Pantomime: Mid-Air Gesture Recognition with Sparse Millimeter-Wave Radar Point Clouds

SAMEERA PALIPANA and DARIUSH SALAMI, Aalto University, Finland
LUIS A. LEIVA, University of Luxembourg, Luxembourg
STEPHAN SIGG, Aalto University, Finland

We introduce Pantomime, a novel mid-air gesture recognition system exploiting spatio-temporal properties of millimeter-wave radio frequency (RF) signals. Pantomime is positioned in a unique region of the RF landscape: mid-resolution mid-range high-frequency sensing, which makes it ideal for motion gesture interaction. We configure a commercial frequency-modulated continuous-wave radar device to promote spatial information over the temporal resolution by means of sparse 3D point clouds and contribute a deep learning architecture that directly consumes the point cloud, enabling real-time performance with low computational demands. Pantomime achieves 95% accuracy and 99% AUC in a challenging set of 21 gestures articulated by 41 participants in two indoor environments, outperforming four state-of-the-art 3D point cloud recognizers. We further analyze the effect of the environment in 5 different indoor environments, the effect of articulation speed, angle, and the distance of the person up to 5m. We have publicly made available the collected mmWave gesture dataset consisting of nearly 22,000 gesture instances along with our radar sensor configuration, trained models, and source code for reproducibility. We conclude that pantomime is resilient to various input conditions and that it may enable novel applications in industrial, vehicular, and smart home scenarios.

CCS Concepts: • **Human-centered computing** → **Gestural input**; • **Hardware** → *Sensor applications and deployments*; Signal processing systems.

Additional Key Words and Phrases: gesture recognition; radar sensing; deep learning

ACM Reference Format:
Sameera Palipana, Dariush Salami, Luis A. Leiva, and Stephan Sigg. 2021. Pantomime: Mid-Air Gesture Recognition with Sparse Millimeter-Wave Radar Point Clouds. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 1, Article 27 (March 2021), 27 pages. https://doi.org/10.1145/3448110

1 INTRODUCTION

Gesture interaction is an active research topic in Human-Computer Interaction (HCI), with a history dating back to the 1960s and Sutherland's Sketchpad [54] or The Ultimate Display [55] projects. Gestures can be categorized into two broad groups: (1) *mid-air* or *motion* gestures, used e.g. in consumer electronics such as gaming consoles, and (2) *stroke* gestures, used e.g. in touch-capable devices such as smartphones. In this paper, we focus on mid-air gesture recognition. Motion gestures do not require a handheld or a dedicated input device (e.g. a stylus) and allow for natural interaction with less spatial constraints than stroke gestures. Indeed, earlier research in tabletop interaction design [18] recognized the constraints of 2D interaction spaces. Mid-air interaction exploits whole body movements through gestures and postures to interact with digital content [20].

Authors' addresses: Sameera Palipana, sameera.palipana@aalto.fi; Dariush Salami, dariush.salami@aalto.fi, Aalto University, Finland; Luis A. Leiva, name.surname@uni.lu, University of Luxembourg, Luxembourg; Stephan Sigg, stephan.sigg@aalto.fi, Aalto University, Finland.

2474-9567/2021/3-ART27 $15.00
https://doi.org/10.1145/3448110

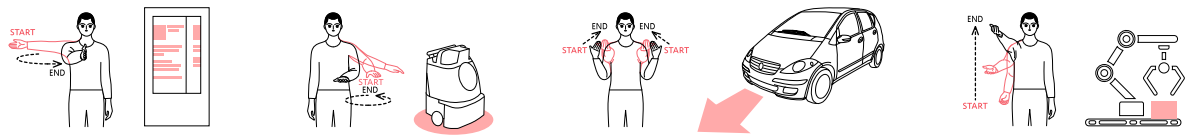


Fig. 1. Some examples of potential application scenarios of our technology. From left to right: swiping screens on a wall display, operating a cleaning robot, remote-controlling a vehicle, and human-robot collaboration in industry.

Recently, electromagnetic radiation has emerged as a popular medium for mid-air gesture recognition, as it enables device-free sensing and facilitates a variety of interaction modalities such as WiFi, radar, infra-red cameras, or RGB-depth sensors. Among these, radar sensing is particularly interesting, as it offers 3D spatial information, is robust to weather conditions, does not require lighting, and can penetrate non-metallic surfaces. Further, millimeter waves are non-ionizing and thus not dangerous to the human body [7].

Miniaturized, radar-on-chip millimeter-wave (mmWave) devices such as Google Soli [26], Texas Instruments' AWR/IWR sensors,[1] and Silicon Radar[2] have democratized radar sensing in several areas of Computer Science, because now the hardware has become a commodity. As a technology that can be embedded beneath surfaces, a miniaturized radar sensor can be incorporated into wearables and IoT devices, where space is at a premium [23]. Commercial applications include occupancy sensing[3], elderly fall detection, and through-wall sensing.[4]

At the same time, there is an increased effort in empowering industrial robots to assist humans for human-robot interaction (HRI) and human-robot collaboration (HRC). Cobots such as ABB's YuMi [1] or KUKA robots [21] are ensuring safety and control by bringing in the versatility and adaptation to dynamic working environments. Traditional enablers of HRI and HRC have been cameras, laser range finders, inertial measurement units or combinations therein. In this paper, we propose millimeter-wave radar as an emerging, alternative technology. Since mmWave radars have small form factors due to small wavelengths, they can be mounted on stationary parts of a cobot body and perform well in challenging industrial environments such as low or direct light, clutter, smoke or steam. Finally, a mmWave radar can enable a standalone sensing method or complement existing technologies like the above-mentioned enabler devices. Gesture recognition is useful in these contexts for safety and control, though we note that such a gesture recognition system is not limited to industrial environments. For example, it can replace touch displays used in shopping malls and restaurants, to alleviate e.g. cross-contamination in pandemic situations. Another aspect is the control of appliances using gestures in smart homes, where Pantomime can enable highly accurate, privacy-preserving gesture recognition that works also in darkness or smoky conditions.

