ThuMouse: A Micro-gesture Cursor Input through mmWave Radar-based Interaction

Ziheng Li, Zhenyuan Lei, An Yan, Erin Solovey, and Kaveh Pahlavan
Worcester Polytechnic Institute, Massachusetts, USA
zli12, zlei3, ayan, esolovey, kaveh@wpi.edu

***Abstract*—In this paper, we propose 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 enables users to move the cursor using frequency-modulated continuous-wave (FMCW) radar. While previous work with FMCW radar in human-computer-interfaces (HCI) has focused on classifying a set of predefined hand gestures, ThuMouse regressively tracks the position of a finger, which allows for finer-grained interaction. This paper presents 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 report on a proof-of-concept demonstration shows how our system can function as a mouse, and identify areas for future work. This work builds a foundation for designing finer micro gesture-based interactions, allowing the finger to emulate external input devices such as a joystick and touch-pad.**

***Index Terms*—Millimeter Wave FMCW Radar; Micro-gesture sensing; Interaction; Deep Learning**

I. Introduction

With interaction continuously moving 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 investigated. Many proposed systems leverage channel information from existing infrastructure such as commodity WiFi [4] and Radio-frequency identification (RFID) [7] to detect human activities and gestures. Under the umbrella of 5G standards that are taking shape recently, wireless technologies are sought to possess increased speed, reduced latency, and to become more energy-efficient and cost-effective. Among the various new standards, mmWave radar's Frequency-Modulated Continuous-Wave (FMCW) variant captures the spatial and temporal information of objects by transmitting a continuous wave modulated in a frequency range. The dynamic profile given by the device, thank to the high signal frequency (usually greater than 60GHz) as well-established processing chain [13] [8] is remarkably precise and capable of achieving sub-millimeter accuracy. In addition, the sensor's feature is relatively light-weight compared to other gesture input methods such as computer-vision based approaches.

Overall, mmWave sensors presents numerous advantages as a gesture sensor, including their characteristic environment-independence, compact size, and low cost both computationally and monetarily, making them candidates upon which novel gesture schemes can be established. In this regard, prior work includes earlier work by Arbabian et al. [1] on high-frequency pulse-band radar, and more recently, Texas Instruments mmWave sensor [13] and Google ATAP's Project Soli [8] have contributed to the design of solid-state mmWave FMCW sensors applications around them.

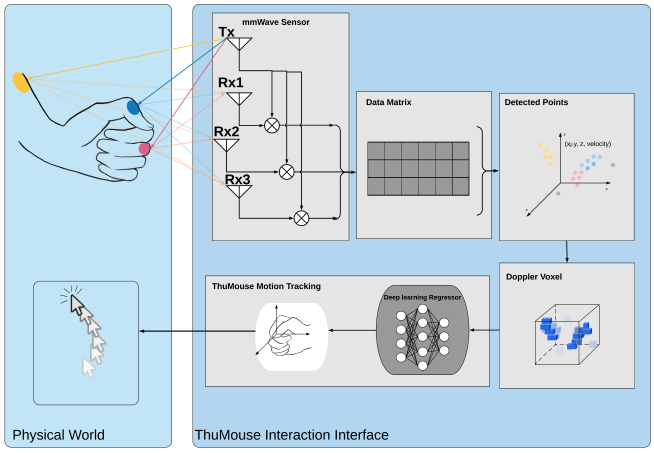

Fig. 1. Processing pipeline of ThuMouse. During a dynamic gesture, the radar device picks up the reflected signal from parts of the hand, recorded as raw input matrices. Detected points are extracted and rasterized into voxel space as deep learning features. The regressor interprets the x-y-z coordinates of the thumb-tip and produces cursor-like interaction.

In this paper, we describe the design and evaluation of a novel gesture paradigm enabled by mmWave sensing; the system overview is featured in Fig 1. This work demonstrates 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 for movement along a single axis, and mouse movement if the displacement is 2-dimensional. This gesture sensing approach has implications for many applications such as automotive applications [11], VR headsets, etc.

