera. According to the location information of the detected point and coherence of testing movement, we can calculate the detected thumb tip for each radar frame. And because the radar system reflect the real-time fluctuation in points, we store the variation of each point from its last time-stamp and train the CRNN model with these changes to predict real-time points movement through radar-frames.

To evaluate the performance of the system in matrix units, all trials are carried out with the hand at the same relative position to the top camera. Doing so ensure the displacement of the finger per pixel of camera-frame is consistent across experiment sessions.The values includes the distance from above camera and hand, resolution of the cam1 and cam2 and the angle of views. As shown in figure below, the sensors are set up in a specific way so that we can collect data from the optimal location.

However, the dual input of radar and webcams still have some problems. Camera sometimes loses focus, which will affect the accuracy of the results, and camera's position will highly influence the results of (x,y,z). At the same time, it always takes time to train the YOLO model firstly before training the ThuMouse model,Greatly reducing experimental efficiency.

4.1.2 Experiment Setup

The overall experiment environment is set up in a way that the radar is always at a same relative position to the cameras (important to ThuMouse as will be discussed in section XX). It also can be used for both IndexPen and ThuMouse, of which the later utilizes the webcams as suppliers of ground-truth. While the system collecting radar frames, video streams from the two webcams, one is above the hand to detect the x and y (cam1) and the other one is placed to detect the y and z (cam2), which are then feed into the YOLO framework to resolve the true x, y, z position of the thumb tip.

$$Res_{cam1Y} = 400 pixels \qquad Res_{cam1X} = 600 pixels$$

$$D_{cam1} = 10cm \qquad AoV = 78^\circ$$

By using these values, we get:

$$\begin{aligned} P_{resX} &= \frac{2 \times D_{cam1} \times cos(AoV/2)}{Res_{cam1X}} = 2.70mm/pixel \\ P_{resY} &= \frac{2 \times D_{cam1} \times cos(AoV/2)}{Res_{cam1Y}} = 4.05mm/pixel \end{aligned} \tag{13}$$

P_{resX} and P_{resY} will be used in studying the discriminating capability of the ThuMouse.

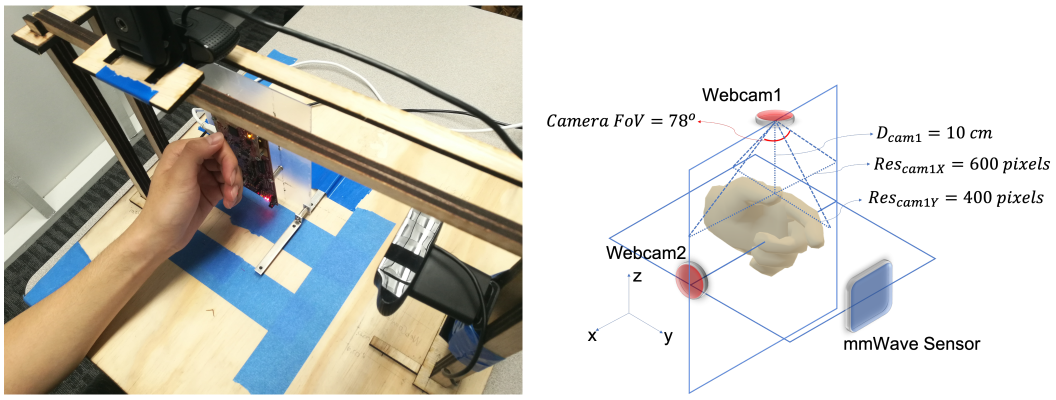


Figure 17: First iteration-experiment setup, one camera is mounted on the top shaft; second camera is located at the side. Middle.

4.1.3 Results

Here we present the validation result to show the tracking capability of ThuMouse and classification accuracy of IndexPen.

IndexPenThe IndexPen gives a total accuracy of 93%. The classification result for each of the five gestures in the alphabet is given by the confusion matrix in figure

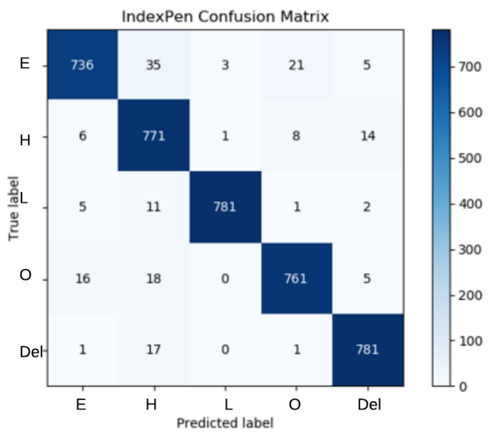


Figure 18: Confusion Matrix of IndexPen classifier: the numbers in the matrix shows how much the classifier is predicting the wrong label. The near-uniform distribution along the diagonal signifies that the model is performing equally well on all gestures from IndexPen alphabet

ThuMouse Overall, ThuMouse performs well in resolving movement of the thumb-tip on the rubbing on the surface of the index finger. For table 1 and figure 9-11, it is evident that the general trend of radar tracking predictions correspond to that of the camera's. In terms of resolving the XYZ displacement of the thumb-tip, the system displays reasonably good efficacy for the x-y plane (Mean Squared Error (MSE)$_{xy}$=16.4 px=1.35 millimeter). The competence along the z-axis, however, is wanting, which can be made a subject for future study to address. At present, we may conjecture that the lack of discriminating power for z is due to the fact that the signal strength drops more sharply along the inclination than it does along the azimuth, resulting in a smaller inclination FoV.