It has been argued that sensing in the radiofrequency (RF) spectrum eschews spatial information for temporal resolution [61], though this is only the case if Doppler features are used for recognition, which only distinguishes gesture articulation on the basis of translational motion. In this paper we show that radar signals can easily leverage spatial information through point clouds (unordered sets of 3D points) further processed from range, angle, and Doppler information. By configuring a low cost commercial radar for gesture recognition, we show that point clouds create unique spatio-temporal structures that can be used to recognize complex movements, including e.g. bimanual and circular gestures. Converting the raw analog to digital conversion (ADC) data from the antenna arrays into point clouds helps massively to reduce the data size from tens of gigabytes to a few megabytes, thus, helping fast data transfer, processing and running powerful machine learning algorithms. In

[1] http://www.ti.com/sensors/mmwave/overview.html
[2] https://siliconradar.com/
[3] https://www.xethru.com
[4] https://walabot.com

addition, unlike spectrograms of Doppler signals, point clouds generated from even complex gestures are easily interpretable because the motions occur in the 3D space.

Motion gesture recognizers require modeling the skeletal body structure [22, 57], which is challenging for a system like ours because the point cloud provided by the radar signal is sparse due to the wavelength and hardware limitations. To solve this, researchers have proposed point cloud voxelization [68], but this is memory-intensive and may preclude real-time interaction. As a solution, we advance a pioneering work that classified stationary 3D shapes directly on point clouds [40, 41] and introduce Pantomime, our mid-air gesture classifier. (Pantomime stands for **P**oint cloud-based **A**ctivity recognition **N**e**T**w**O**rk using **MI**lli**ME**ter wave radar.)

Pantomime is designed to recognize sparse gesture motions by means of a hybrid model architecture for optimized spatio-temporal feature extraction. To improve the recognition results with motion point clouds, the following technical contributions have been developed and implemented in Pantomime's neural network architecture: i) Spatio-temporal feature extraction of point clouds, ii) addressing the sparsity problem, and iii) reducing model complexity to prevent overfitting when addressing the above two problems. As explained later, a Pointnet++ and LSTM combination is used for addressing the first problem, whereas, the sparsity problem is addressed by using the aggregated point cloud of the whole gesture as input to another PointNet++ block. We address the third problem by sharing weights among the parallel PointNet++ blocks. The combination of these three solutions culminates in a 4D point cloud classification architecture which defines our main technical contribution. We further address key issues that are common to indoor environments such as environment dependence due to static reflections and the effect of articulation speed, angle, and distance to the sensor. To do so, we use a combination of preprocessing techniques, data augmentation, and adapt different radar configurations at operational time, to be described later. We also compare the performance of Pantomime against four state-of-the-art point cloud classifiers, including a motion gesture recognition system from Microsoft Kinect point clouds [19] and an activity recognition scheme using mmWave point clouds [51]. We show that Pantomime outperforms current state-of-the-art approaches by a large margin.

Our main contributions can be summarized as follows:

(1) A mid-air gesture recognition system using (a) sparse 3D point clouds computed from radar signals, and (b) a deep learning architecture that feeds on the point cloud directly.
(2) A user evaluation on a challenging set of 21 gestures, including one-hand, bimanual, linear, and circular gestures, performed by 45 participants in two indoor environments.
(3) Additional experiments that demonstrate the robustness of our system under various articulation conditions, including speed, angle, and distance.
(4) A mmWave point cloud gesture dataset along with our radar sensor configuration, trained models, and source code, to allow others to reproduce and build upon our results [36].

2 RELATED WORK

The RF spectrum spans 30 Hz to 300 GHz. RF technology is widely used e.g. in military and telecommunication applications [52], however only recently it has been exploited for gesture interaction. In the following we review previous work that has tapped into various technologies to recognize motion gestures. Then, we review related work that specifically focused on radar sensing for gesture recognition.

2.1 Motion Gesture Recognition Technologies

Mid-air and full-body gesture recognition technology includes RGB cameras, depth sensors, WiFi, Leap Motion, or Google Soli; cf. Table 1. Vision-based systems such as Kinect provide 2D color frames, full-body 3D skeleton, and 3D point clouds [27]. However, they suffer from occlusion and darkness issues, and (because of the camera)

often they raise privacy concerns [8]. Extensive surveys on vision-based gesture recognition were published by Wachs et al. [60] and Rautaray and Agrawal [43].