Our main contributions are: (1) By leveraging the sensing ability powered by the signal processing chain from [13], we detect the spatial position of objects as well as their velocity, making it possible to track the finer gesture. (2) We describe the design of real-time tracking with an end-to-end gesture pipeline using the radar point cloud combined with several

data augmentation methods to enrich the feature and build more robust models. (3) We illustrate the implementation and evaluation of 3D Convolutional Long-short-term-memory (LSTM) deep learning model the processed the radar point cloud data (PCD) to achieve the motion tracking and gesture classification. (4) We propose a dual-input training system utilizing computer vision (CV) to automate labeling the tracking information.

II. Background in mmWave-gesture

As a sensing technology, mmWave FMCW radar possesses high resolution in capturing motions, and for the purpose of this paper 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.

Early work [12] [1] laid a hardware and signal processing foundation in utilizing mmWave in gesture detection. More recently, Soli [8] [2] [5] [6] 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 [8] . On the other hand, prior work [11] implemented car infotainment interaction with the sensor [8] 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 [8] [11] and Support Vector Machine (SVM) [17] have been used. In recent years, deep learning has shown great potential in performing through its capability to learn intermediate representations of raw data and has been shown to routinely beat traditional machine learning methods [2].

Wang et al. investigated the use of deep learning models in resolving the dynamic profile of various gestures given by mmWave sensors [2]. Namely, the features used are the range-velocity map on a per-frame bases. Noting the fact this type of feature is essential a 2D image-like array, Wang et al. [2] applied Convolutional Recurrent Neural Network (CRNN), typically used for video classifications and arrived at accuracy of 87% on 11 gestures across multiple subjects.

While these approaches have shown promise, we summarize the limitations as follows: (1) Compared 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. Because of this, distinguishing similar gestures suffers because the moving parts (i.e. specific fingers) reside in close proximity to each other. (2) Current approaches feed to the machine learning model 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. This data can vary across different platforms in their size and resolution, which calls for domain specific predicting models. In contrast, image data from cameras possess 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 are relatively unique compared with other sensing technologies, which makes it difficult to adapt existing pre-processing and predicting methods.

III. System Architecture

In this section, we review principles of the FMCW radar that allow the recognition of physical information about objects. We discuss the detection accuracy of mmWave sensors and the parameters affecting performance. We also discuss the hardware used for this study and the radar signal we designed specifically for micro-gesture sensing.

A. Point Detection

1) mmWave Sensing Principles: 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 transmitted signal is then mixed with the reflected signal picked up by the receiver antennas (Rx);

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

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

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

From the above equations, we can get the intermediate frequency (IF) signal 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 correspond to the distance between the radar and the reflecting object. This FFT is known as 1-dimension FFT (1DFFT).

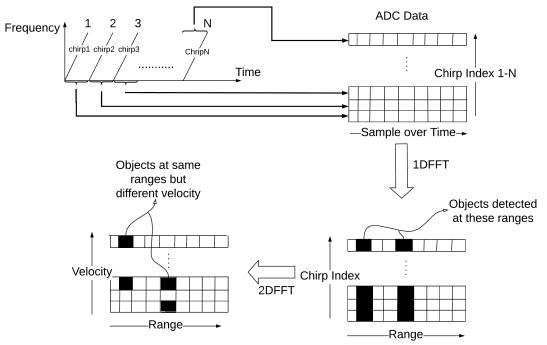

Fig. 2. The analog to digital converter (ADC) data is extracted from the intermediate frequency (IF) signals and the points become rows of the 2 dimensional matrix. Using the range-FFT on each row, the range information is calculated and a Doppler-FFT on the column provides the velocity information [13]

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 the 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 reside within R_{res}, but during dynamic gestures, fingers or parts of the hand typically move at different speeds relative to each other. This creates a dynamic profile which provides a fertile feature space for micro-gesture study.