Table 2: System performance along each axis. The last row shows the statistics for x-y combined as a plane

	Mean Squared Error	Standard Deviation
X	9.23 px (1.27 mm)	3.03 px (1.12 mm)
Y	23.6 px (1.44 mm)	4.86 px (1.20 mm)
Z	64.4 px	8.02 px
XY	16.4 px (1.35 mm)	4.05 px (1.16 mm)
Overall	32.4 px	5.69 px

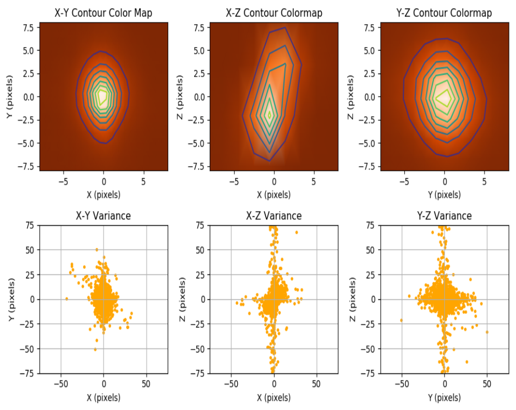


Figure 19: Left:X-Y Contour Color Map;Middle:X-Z Contour Color Map; Right:Y-Z Contour Map. From the figure, the XY contour map is most stable and gather almost in one area

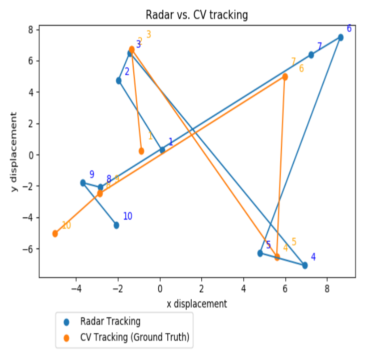


Figure 20: Radar tracking vs. CV tracking: the chart shows 10 consecutive tracking result on the x-y plane. The number in the graph are the index in time.

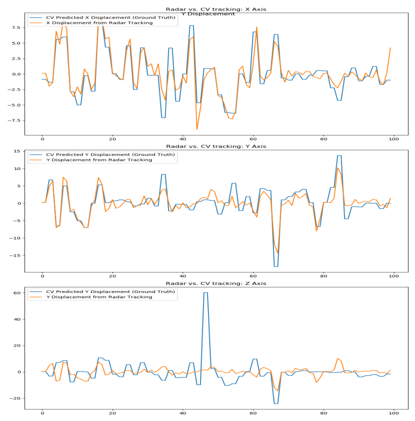


Figure 21: Radar tracking vs. CV tracking: the charts shows 120 consecutive tracking along x, y and z axis.

4.2 Second Iteration

For this second iteration, we completely changed the hardware and the groundtruth system. Instead of IWR6843ISK and Webcams, we use IWR6843AOP and Leap-Motion for both Thumouse and IndexPen. Here, we also came up new refined versions of Thumouse and IndexPen.

4.2.1 Experiment Environment

IndexPen To explore the accuracy of text entry with *IndexPen*, we collected a large dataset where an individual performed the 29 gestures. The goal was to explore the accuracy with which our system could classify the 29 gestures.

An individual would write each character during a fixed interval; the interval was set at 4 seconds as we find this is a typical time to perform one writing with *IndexPen*. Each character was repeated ten times in succession, which generates 10 writing samples per loop. The process is repeated 5 times, each time with a different letter, to produce a group of 50 samples (10 of each letter); we define this as a session. We collected 200 samples per character. As there are 29 classes (*A* to *Z*, *Space*, *Delete*, and *Enter*), the total number of samples produced is 5800 (29 $classes \times 50$ $repetition\ per\ loop$). Each sample is consisted of 121 frames (4 $sec\ per\ sample \times \frac{1}{30ms}$ $frames\ per\ second$). Therefore in total, the study collected 708,100 (5800 $samples \times 121$ $frames\ per\ sample$) frames of radar profiles over all 29 classes.

ThuMouse The evaluation of ThuMouse extends the previous implementation to make it more practical for real-world interaction, bringing the application-level in actuating the control of the cursor. For this, we first created a user interface for exploring and calibrating the ThuMouse tracking information to an on-screen interface. We also performed a replication study to explore the accuracy of the ThuMouse tracking, but using a different source (LeapMotion) as the ground truth. LeapMotion is a new approach to work as the ground truth of ThuMouse. It detects user's hand and simulates a 3D output shown in desktop. The detailed information about Leap Motion is in 3.4.2. As a tracking device, it is vital that the neura l network is trained with proper labels. The previous work (9) took the approach of using webcams and CV for this end. One of the problems with using YOLO is that it does not provide a real-time feedback. We had to manually labeled the system so that we can test it against the radar.

To this end, we use the LeapMotion sensor to resolve the ground truth for tracking information. For a system that is intended to be an input devices, it is beneficial if its training data is also collected in an interactive manner. That is, the ground-truth system needs to provide real-time feedback, creating a mode of interaction. The system under study, when being trained, needs to approximate the interaction defined by the ground-truth.

LeapMotion automatically and accurately labels for tracking, thus making it a reliable ground truth. The sensor intakes all the movements of the hand as well as the fingers and outputs a model of the hand with all the coordinates printed above. With LeapMotion, the user can perceive the model of their hand on the screen in real-time. Data collection is much more natural compared to the computer vision approach in earlier work. We can visually intake information on whether we are performing the right gesture or not as well as whether the sensor is recognizing our hand or not. To explore the cursor control enabled by ThuMouse, we created a simple user interface where the user moved a red sphere on the screen. The red ball moves accordingly with the ThuMouse. For this system to work, we had to tune the system with specific parameters for each person. Upon confirming that the system works, we tested the radar with LeapMotion as a ground-truth.