Table 1. Popular systems for gesture recognition. IR: infrared, RSS: received signal strength, CSI: channel state information

Technology	Spectrum	Data Type	Size	Resolution	Range	Privacy	Occlusion	Darkness
RGB camera [33, 38]	visible light	images	small	high	>10 m	✗	✗	✗
MS Kinect [19, 35]	visible light, IR	point cloud	large	high	5 m	✗	✗	✓
Leap motion [16, 29]	IR	point cloud	small	high	1 m	✓	✗	✓
Google Soli [26, 61]	60 GHz	range, angle, Doppler	small	medium	30 cm	✓	✓	✓
Aryokee [25, 67]	5.46 GHz	range, angle, Doppler	large	low	9–12 m	✓	✓	✓
WiFi [28, 39]	2.4/5 GHz	RSS, CSI, Doppler	medium	low	10 m	✓	✓	✓
Pantomime (ours)	77 GHz	point cloud	small	medium	5 m	✓	✓	✓

As can be observed in Table 1, Pantomime is positioned in a unique region of the RF technology landscape as a miniaturized medium-resolution medium-range high-frequency RF sensing approach, which is privacy-aware and robust to occlusion, weather conditions, and visibility issues. High operating RF frequencies allow for smaller radar sensors, so that they can be placed everywhere. Unlike Soli, which operates in short ranges, Pantomime is ideal for sensing full-body motion gestures. The chirps used in the Soli sensor are shorter, causing a higher velocity resolution in short ranges than Pantomime, while Pantomime sacrifices velocity resolution to leverage spatial resolution in order to detect full-body movements in mid ranges. Further, we argue that a medium-resolution device provides the right balance for device performance: If the carrier frequency is too low, it is not possible to recognize fine-grained gestures. But if it is too high, the signal attenuation increases and occlusion problems may arise.

2.2 Gesture Recognition from RF Sensing

RF gesture recognition systems can be categorized by the operating frequency into sub-6 GHz and millimeter waves. Sub-6 GHz gesture recognition either leverages commodity narrowband devices with low-granular received signal strength [2], multi-carrier channel state information [24, 28, 58, 59], Doppler [39], or measure time-delay of individual multipath components via e.g. frequency modulated continuous wave (FMCW) radar [25, 67]. Below 6 GHz, gesture recognition is limited as the wavelengths are larger than 5 cm and this region has a small bandwidth. Additionally, these wavelengths mandate large apertures for antenna arrays; for example, the 12×12 antenna array in [25] required a 30×30 cm^2 RF front-end. In contrast, mmWave sensing features high bandwidths (4–7 GHz), antenna apertures of few centimeters, high directivity, and unique scattering properties.

Sensing solutions for mmWave radars in general are either model-driven [26, 62] or data-driven [5, 31, 49, 51, 61, 68]. Model-driven approaches are limited to few gestures in downstream tasks (e.g. classification or tracking) while data-driven solutions allow recognition of larger gesture sets. Data-driven approaches in general combine convolutional neural networks (CNN) and long short term memory (LSTM) modules to process Doppler, range-Doppler, and/or angle-Doppler features [5, 49, 61]. Unfortunately, the interpretability of these features is limited and, consequently, distinguishing simultaneous movement of different body parts becomes difficult.

2.3 Radar Sensing in HCI

The potential of radar sensing in HCI has been demonstrated in detecting subtle, nonrigid motion gestures [26, 61], mostly articulated with hand and fingers: e.g. rubbing, pinching, or swiping. The Magic Carpet [37] was among the first systems using radar sensing to detect body motion, though it required excessive space and a complex setup.

Recent research focused on radar sensing for augmented reality [11] and interaction with everyday objects [56]. Other applications include music [6, 48], activity recognition [4], or through-wall tracking [3, 66]. In the health domain, radar enables non-invasive monitoring of glucose levels [34, 50], sleep monitoring [42], and emotion tracking [65]. Finally, a radar sensor can distinguish various materials when placed on top of it [30, 63].

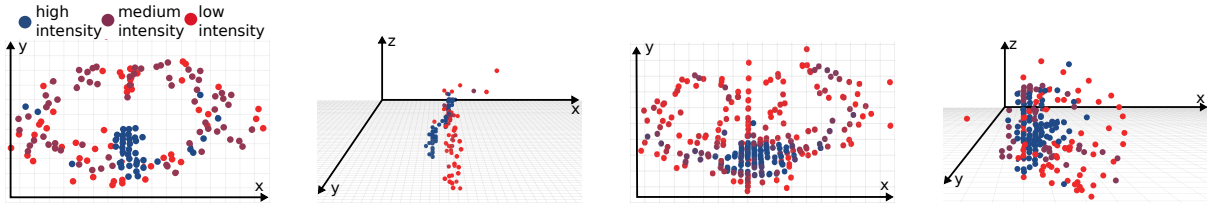
2.4 Gesture Recognition with Point Clouds

A point cloud is a sequence of *frames*, each comprising an unordered set of points in a 3D space. Gesture recognition from point clouds has been used for hand pose estimation [13] and hand motion recognition [19, 35]. Hand pose estimation requires only spatial information, whereas motion gesture recognition demands spatio-temporal information. As hinted previously, recent approaches are based on CNNs [14, 15, 51, 68]. Since conventional CNNs cannot handle 3D spatial coordinates, either multi-view CNNs are used to project the point cloud onto multiple planes [14, 53] or a 3D CNN is applied on a voxelized point cloud [35, 51, 68]. Drawbacks of these methods include information loss due to projection or voxelization, and cubical growth of computational complexity and memory, resulting in inefficient computations of sparse 3D convolutions.