With this, 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 in the IF signal across different Rx's. Moreover, with antennas arranged both horizontally and vertically, the sensor, with a 3DFFT over the bins obtained across different Rx's, is able to resolve the angle of arrival (AoA) on both azimuth and inclination plane.

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(θ) 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 θ increases and approaches 90 degrees, the angle estimation accuracy degrades as the sin(θ) value does not change much.

With the radial range, angles including azimuth and inclination, and velocity, the points are essentially velocity heatmap 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) The radial distance (relative to the radar) between to reflecting surface is greater than R_{res}. (2) if (1) is not satisfied, then the 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 (from 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.

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

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

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 [8]. (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-parameters affect the precision of tracking. Without an in-depth analysis of tuning signal configuration and with consideration on hardware limits, we apply

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

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

3) Hardware: 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. Benefiting from the on-board digital processing unit and hardware accelerator aimed at quickly computing FFT and resolving 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.

B. Gesture Detection

The software gesture pipeline starts with detected points, which represent the objects in front of 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 analyzes the pre-processed feature and how the gesture schemes are actuated.

1) Point Matrix: The preprocessing step transforms the detected points into suitable machine learning features for which a neural network can study. The points are first clustered and filtered, so that 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 consists 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 are satisfied. We can refer to the matrix as the point array as each row denotes a detected point.

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

2) 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 outliers; 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 [15] 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 dependant 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-processing), the spatial coordinates in $P_{filtered}$ need to be within the range of 0-1. Therefore, the xyz value are the min-max normalized in the bounding volume with:

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{11}$$

The above operation extends to the y and z axis, where the minimal and maximum values are given by the bounding volume:

$$\begin{aligned} x_{min} = y_{min} = z_{min} = -R_{bound} \\ x_{max} = y_{max} = z_{max} = R_{bound} \end{aligned} \tag{12}$$

3) Voxelization: As the last step in pre-processing, we rasterize $P_{tfiltered}$ 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 success in extracting and representing point cloud data if put in voxel form [16]. 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 units 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 fourth column of $P_{filtered}$ is the velocity (doppler) of each observed points. Unlike typical point-cloud voxelization methods [9], 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.

4) 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. In this study, we utilize data augmentation methods [9] [10] to enlarge our data sets. 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 non-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 spatial coordinates of the detected points. It is for simulating an added and small Gaussian noise to the data. In other words, each point is translated by a small displacement; the amount of displacement is defined by the noise distribution which has a mean of 0 meters and a standard deviation of 0.02 meters. The numbers are derived from empirical observation for our study case for the noise can not be too large to modify the original data-set in a substantial way nor can it be so small that its augmentation value is barely noticeable.

Scale linearly alters the coordinates of the points along 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 individuals with different shaped hands than those of the participants.

Rotation is used 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 follows the same distribution as in the translation and scale.

Algorithm 1: Data augmentation algorithm applied to Detected Points

Result: an augmented sample P_{aug} from $P_{filtered}$
if $translate$ **then**
  for $pin P_{filtered}$ **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

5) 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' [2] [11], 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. The surface defined by the index finger is emulating the mouse pad, which is natural, soft and unobtrusive. This gesture can guarantee that the index finger surface has 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.

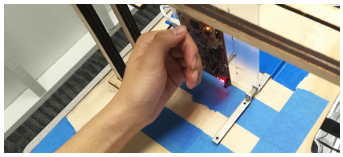

Fig. 3. ThuMouse gesture is performed with the thumb of the acting hand pointing at the center of the antennas to achieve best angle resolution

C. Experiment 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. 4.

In contrast to work using the Range-Doppler profile as input [2], 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.