4.2.2 Experiment Environment

The new environment is set up the same way above using radar and Leap Motion. The Leap Motion is fixed above the radar. The system collects radar frames and video streams from Leap Motion together to train the model. The new experiment setup is shown in the Figure below.

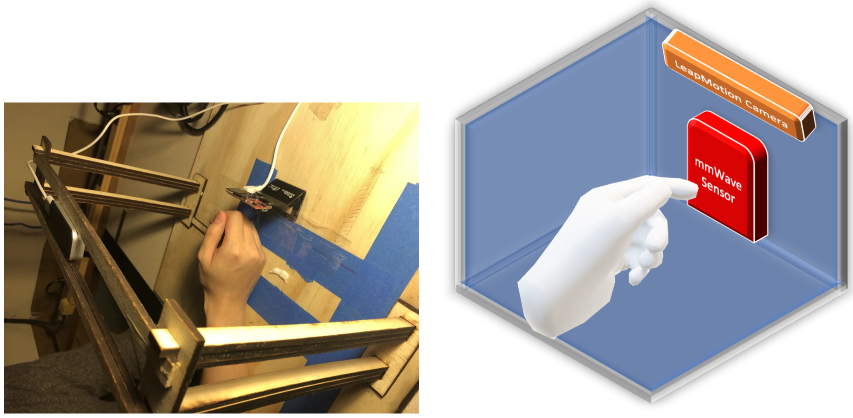


Figure 22: Second iteration-experiment setup, Leap Motion is above the radar

4.2.3 Results

IndexPen The confusion matrix for the *IndexPen* gesture classification is shown in Figure 23, which represents the prediction results for all 26 alphabet and three special commands (*Space*, *Delete*, *Enter*). Each row contains 200 prediction results for a particular character. Overall, we achieved 99.7% accuracy.

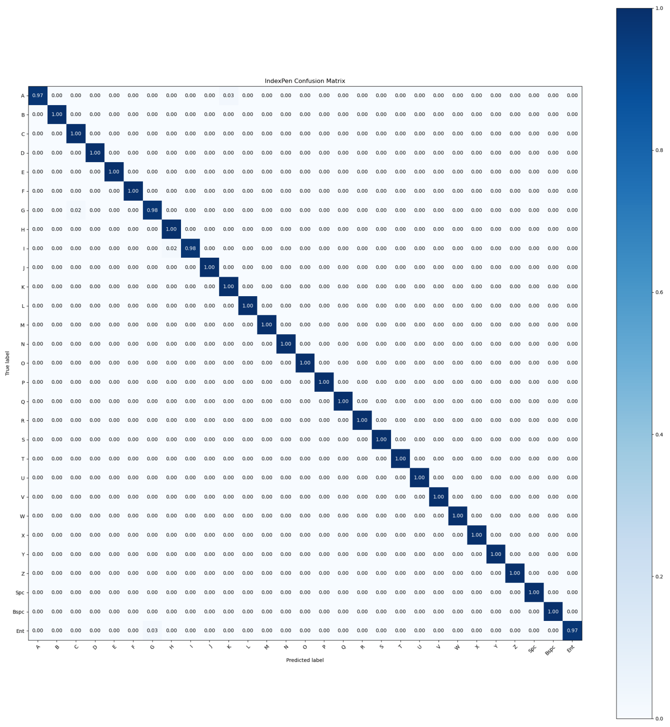


Figure 23: Confusion Matrix of IndexPen classifier: Alphabets on the vertical axes shows the actual input label, horizontal is the prediction from IndexPen classifier. The numbers in the matrix represent the percentage of labels predicted versus the true label. The high distribution on the diagonal shows that the IndexPen has high classification accuracy for all 29 gestures.

It is interesting to note for these gestural characters that did not achieve unit accuracy, they were consistently mis-classified to one other symbol. Namely, the pairs include A-K, G-C, I-H, Enter-G. One explanation for this behavior of uniform mis-classification is the similarity between the strokes of these pairs. As seen in the *IndexPen* gesture table, C and G only differs in the last horizontal swipe. Others pairs are more challenging to understand (i.e., Enter and G). However, it is important to note that the features given by the radar are the range-doppler and the range-azimuth-elevation matrices. Two gestures appearing similar as pixels does not necessarily imply that they also look similar in the radar features, and vice versa. Further, we expect that confusion matrices will be unique to an individual, based on the way they perform the gestures. As we test this with a wider group of users, we will explore this.

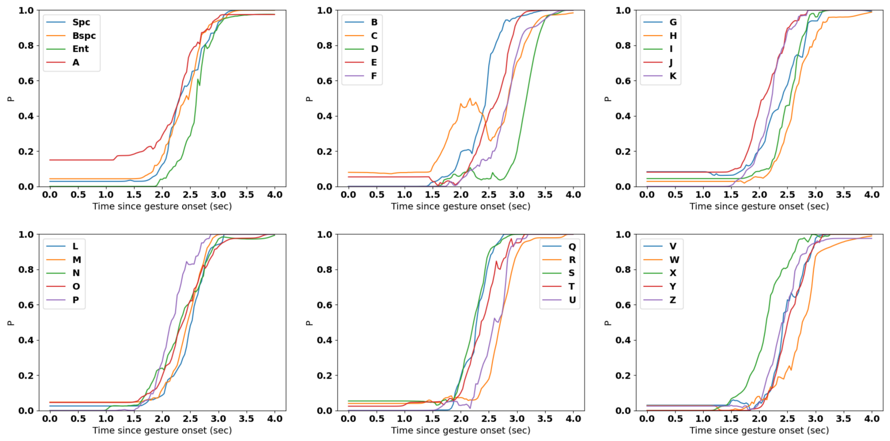


Figure 24: Averaged temporal probability evolution for the 29 IndexPen gestures.