To address these issues, PointNet [40] and PointNet++ [41] classify and segment stationary 3D shapes *directly* from point clouds. We advance this work to sense mid-air gestures of point clouds from commercial mmWave radars. In a similar attempt, Min et al. [32] translated RGB-depth information from a Kinect sensor into point clouds and used PointNet++ to classify hand gestures. Unfortunately, their architecture is not described in sufficient detail for us to reproduce and there is no software implementation publicly available.

3 SPATIO-TEMPORAL PROPERTIES OF RADAR SIGNALS

A radar sensor sends an electromagnetic wave with a transmitting antenna (Tx) which is reflected, scattered, and absorbed by objects in the environment, and then detected with a receiving antenna (Rx). In this paper, we work with the IWR1443 sensor from Texas Instruments, an FMCW–MIMO radar sensor that operates in the 77 GHz RF band. FMCW is used for range estimation and MIMO computes both elevation and azimuth angles, yielding 3D motion coordinates that can be converted into a point cloud for every frame (see Section A.1 for the detailed point cloud extraction process). Our mmWave point clouds exhibit unique spatio-temporal characteristics for mid-air gesture interaction. In this section, we discuss these properties to motivate the design of Pantomime.



(a) Two-hand lateral-to-front (b) Push-down (c) Two-hand circle (d) Circle clockwise

Fig. 2. Different views of four gestures from our dataset.

3.1 Spatial Properties

Figure 2 shows point cloud examples computed for gestures that are articulated in different planes; e.g., gestures *(a)* and *(c)* are conducted in the XY-plane, whereas *(b)* and *(d)* are conducted in the XZ-plane. The figure also shows the intensity of each point; i.e., the amount of energy reflected at each point. As can be observed, intensity

is high (blue) for points related to torso reflections, while it is low (red) for points related to the arms and hands. Further, the point cloud related to the torso is denser than for the arms, because the torso is a stationary object and has a larger radar cross-section, while arms are in motion and have a smaller radar cross-section. Consequently, point density and point intensity are not uniform within the point cloud. We also observe some noise in the point cloud, caused by reflections from nearby objects. Fortunately, reflections from a human subject have a higher density, due to the smaller distance to the radar sensor, and so are easier to detect.

As evidenced in Figure 2, gestures performed in the XY-plane are captured with high granularity. This behavior is attributed to the following reasons. First, the point cloud is extracted after a series of range-based fast fourier transform (FFT), Doppler-based FFT, constant false alarm rate (CFAR), and angle-based FFT operations (see Section A.1). The CFAR algorithm [44] relies on range and Doppler signal, so that the detected cloud points are sensitive to the motion of the detected object. In addition, the radar device has different antenna resolutions. For example, motion in the X axis is resolved by the horizontal antenna array (Section A.1), which has 8 virtual elements providing a resolution of 14.3°, while the two virtual antennas in Z axis provide a resolution of only 57°. Due to these reasons, the granularity in the XY-plane is higher compared to other planes. In the XZ-plane, linear motion can be captured with high resolution, whereas circular motion has lower granularity; c.f. Figure 2(b) vs. Figure 2(d).

Another interesting spatial characteristic of the captured point cloud is that it is denser when the gesture is performed closer to the radar sensor. This is explained by the fact that the radar cross-section is smaller when the user is located farther away from the sensor; therefore cloud points corresponding to arms and hands become sparser in that case. There is also an effect attributed to the angular resolution of the radar sensor. For example, the distance between two points captured at a resolution of θ degrees is proportional to $R\theta$, which depends on the range R of the radar sensor. This property increases the density of the point cloud by a scaling factor R which depends on the distance to the detected object. We will show in a later experiment that the operational distance at which gestures can be articulated is a critical design parameter.

3.2 Temporal Properties

The number of 3D cloud points is different at every frame. This makes gesture recognition a challenging task, as it must analyze the evolution of the point cloud over time. By way of example, Figure 3 shows a 2 s long gesture (*two-hand lateral-to-front*), where each snapshot in the figure contains points collected over 250 ms.

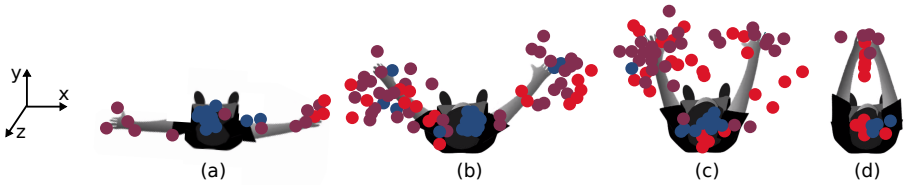


Fig. 3. Progression of 'two-hand lateral-to-front' gesture over four consecutive time frames (a, b, c, d) overlaid with the computed point cloud from the radar.