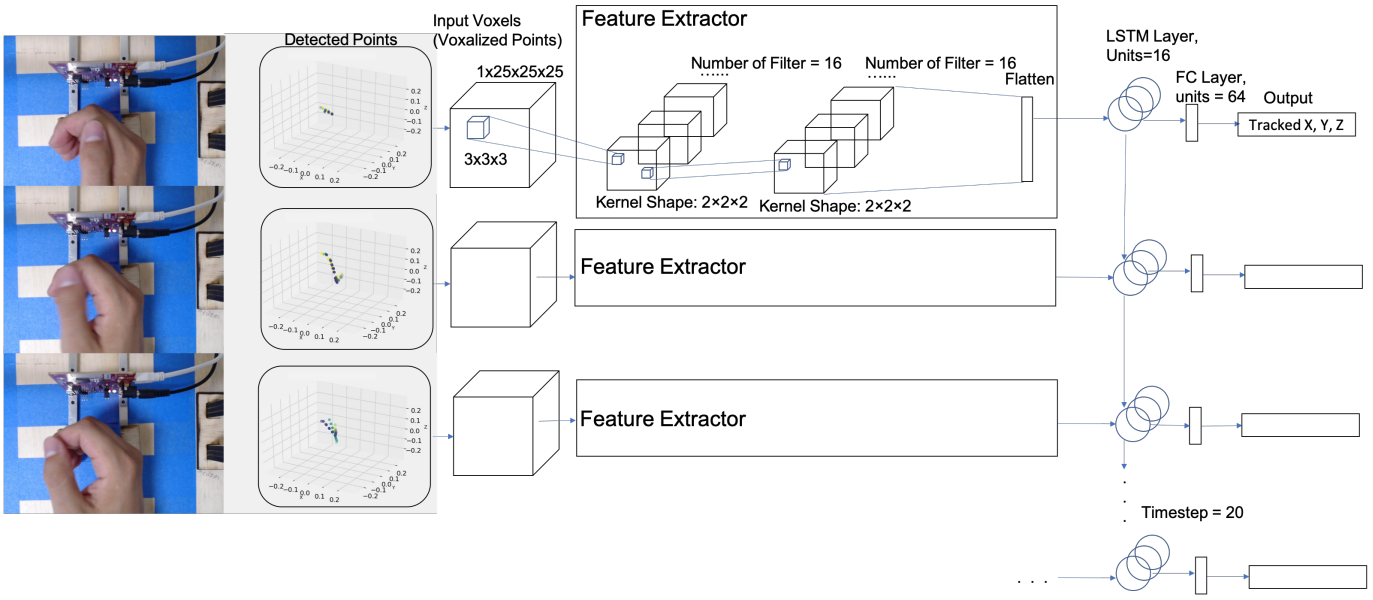

Fig. 4. ThuMouse CRNN Architecture: the input layer reads the voxalized detected points from the mmWave sensor; 3D convolution is then performed on the 3D volumes to produce the feature map which is feed into LSTM layers. LSTM cells propagate information into fully connected layers and outputs the x, y and z as tracked position of the finger

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.

D. 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. The ThuMouse needs to resolve how much the cursor moved at each radar frame, given that the radar is capturing detected points at 20 FPS, manually labeling the data is obviously out of the question.

1) Computer Vision as Training Base: Inspired by Huang et al. [3], 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.

2) YOLO Object Detection: To get the location information of thump tip in each camera frames, we use the YOLO [14]. 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.

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.