As the classifier deals with streams of data, it is also important to see how the predicted probability evolves as a gesture is being performed. Figure 24 gives as the evolving predicated probability as the radar frames is being feed into the network while the user is performing the gesture. It is evident that all the probabilities for all the gestures shows a sudden surge upward at the 2nd second since the gesture onset. Moreover, the probabilities all tend to converge towards the end of the 3rd second. One implication of this behavior for real-time input systems is that the detection system should set a threshold value for determining if a gesture has actually being performed.

The high within-user accuracy shows high promise in bringing the system to the interaction stage. However, the result is not conclusive for all the samples are collected on a single user. The preliminary results shows that we can accurately detect *IndexPen* gestures of a user with a model trained on their own data. If the high within user accuracy holds true, it can be permissible to suggest that in real interaction settings, user-specific calibration can significantly improve the stability and precision of the proposed system.

Generally, ThuMouse performs well in resolving the position of moving objects in 3D space. The predicted path from the radar is generally with the real path, which is by the LeapMotion sensor. In terms of resolving the XYZ displacement of the thumb-tip, the system displays reasonably good efficacy for the x-y plane (Mean Squared Error $(\mathrm{MSE})_{xy}$=$7.55e^{-4}$millimeters px= $4.35e^{-4}$ millimeters). As the table above indicates, our mean squared error is significantly reduced by using LeapMotion as groundtruth compared to our WebCam results. RMSE of X is reduced from 1.27mm to 0.000435mm, RMSE of Y went from 1.44mm to 0.000975mm. The advantages of having a better system are not only apparent in terms of number but also apparent for us testers. We feel that Thumouse is much smoother and more responsive compared to our previous work. Overall, both *ThuMouse* and *IndexPen* performs well in resolving movement of finger-tip and shows good efficacy. LeapMotion ThuMouse shows significant decrease in mean-squared error compared to YOLO ThuMouse. IndexPen achieved an accuracy of 99.7% across 29 characters.

Table 3: System performance for two different systems. The second last row shows the statistics for x and y axis combined as a plane, and the row headed 'XYZ' is for the overall performance.

	LM ThM(mm)	WC ThM (mm)
RMSE X	$4.35e^{-4}$	1.27
RMSE Y	$9.75e^{-4}$	1.44
XY combined	$7.55e^{-4}$	16.4
X to Y ratio (no unit)	0.45	0.88

4.3 Third Iteration

4.3.1 mGesf GUI

To further modulate the experiment procedure, we developed an accompanying GUI in the second iteration.

The Micro-gesture Sensor Fusion Graphic User Interface (mGesf GUI) is designed to enable an all-in-one experiment pipeline including the connect to the sensors, parsing streams from them, recording experiment data, training evaluating the ML models in interpreting the data collected, and to carry out user studies with the established models.

The GUI includes two major components - the sensor and the experiment interface. The former is in charge of establishing serial interface with the various sensors (radar, camera, and so on) and relaying and visualizing the real-time data streams. The experiment interface is for researchers to conduct experiment sessions for the detection of micro-gestures. The current implementation includes experiment interface for IndexPen and ThuMouse. However, the software is not limited to experiment with this two already-defined gesture schemes. The IndexPen and ThuMouse problems can be generalized to common classification and tracking tasks respectively, and the GUI allows the user to gear the interface to their own customized experiments.

Controls At launch, mGesf GUI starts with the control tab, under which presents the control panels for various sensors. The current implementation includes the full control for the mmWave radar but limited for the LeapMotion and the UWB sensor as they are still under development. It is worth noting that each sensor's group presents a run-time area. This area would display real-time visualization for its relating sensor if that particular hardware is running. The run-times are to provide a holistic view to the user so that one does not need to switch to each sensor's individual tabs to confirm the running status of that sensor.

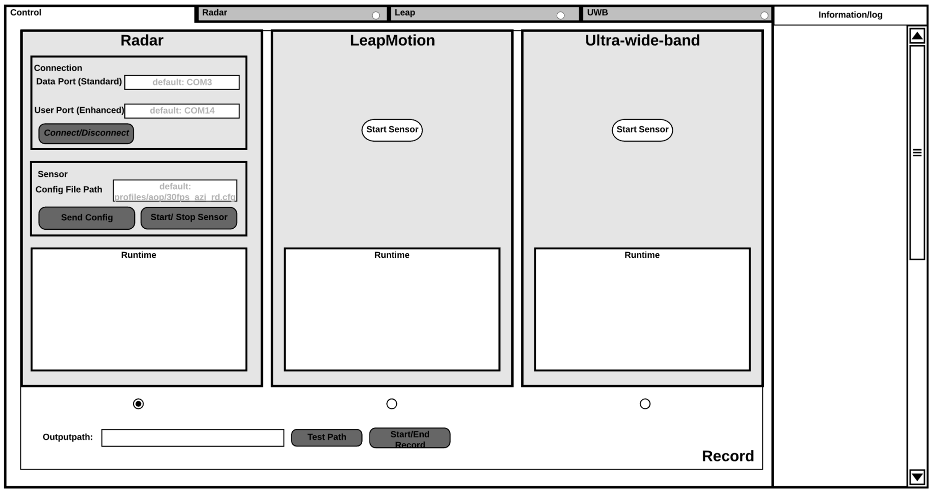


Figure 25: The control tab of mGesf GUI, each sensor has a panel for its controls. The common theme among the panels is the run-time window, which displays the data stream from the respective sensors if they are running.