As can be observed in the figure, the user has both hands placed along the X axis before starting the motion *(a)*, and in the final state *(d)*, arms converge to the Y axis. We can see that the two resting stages *(a, d)* have a smaller number of points and are concentrated in a smaller region compared to the motion stage *(b, c)*. Additionally, the points detected from the two hands are not equal nor symmetric. It can also be observed that the arm motion constitutes a spatio-temporal structure due to the relative position of the torso and arms that relates to the preceding and subsequent snapshots. The relative positions of the point cloud allows us to detect gestures in

opposite directions, yet with similar structure (cf. 'push-down' and 'lift-up' or 'push' and 'pull' gestures in Figure 7).

From these observations we can see that, even though a skeletal structure of the body is not apparent, the combination of the torso and arm movements reveal unique spatial structures in the point cloud and this can be exploited for motion gesture recognition. These geometrical challenges are addressed by PointNet++, our underlying *spatial* recognizer, which we adapt as explained in the next section. However, another important aspect of point cloud data is the variability in gesture execution between subjects, either in terms of e.g. speed and angular position. Therefore, we need to make our recognizer resilient to variations in gesture articulation.

4 SYSTEM ARCHITECTURE

The architecture of Pantomime consists of two main blocks: *Preprocessing* and *Classification*. The preprocessing block produces a clean sparse point cloud at every frame and the classification block extracts spatio-temporal features to characterize each gesture. In the following, we describe each of these blocks in detail.

4.1 Point Cloud Preprocessing

As shown in Figure 4, the preprocessing block consists of four stages: outlier removal, rotation and translation, frame divider, and resampling.

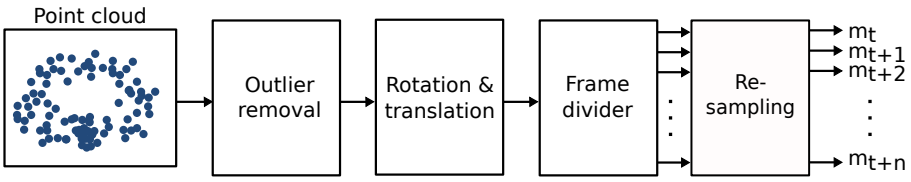


Fig. 4. Point cloud preprocessing block.

4.1.1 Outlier Removal. As hinted previously, the point cloud contains noise because of scatterings and reflections from the environment. This happens often in cluttered environments. Fortunately, this noise is sparse and can be removed using the following procedure.

The point cloud is received at a fixed frame rate but the number of points in each frame can vary due to the CFAR operation, see Section A.1 for more details. We first wait until all frames corresponding to a single gesture are collected over time, and then aggregate all the frames to construct the point cloud of the gesture; see Figure 2 for some examples. When the whole point cloud is available, it is easy to separate the outliers based on point density. For this, we apply the DBSCAN algorithm [12] with $\epsilon = 1$, which is the maximum distance between two points that can be considered as neighbors, and set the minimum number of points in a cluster to 3. We fine-tuned these parameters using grid search. The output of the DBSCAN algorithm produces a major cluster around 0.5 m diameter, representing the points reflected from the user's body and few small clusters representing the environmental noise. We keep the points in the major cluster and consider all other clusters as outliers.

4.1.2 Rotation and Translation. Once the major cluster is identified, we measure the cluster centroid's angle with respect to the boresight angle (i.e., in relation to the centre-line of the antenna) and estimate its distance with respect to a reference distance. In our experiments the reference distance is set to 1.5 m. If the centroid deviates from the reference angle and reference distance, the detected object is rotated and translated by the deviated amount. This is done to normalize the position and to reduce the effect of the position on the recognition accuracy.

4.1.3 Frame Divider. The aggregated point cloud is rearranged into ordered frames, based on the time at which each point was received. To reduce the time complexity of our model and to study the effect of the number of frames on the recognition accuracy, we decrease the original number of frames received from the radar device; e.g. for a 2 s gesture we reduce the number of frames from 100 to 2, 4, or 8 but retain all the points in those frames. To do so, we apply *time decay* on the points such that we rearrange the first set of k/f points in the first frame, where k is the total number of points in a gesture and f is the desired number of frames; second set of k/f points as the second frame, and so on. Although this procedure fixes the number of frames, we still have gestures with different number of points per frame. However, the PointNet++ architecture (see 'Framewise Spatial Feature Extraction' section) dictates a fixed number of points in each frame for all the gestures, so we apply a resampling algorithm, as explained next.

4.1.4 Resampling. We resample the number of points in a frame to achieve a fixed number of points in each frame while preserving the shape and the density of the point cloud. Following Cohen et al. [9], we use Agglomerative Hierarchical Clustering (AHC) for upsampling and the K-means algorithm for downsampling. In each step of the AHC algorithm, the centroid of each cluster is added to the point cloud as a new point, which prevents producing singleton clusters. We run AHC iteratively until reaching our predefined fixed number of points in a frame. For downsampling, we use K-means with K equal to the fixed number of points per frame and select the centroids of the clusters as the points in the point cloud.

4.2 Point Cloud Classification

We propose a hybrid architecture combining the PointNet++ architecture followed by LSTM modules for frame-wise spatio-temporal feature extraction, as shown in Figure 5. We follow this approach because, on the one hand, in highly sparse point clouds, applying PointNet++ on each frame may not capture the spatial features efficiently because fine-grained local features (e.g. skeletal structure of the hand) are not well preserved between consecutive frames. On the other hand, applying PointNet++ on the aggregated point cloud will preserve spatial features but the directionality of certain gestures (e.g. clockwise circling and anti-clockwise circling) would be lost because temporal features are not captured.