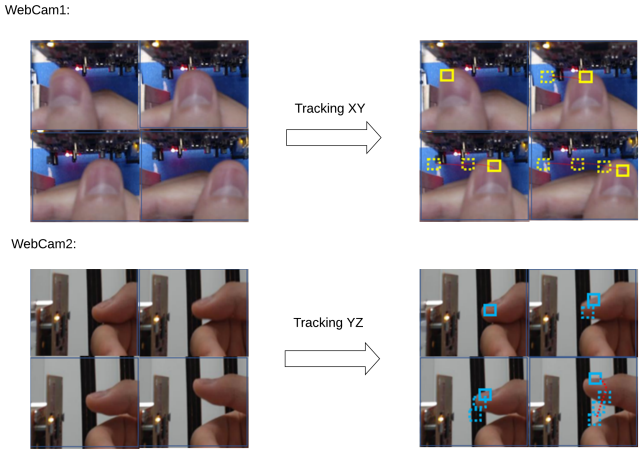

Fig. 5. YOLO Sample: YOLO algorithm is used to predict thumb tip's location with the top and side cameras. The tracked box produced forms the ground-truth for evaluating the radar tracking performance. Further reading about YOLO network's detail in [14]

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.

IV. Evaluation and Results

Here we discuss how the ThuMouse system performs. Our quantitative evaluation analyzes the tracking effectiveness of ThuMouse by showing the system's efficacy on the validation set.

A. Experiment Environment

The experiment environment is set up in a way so that the radar is always at a same relative position to the cameras. While the system collects radar frames, videos are streamed from two webcams, one above the hand to detect the x and y (cam1) and the other one 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.

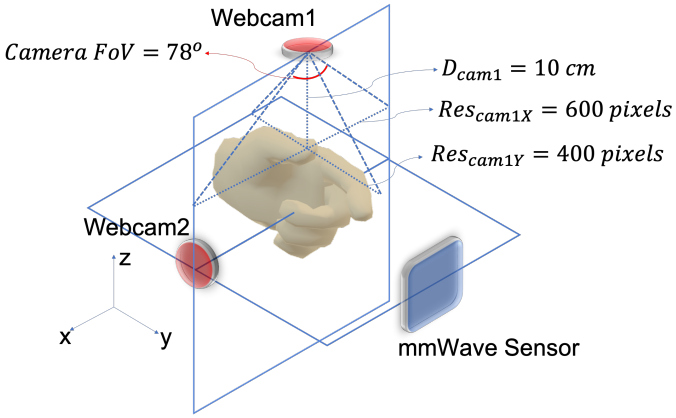

Fig. 6. Left: Experiment setup, one camera is mounted on the top shaft; second camera is located at the side. Middle: ThuMouse Gesture, the thumb rubs on the planar surface of the index finger

At the same time, to evaluate the system's performance in matrix units, all trials are carried out with the hand at the same relative position to the top camera. Doing so ensures the displacement per pixel is consistent across sessions. The values include the distance from above camera and hand, resolution of the cam1 and cam2, and the angle of views.

$$Res_{cam1Y} = 400pixels \qquad Res_{cam1X} = 600pixels$$
$$D_{cam1} = 10cm \qquad AoV = 78°$$

Using these values, we get:

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

B. Quantitative Results

Here we present the validation result to show the tracking capability of the proposed system. As of the time of this work, we are not aware of any previous work on tracking the position of finger through mmWave sensors. Thus there is no sound base for comparison.

TABLE I
System performance along each axis. 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.

	Mean Squared Error	Standard Deviation
X	9.23 px^2(1.27mm^2)	3.03 px^2(1.12mm^2)
Y	23.6 px^2(1.44mm^2)	4.86 px^2(1.20mm^2)
Z	64.4 px^2	8.02 px^2
XY	16.4 px^2(1.35mm^2)	4.05 px^2(1.16mm^2)
XYZ	32.4 px^2	5.69 px^2