Gestures The Gesture tab is a more complicated interface loaded with more substructures, but most of its idea coincides with that of the controls. The run-time windows from the control tab made their reappearance here because it is important for the experimenter to see the real-time profiling of sensors under study.

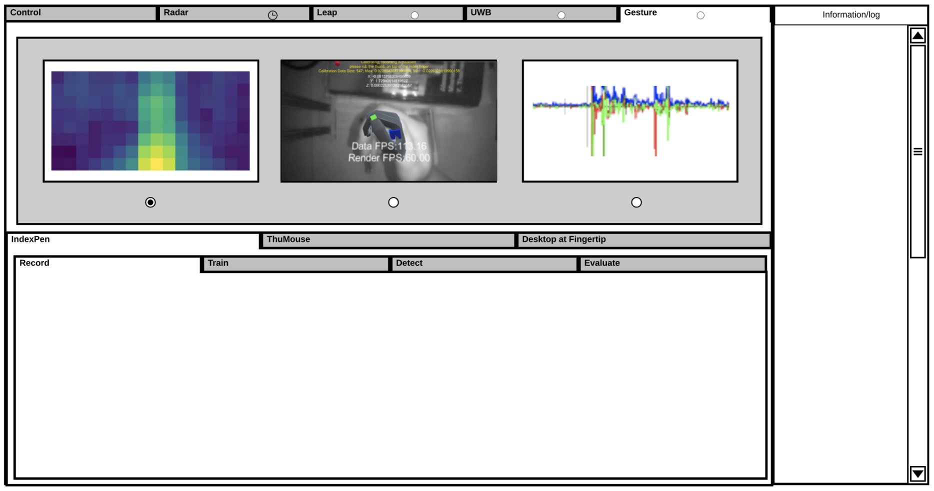


Figure 26: The gesture tab of mGesf GUI. The three figures above are the run-time plots for each sensor (for the ones that are running at the time). The lower-half presents specific options for carrying out gesture experiments.

The lower half of the Gesture tab provides a control interface to the experimenter and instructions to the subject. The current implementation includes three types of gesture paradigms - IndexPen, ThuMouse, and DFT, among which the DFT's is still under development and would be noted as part of the future work. The recording interface for Index follows the data collection methodologies.

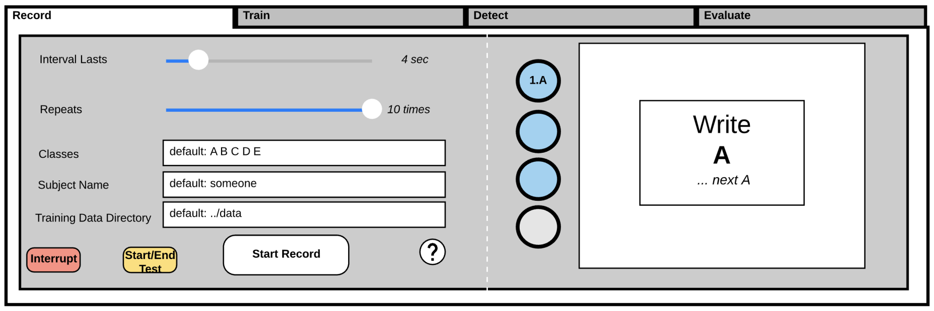
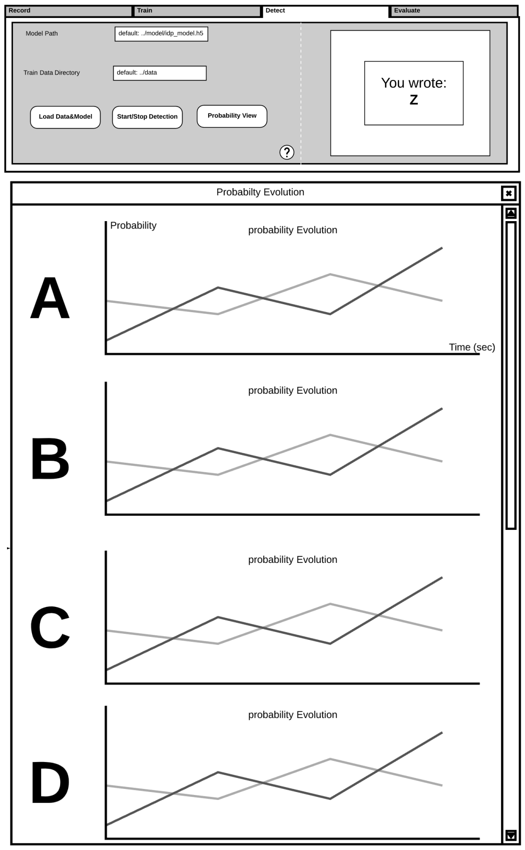


Figure 27: Experiment interface for recording the IndexPen data.

The interval slider tells how a single gesture performance lasts, where the repeats tells how many repetition the subject need to do for each gesture before moving on to the next. The user can define the categories of gesture of this experiment session in the classes text box. The right hand side is the instruction area for the subject. A metronome-like timer counts the second of each gesture performing interval, and its bubbles are lit as an interval progresses.