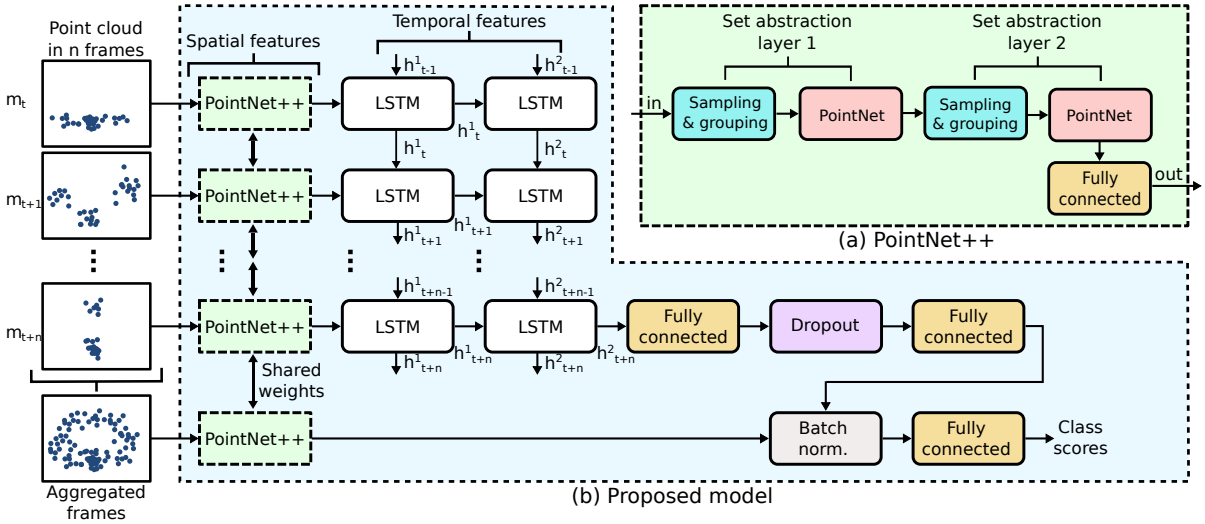


Fig. 5. Sparse point cloud classification architecture.

4.2.1 Framewise Spatial Feature Extraction. PointNet++ is a neural network architecture mimicking the functionality of classic 2D CNNs to extract hierarchical spatial features from 3D point clouds [41]. Local features are first extracted to capture important structures in small regions, and the local features are further grouped into larger units to extract higher level features. The same steps are repeated until the features of the whole point set are computed.

To capture features of the point cloud iteratively, PointNet++ uses multiple set abstraction levels to mimic the multiple convolution levels in CNNs. A single-set abstraction level consists of three important layers: *Sampling layer*, *Grouping layer* and *PointNet layer*. The Sampling layer outputs a set of points from the input points that define the centroids of local regions. The Grouping layer then builds sets of local regions by selecting "neighboring" points of the selected centroids. Finally, the PointNet layer encodes the local region patterns into feature vectors.

Mathematically, the last set abstraction level maps a given unordered point set $\{z_1, z_2 \dots z_N\}$ with $z_i \in \mathbb{R}^n$ to a sparser subset with higher dimension $\{y_1, y_2 \dots y_M\}$ with $y_i \in \mathbb{R}^m$, where $m > n$ and $M < N$. The mapping function $q(\cdot)$ can be written as

$$q(z_i) = y_i = \left(\mathrm{FC}_1 \circ \max_{z_j \in S(p_i),\ j=1,\dots,N} \circ \mathrm{FC}_2 \circ g\right)(z_{p_i}, z_j) \tag{1}$$

where the $\circ$ operator denotes a function composition; i.e. given two functions $a(\cdot)$ and $b(\cdot)$, $(a \circ b)(\cdot) = a(b(\cdot))$.

In Equation 1, FC_1 and FC_2 are fully-connected layers, and g is a grouping layer that finds all the points within a radius from the centroids z_{p_i} in the point set z_i to construct local regions based on the spatial neighborhoods $S(p_i)$. The centroids are selected using the farthest-point sampling algorithm [10].

Accordingly, we feed the n frames from the preprocessing stage to n parallel PointNet++ layers, as shown in Figure 5(b), to extract spatial features from the frames which are then transferred to two LSTM layers to extract temporal features. Further, to reduce the size of the temporal feature vector and obtain fine-grained features, we add two fully-connected layers with a dropout layer (0.5 drop rate) after the LSTM layers, for regularization purposes.

Weight sharing. To reduce the model's complexity in terms of trainable parameters and to prevent overfitting, our model shares all weights among parallel PointNet++ blocks. Each PointNet++ block has 1.47M trainable parameters, so for example eight parallel PointNet++ blocks would comprise 11.76M parameters. By sharing these weights among parallel blocks, we decrease the trainable parameters to 1.47M. Note that each of these PointNet++ blocks is responsible for calculating the spatial representation of the input frame, without considering the temporal properties. These blocks therefore calculate the same set of fine-grained features but with different values for each input frame.