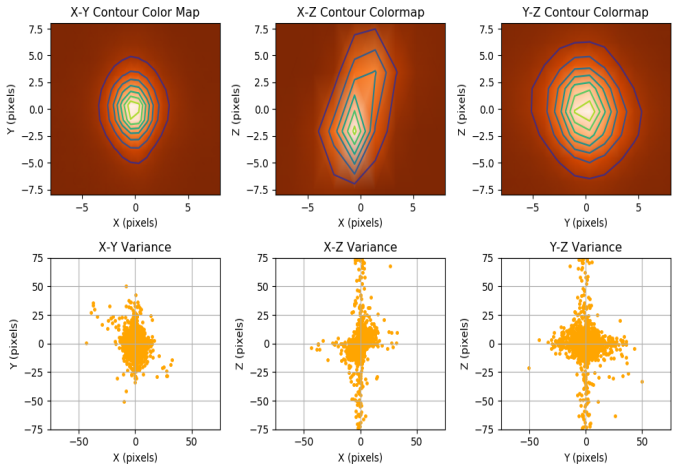

Fig. 7. Left:X-Y Contour Color Map;Middle:X-Z Contour Color Map; Right:Y-Z Contour Map. From the figure, the XY contour map is most stable and gather almost in one area

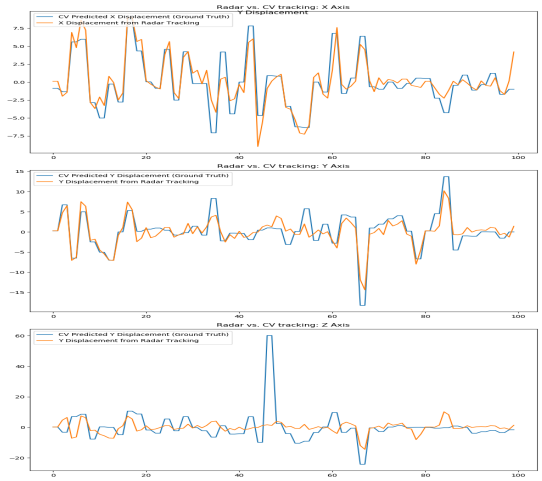

Fig. 9. Radar tracking (Orange Line) vs. CV tracking (Blue Line): the charts shows 120 consecutive tracking along x, y and z axis.

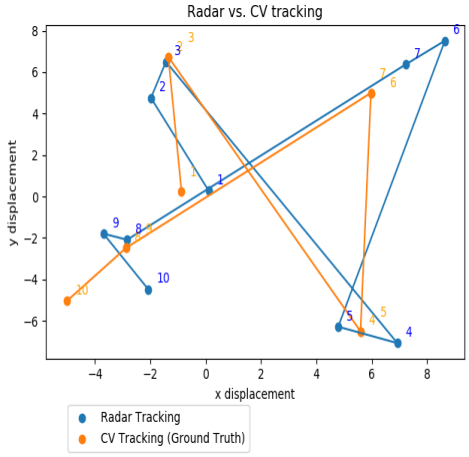

Fig. 8. Radar tracking vs. CV tracking : the chart shows 10 consecutive tracking result on the x-y plane. The number in the graph are the index in time.

Overall, ThuMouse performs well in resolving movement of the thumb-tip on the rubbing on the surface of the index finger. For above statistics and Figure 5-7, it is evident that the general trend of radar tracking predictions correspond to that of the cameras. 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 milimeter). 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.

V. Discussion and Future Work

We introduced ThuMouse, a novel micro-gesture that tracks the position of a finger to realize intuitive cursor-like input based on the mmWave technology. By using the end-to-end training model composed by Convolutional Neural Network and Recurrent Neural Network, we can achieve high accuracy, real time tracking and realize the gesture control. On the other hand, we acknowledge the following limitation of the system: the tracking performance along the y and z axis is not as desirable comparing to that along the x-axis. The relatively narrower inclination FoV may contribute to the poor sensitivity in resolving vertical movement. Besides, the location of the hand affects the distribution of points (features).