Model Path: default: ../model/idp_model.h5

Train Data Directory: default: ../data

You wrote:
z

Figure 28: (above) The detection tab for evaluating the IndexPen classification model (below) The detailed graph showing the temporal probability evolution for each class in the IndexPen alphabet

Once, the data is collected to a desired amount within the recording table. The experimenter can then proceed to the train tab (not shown in the figures) to establish a customizable machine learning model that resolve the gestures. The detection table is used for evaluating and testing the IndexPen model built through the same pipeline. Here, the user can load the trained model and perform real-time evaluation with it. This interface also provides another degree of insight into the model’s performance via the probability view. Here, the user can see how the predicted probability evolve as time progresses. This visualization of the temporal probability evolution is to give the experiment an understanding of the inner-working of their model, and facilitate them to improve on the model design that leads to more accurate inferences on the gestures.

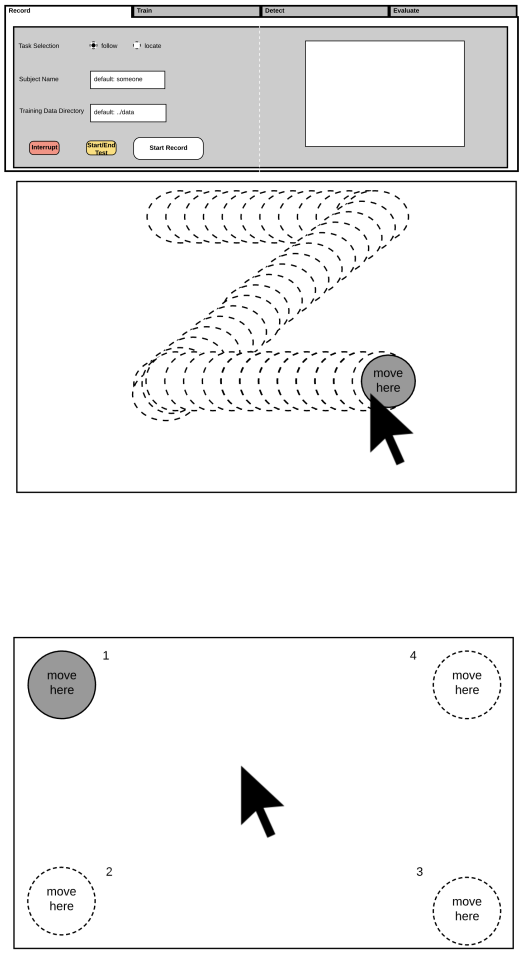


Figure 29: (one above) The recording tab for ThuMouse, that gives two types of interaction tasks. (two below) The two interaction tasks act as the recording session for ThuMouse, including that of ’following’ and ’locating’.

The ThuMouse recording tab is organized mostly the same way as that of the IndexPen's. Nevertheless, the recording session of ThuMouse is organized into interactive tasks instead of plain-gesture performance as seen for IndexPen. It is also worth noting that the interaction in the recording tasks are actuate through the ground-truth system. That is, the subject control the movement of the cursor through the ThuMouse gesture, and the gesture input is interpreted into cursor movement by the sensor that yields the ground-truth; in our second iteration, it is the LeapMouse app with the LeapMotion sensor.

The two interactive task are 'follow' and 'locate'. In the former, the user is asked to move the cursor following a predefined path displayed on the screen. The locate tasks shows spheres at the four corner of the instruction area, one at a time; the subject is asked to move the cursor to where the sphere is current located.

The interactive mood of collecting the ThuMouse data is significantly different from how the team did in the first iteration, where the subject is only asked to performed arbitrary movement on certain axis. Though due to the time constraints, the team has not been able to collect data in this interactive way, it is permissible to say the new way concurs more with how the people would use the system in real life and would lead to collect well-founded data that results in establishing more robusts statistic models for resolving the gestures.

5 Conclusion

In the development of input interfaces, every steps takes some new physical action and map it to the virtual world. From computer mouse, touch screen and to gestures, each new input paradigm not only excite significant interest of the public and the market. In the past decades, reflecting this trend, a series of technologies including alternative realities cite, eye-tracking cite, and gesture interface have emerged in our lives and build a high connection with our life. No matter we are at the home or cars, using these techniques gives us a better way to enjoy our life.

We presented a new concept called ThuMouse as a type of in-air gesture tracking. ThuMouse is performed by moving and clicking with the thumb on the plant of the index finger in order to move the cursor and realize the function of the click. The proposed ThuMouse scheme is different from most existing literature in the way that: Most literature focuses on large-scale gestures, usually involving the movement of multiple fingers and even hands. Compared with these gestures, micro-gesture is more preferable and practical in the mobile context (as shown in TipText 2019, Pyro 2017, Soli 2016). Besides, part of the literature such as TipText 2019 are allowed to put some sensors on the subjects, which will bring some limitations on use situations and make the users feel unnatural. Besides, the gestures proposed by most literature does not give haptic feedback.

ThuMouse, is a novel gesture paradigm by taking the mmWave sensor as the main hardware and two cameras as the ground truth. It is a complicated combination of the hardware and software, which include radar signal processing, Computer Vison, data analyse and Deep learning. Aimed to realize the function of moving and clicking, radar sensor picks up the reflected signals from the hands and recorded it as raw input matrix firstly. Then, these detected points are extracted and rasterized in the voxel space as the deep learning features. Besides, we recorded the radar frame along with two cameras, which put on the top and left. Then, we use the trained YOLO model to process these video frames and get the location information of the thumb. By processing these radar frames and video frames, we can realize the basic function of the ThuMouse.