4.2.2 Temporal Feature Extraction. Having extracted the spatial features of the point cloud with PointNet++ at every frame, we feed those features to two stacked LSTM layers to extract temporal features. We experimented with higher numbers of LSTM layers but without substantial improvement (see Table 2). We have the following equations for an LSTM unit:

$$f_t = \sigma_g(W_f x_t + U_f h^p_{t-1} + b_f) \tag{2}$$
$$i_t = \sigma_g(W_i x_t + U_i h^p_{t-1} + b_i) \tag{3}$$
$$o_t = \sigma_g(W_o x_t + U_o h^p_{t-1} + b_o) \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + U_c h^p_{t-1} + b_c) \tag{5}$$
$$h^p_t = o_t \odot \sigma_h(c_t) \tag{6}$$

where $\odot$ is the hadarmard product, x_t is the input at time step t, $h^p \in \{h^1, h^2\}$ is a hidden state, h^p_{t-1} is the hidden state from the previous time step, W and U are weight matrices, b is the model bias, σ is the sigmoid activation function, and f, i, o, c are forget, input/update, output, and control gates, respectively.

The output of temporal feature extraction can be defined as follows:

$$L_k = (\text{LSTM}_1 \circ \text{LSTM}_2 \circ \text{FC}_3 \circ q \circ q)(z_i) \tag{7}$$

where $k \in \{1, 2, \ldots, K\}$ denotes the frame index in a sequence of K frames, $\text{LSTM}_r(\cdot) = h^p_t(\cdot, h^p_{t-1})$ is an LSTM layer and $r \in \{1, 2\}$ denotes the layer number, and FC_3 is another fully-connected layer.

4.2.3 Aggregated Feature Extraction and Classification. As we collect two sets of features from a hybrid model; the first set is the output of spatio-temporal features from n frames and the second set is the spatial features of the aggregated point cloud. We concatenate these two feature vectors to form a consolidated feature vector of size 512, before applying a batch normalization layer to equalize two different sets of feature vectors, since the features come from two different sources and have two different scaling factors. Finally, we use a fully connected layer with no activation function to calculate the class score vector of size C, where C denotes the number of gesture classes in our dataset. The recognized gesture class is computed via argmax operation.

Mathematically, the model can be represented as follows:

$$\text{Output} = (\text{FC}_4 \circ \text{Norm} \circ L_{k=K} \cup \alpha \circ q \circ q)(z_i) \tag{8}$$

where z_i is the aggregated point cloud, $\text{Norm}(\cdot)$ is the batch normalization layer, and FC_4 is a fully-connected layer.

4.3 Improving the Recognition Performance

We train our classifier with different persons articulating the gestures at a fixed distance of 1.5 m and at the boresight angle (i.e., directly in front of the radar) as explained later in Section 5. Then, to improve the generalizability of Pantomime, we apply on-the-fly data augmentation during training. This is done by (1) translating the recorded point cloud up to 10 cm, (2) scaling the point cloud by a factor of 0.8–1.25, (3) applying jitter (random translation) using a Gaussian distribution $\mathcal{N}(0, 0.01)$ and (4) applying clipping of 0.03 m. Further, to improve the robustness of our model, we consider the effect of different environments, distances, and angles, as explained in the following sections.

4.3.1 Different Environments. First of all, as already mentioned in Section 4.1.1, we remove the reflection noise of the environment during the preprocessing stage via outlier detection. Then, during the radar operational stage, we reduce the impact of far and nearby objects as follows. On the one hand, to reduce the impact from reflection noise coming from static objects that are far away from the sensor, the maximum range of the radar is fixed to the distance of the user by adjusting the analog to digital converter sampling rate (Equation 11). On the other hand, to reduce the impact from reflection noise coming from nearby objects, we remove the first half of the range bins in the point cloud estimation step.

4.3.2 Different Articulation Angles, Speeds, and Distances. As mentioned in Section 4.1.2, we translate and rotate the point cloud to the reference articulation condition (1.5 m, 0°). Regarding different speeds, since the system does not generate points for the static parts of the body, the frame divider block shown in Section 4.1.3 reduces the effect of speed in the preprocessed point cloud. However, different distances have more impact on the point cloud distribution than the angles or speeds, even after preprocessing. Thus, to make the model resilient to different distances, we apply the following computations at the radar operational stage. When the human subject enters the area of interest, the radar detects the subject's range and adjusts the chirp configuration (see Section A.2) to optimize a high range and velocity resolution at that range. Consequently, we use 4 chirp configurations for 4

distance ranges: 0–2 m, 2–3 m, 3–4 m, and 4–5 m. As the point cloud changes with increasing distances, we train the model from 1.5 m distance and incorporate later a smaller set of training gestures articulated at 3, 4 and 5 m.

5 EVALUATION SETUP

In this section we describe our device configuration, the experimental environments, gesture sets, participants, and model fine-tuning. In the next section we report the recognition performance of Pantomime in various input conditions.