As for future work, we wish to highlight four potential aspects. (1) The YOLO object detection can be used as ground-truth in application-specific scenarios as we did. More generalized gesture product such as LeapMotion may be able to to further facilitate the training of radar gesture models. (2) mmWave sensors have the ability to penetrate certain materials, so it is can be possible to track the finger movement in non-line-of-sight (NLOS) settings where gesture performing area is obstructed, such as writing with the index finger in the center of the palm. (3) Another interesting point is that the radar can detect objects with the same range but different velocity, which entails that it has the potential to track multiple movements at the same time. Thus, in the future we could track more fingers or more fine-grained movements. (4) The software pipeline presented in this work runs on a system with Intel i9-8950HK and GTX1080 graphics card and achieved a relatively smooth interaction. The real-life deployment of such system would involve further study on optimizing the model to give higher frame rate on low-end systems.

Conclusion

In this paper, we propose the ThuMouse for real-time tracking, based on mmWave technology. Our technique depends on the end-to-end trained, which combined the YOLO, Convolutional Neural Network and Recurrent Neural Network. By using these methods, we can realize that the real time tracking of the fine grained gesture. Future work could decrease the

latency of the ThuMouse and explore more gesture tracking potential of the mmWave technology.

ACKNOWLEDGMENT

We would like to express our gratitude to Meijie Wang, colleague from the CS department, who lead her expertise with constructing our experiment environment. Thanks to Hanfei Sun with his contribute to the data augmentation algorithm. We would also like to extend our thanks to Alex Galvan, and our coworkers at the lab who participated in our experiments, allowing us to collect valuable data within time constraint. Lastly, we are grateful to Aung Khant Min and Daliang Shi for their editorial effort on our paper.

REFERENCES

[1] Arbabian, A., Callender, S., Kang, S., Rangwala, M., and Niknejad,A.M. "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.

[2] Wang, S., Song, J., Lien, J., Poupyrev, I., Hilliges, O. "Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum." Proceedings of the 29th Annual Symposium on User Interface Software and Technology. ACM, 2016.

[3] Huang, D., Zhang, X., Saponas, T. S., Fogarty, J., and Gollakota, S. "Leveraging dual-observable input for fine-grained thumb interaction using forearm EMG." Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology. ACM, 2015.

[4] Pu, Q., Gupta, S., Gollakota, S., and Patel, S. "Gesture recognition using wireless signals." GetMobile: Mobile Computing and Communications 18.4 (2015): 15-18.

[5] Poupyrev, Ivan. "Radar-based gesture sensing and data transmission." U.S. Patent No. 9,811,164. 7 Nov. 2017.

[6] Saboo, Krishnakant Vijay, and Sandeep Rao. "Gesture recognition using frequency modulated continuous wave (FMCW) radar with low angle resolution." U.S. Patent No. 9,817,109. 14 Nov. 2017.

[7] Zou, Yongpan, et al. "Grfid: A device-free rfid-based gesture recognition system." IEEE Transactions on Mobile Computing 16.2 (2016): 381-393.

[8] Lien, Jaime, et al. "Soli: Ubiquitous gesture sensing with millimeter wave radar." ACM Transactions on Graphics (TOG) 35.4 (2016): 142.

[9] Papon, J., Abramov, A., Schoeler, M., and Worgotter, F. "Voxel cloud connectivity segmentation-supervoxels for point clouds." Proceedings of the IEEE conference on computer vision and pattern recognition. 2013.

[10] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[11] Smith, K. A., Csech, C., Murdoch, D., and Shaker, G. "Gesture recognition using mm-wave sensor for human-car interface." IEEE Sensors Letters 2.2 (2018): 1-4.

[12] Zeng, Y., Pathak, P. H., Yang, Z., and Mohapatra, P. "Human tracking and activity monitoring using 60 GHz mmWave." 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). IEEE, 2016.

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

[14] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[15] Sander, J., Ester, M., Kriegel, H. P., and Xu, X. "Density-based clustering in spatial databases: The algorithm gdbscan and its applications." Data mining and knowledge discovery 2.2 (1998): 169-194.

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

[17] Dardas, Nasser H., and Nicolas D. Georganas. "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.