Compared with the old, we change the ground truth from the two cameras into the LeapMotion. Compared with cameras, which often lose focus and have high dependence on environment, LeapMotion is a more stable and more powerful hardware. Besides, we take three user studies, which can make the ThuMouse more practical.

All in all, our contributions are: (1) introduce a new method for text-entry based in-air gesture recognition paradigm to detect the micro-gesture. (2) make an improvement of ThuMouse and take it into a application level. (3) Test different algorithms for ThuMouse in order to get the highest accuracy. (4) conduct three times of user study to improve the use experience.

Although our product shows significant promise in terms of IndexPen, it was wanting in terms of tracking performance of ThuMouse along the y and z axis. After testing the IndexPen, we have a 93% of recognition accuracy. After testing ThuMouse with our proof of concept website, it was brought to our attention that our y-axis and z-axis (elevation) was not working as well as our x-axis. We had to disable the two axes to make our product work. Figure below shows a diagram of our prototype. There are two cameras one to detect x, y axis and another one from above to check the y, z axis. We have concluded that our current design is not operational for everyday use and we have decided to change our product design as well as our training models to make further developments on this project.

Instead of working on our project in the usual waterfall fashion, we decided to work on it using agile methods. So, in total, we formed 6 iterations to work on across the year. As of now, we have finished two interactions. Each iteration consists of modifying and adapting to whatever the users need. There will be user studies being conducted whenever the development is operational. Figure above indicates how the project has established across the two interactions before finishing A-term.

From March to July, the first iteration was ran. During the Preliminary Research phase, we mainly searched for design tools, wrote out schematics, create prototype requirements, and assembled the prototype. From July to October, the second iteration was ran. Our main objective during this iteration was to develop our project to hit the CHI conference schedules. First, we did a literature review and collected data to create a ThuMouse model and the IndexPen application. We also did a proof of concept demonstration by creating a website to allow the users experience our model of ThuMouse. Although we did not catch the CHI submission deadline, we were able to submit the results to ICCE conference. We plan to use the first ten days of October to retrospection and documentation of the work that we have done.

From October to December, we presented ThuMouse at IEEE ICCE, and studied new radar module from the Texas Instrument and used LeapMotion instead of YOLO. We worked on the LeapMouse applications and LeapMotion Radar Fusion design options. We presented our ThuMouse prototype at the ICCE conference.

From January to May, we focused our energy on UIST conference submission. At first, we were going to make a new version of ThuMouse to submit to the conference. Due to timing restraints, we had to shift our gear and use "Desktop at Fingertip" which incorporated a new version of ThuMouse along with the IndexPen work we did. Afterwards, from After that, we plan to brainstorm and develop new products using the results, feedback, and experience that we have gotten from the iteration 2.

6 Future Work

For *IndexPen*, given the high classification of IndexPen, we assume that similar gestures can be fitted to alphabets in addition to, or other than that of the English. An immediate improvement to the proposed IndexPen is to add the Arabic numbers to the gesture set. On the other hand, it is worth investigating to what size an *IndexPen* gesture set would be so the accuracy would reach its ceiling. That is to study how the classification error varying given more character categories, and possibly extend to resolve characters from ideographic languages where the number of written characters are more pronounced.

We have shown that *ThuMouse* enables the user to control the movement of the cursor. To extend this gesture to have the same capability as a computer mouse or track-pad, additional gestures need to be considered to cover controls such as click, scrolling, and zooming. As the core idea of *ThuMouse* centers upon natural mapping, we recommend future work to investigate how an in-air micro-gesture system can achieve the same level of control as the everyday desktop input devices.

In near future, we decided to finish the user-studies that we planned on conducting. The results on the user-studies could go into the CHI conference submission next semester. We would like to explore our options with Ultra-wide band. We would also like to finish developing the GUI that we are working on for the sensor fusion approach as that GUI would make collecting data much more efficient.

References

[1] Fadel Adib and Dina Katabi. 2013. *See through walls with WiFi!* Vol. 43. ACM.

[2] Amin Arbabian, Steven Callender, Shinwon Kang, Mustafa Rangwala, and Ali M Niknejad. 2013. A 94 GHz mm-wave-to-baseband pulsed-radar transceiver with applications in imaging and gesture recognition. *IEEE Journal of Solid-State Circuits* 48, 4 (2013), 1055–1071.

[3] Parvin Asadzadeh, Lars Kulik, and Egemen Tanin. 2012. Gesture recog-

nition using RFID technology. *Personal and Ubiquitous Computing* 16, 3 (2012), 225–234.

[4] Edwin Chan, Teddy Seyed, Wolfgang Stuerzlinger, Xing-Dong Yang, and Frank Maurer. 2016. User Elicitation on Single-hand Microgestures. 3403–3414. DOI:http://dx.doi.org/10.1145/2858036.2858589

[5] Liwei Chan, Rong-Hao Liang, Ming-Chang Tsai, Kai-Yin Cheng, Chao-Huai Su, Mike Y Chen, Wen-Huang Cheng, and Bing-Yu Chen. 2013. FingerPad: private and subtle interaction using fingertips. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 255–260.

[6] Nasser H Dardas and Nicolas D Georganas. 2011. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transactions on Instrumentation and measurement* 60, 11 (2011), 3592–3607.

[7] Artem Dementyev and Joseph A. Paradiso. 2014. WristFlex: Low-Power Gesture Input with Wrist-Worn Pressure Sensors. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. Association for Computing Machinery, New York, NY, USA, 161–166. DOI:http://dx.doi.org/10.1145/2642918.2647396

[8] Zehua Dong, Fangmin Li, Julang Ying, and Kaveh Pahlavan. 2018. Indoor Motion Detection Using Wi-Fi Channel State Information in Flat Floor Environments Versus in Staircase Environments. *Sensors* 18, 7 (2018), 2177.

[9] Anonymous for review and submitted anonymized version as supplemental file for reference. 2020. ThuMouse: A Micro-gesture Cursor Input through mmWave Radar-based Interaction. In *2020 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 1–9.

[10] Yishuang Geng, Jin Chen, Ruijun Fu, Guanqun Bao, and Kaveh Pahlavan. 2015. Enlighten wearable physiological monitoring systems: On-body rf characteristics based human motion classification using a support vector machine. *IEEE transactions on mobile computing* 15, 3 (2015), 656–671.

[11] Chen-Yu Hsu, Yuchen Liu, Zachary Kabelac, Rumen Hristov, Dina Katabi, and Christine Liu. 2017. Extracting gait velocity and stride length from surrounding radio signals. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2116–2126.

[12] Donny Huang, Xiaoyi Zhang, T Scott Saponas, James Fogarty, and Shyamnath Gollakota. 2015. Leveraging dual-observable input for fine-grained thumb interaction using forearm EMG. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 523–528.

[13] Cesar Iovescu and Sandeep Rao. 2017. The fundamentals of millimeter wave sensors. *Texas Instruments, SPYY005* (2017).

[14] Eva Kollorz, Jochen Penne, Joachim Hornegger, and Alexander Barke. 2008. Gesture recognition with a time-of-flight camera. *International Journal of Intelligent Systems Technologies and Applications* 5, 3 (2008), 334.

[15] Jaime Lien, Nicholas Gillian, M Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 142.

[16] Yongsen Ma, Gang Zhou, Shuangquan Wang, Hongyang Zhao, and Woosub Jung. 2018. Signfi: Sign language recognition using wifi. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–21.

[17] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2014. Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 1565–1569.

[18] Daniel Maturana and Sebastian Scherer. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 922–928.

[19] Kaveh Pahlavan, Julang Ying, Ziheng Li, Erin Solovey, John Loftus, and Zehua Dong. 2020. RF Cloud for Cyberspace Intelligence. *IEEE Access* (2020).

[20] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. 2013. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2027–2034.

[21] Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* (2017).

[22] Ivan Poupyrev. 2017. Radar-based gesture sensing and data transmission. (Nov. 2017). US Patent 9,811,164.

[23] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 27–38.

[24] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2015. Gesture recognition using wireless signals. *GetMobile: Mobile Computing and Communications* 18, 4 (2015), 15–18.

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 652–660.

[26] Kun Qian, Chenshu Wu, Zimu Zhou, Yue Zheng, Zheng Yang, and Yunhao Liu. 2017. Inferring motion direction using commodity wi-fi for interactive exergames. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 1961–1972.

[27] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.

[28] Herbert Richter. 2000. Palm pilot holder. (Sept. 26 2000). US Patent App. 29/114,214.

[29] Krishnakant Vijay Saboo and Sandeep Rao. 2017. Gesture recognition using frequency modulated continuous wave (FMCW) radar with low angle resolution. (Nov. 2017). US Patent 9,817,109.

[30] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 1998. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery* 2, 2 (1998), 169–194.

[31] T Scott Saponas, Desney S Tan, Dan Morris, and Ravin Balakrishnan. 2008. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 515–524.

[32] T Scott Saponas, Desney S Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A Landay. 2009. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. 167–176.

[33] Karly A Smith, Clément Csech, David Murdoch, and George Shaker. 2018. Gesture recognition using mm-wave sensor for human-car interface. *IEEE Sensors Letters* 2, 2 (2018), 1–4.

[34] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. 2016. Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, 851–860.

[35] Yao Wang and Qin-Fan Zhu. 1998. Error control and concealment for video communication: A review. *Proc. IEEE* 86, 5 (1998), 974–997.

[36] Zheer Xu, Pui Chung Wong, Jun Gong, Te-Yen Wu, Aditya Shekhar Nittala, Xiaojun Bi, Jürgen Steimle, Hongbo Fu, Kening Zhu, and Xing-Dong Yang. 2019. TipText: Eyes-Free Text Entry on a Fingertip Keyboard. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 883–899.

[37] Yunze Zeng, Parth H Pathak, Zhicheng Yang, and Prasant Mohapatra. 2016. Human tracking and activity monitoring using 60 GHz mmWave. In *2016*

15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 1–2.

[38] Zhengyou Zhang. 2012. Microsoft kinect sensor and its effect. *IEEE multimedia* 19, 2 (2012), 4–10.

[39] Chen Zhao, Ke-Yu Chen, Md Tanvir Islam Aumi, Shwetak Patel, and Matthew S Reynolds. 2014. SideSwipe: detecting in-air gestures around mobile devices using actual GSM signal. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 527–534.

[40] Yongpan Zou, Weifeng Liu, Kaishun Wu, and Lionel M. Ni. 2017. Wi-Fi Radar: Recognizing Human Behavior with Commodity Wi-Fi. *IEEE Communications Magazine* 55, 10 (Oct 2017), 105–111. DOI:http://dx.doi.org/10.1109/MCOM.2017.1700170

[41] Yongpan Zou, Jiang Xiao, Jinsong Han, Kaishun Wu, Yun Li, and Lionel M Ni. 2016. Grfid: A device-free rfid-based gesture recognition system. *IEEE Transactions on Mobile Computing* 16, 2 (2016), 381–393.