5.1 Device Configuration

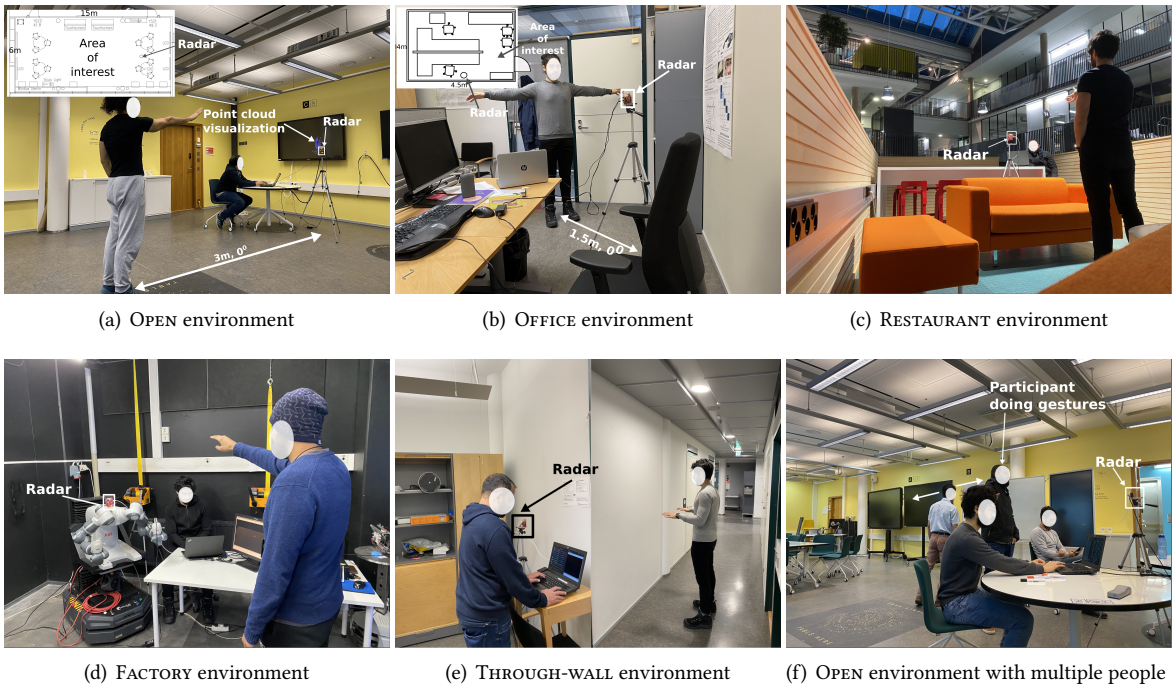
Our radar device (IWR1443) uses a built-in Cortex-R4F microcontroller (clocked at 200 MHz) and the universal asynchronous receiver-transmitter protocol for data transfer. The device is resource-constrained, both in terms of computation and throughput. Therefore, it requires a proper configuration of the *chirp* and *frame* properties (see Section A.1). We configure those parameters in order to achieve a frame rate of 30 fps, a range resolution of 0.047 m, a velocity resolution of 0.87 m/s, and a maximum velocity of 6.9 m/s up to a maximum range of 5 m. The starting frequency is 77 GHz and our selected range resolution dictates a bandwidth of 3.19 GHz. Note that the number of cloud points detected in each frame is not constant, even when using a fixed frame rate, mainly due to the CFAR operation. The transmit power of the radar is 12 dBm and the receiver gain is set at 48 dB.

5.2 Experimental Environments

We chose five indoor environments (Open, Office, Restaurant, Factory and Through-wall) as shown in Figure 6 to evaluate the performance of Pantomime. Open is an environment of 91 m^2 with no furniture around the radar sensor. We use it for single and multiple people experiments. Office is a cluttered environment of 18 m^2 with nearby objects. Restaurant is a 23.36 m^2 cluttered environment. Both Restaurant and Factory environments are multipath-heavy, representing thus more challenges for gesture recognition and fitting the chosen applications. On the one hand, the restaurant consists of typical clutter such as tables, chairs, sofas, and walls within a 3m radius from the radar sensor. On the other hand, the factory environment consists of two robots (a cobot and a robot within a 2m distance from each other), tables, chairs, and 3D printing machinery within a 3m radius from the radar. The radar is mounted on the stationary parts of a YuMi cobot and the user performs gestures within a 2 m distance from the radar. In Through-wall, the radar penetrates a typical indoor wall that partitions a room and a corridor ($\approx$ 10 cm thick single wall consisting of two decoupled and insulated plaster partitions). Each plaster partition was $\approx$ 4 mm thick.

5.3 Gesture Sets

We collected data for 21 types of mid-air gestures, as shown in Figure 7. As we have designed a mid-range gesture recognition system, the gestures require movements of body parts with relatively large radar cross-section to enable detection up to 5 m of distance. We divide the gestures in two sets (Easy and Complex) based on their execution difficulty. The first 9 gestures (a–i) belong to the Easy set and comprise single-hand gestures that are easy to perform and remember. The remaining 12 gestures (j–u) belong to the Complex set and comprise bimanual, linear, and circular gestures. Finally, we also consider all gestures from both sets in a later experiment, which we will refer to as the All set. We conduct experiments on each set and environment separately, in order to challenge the performance of our recognizer.



(a) Open environment (b) Office environment (c) Restaurant environment

(d) Factory environment (e) Through-wall environment (f) Open environment with multiple people

Fig. 6. Our indoor environments. In all conditions, the experimenter could see the point cloud of the articulated gestures in real-time in a nearby monitor (left) or a nearby laptop (right).

5.4 Participants and Procedure

We recruited 45 participants (13 female) aged 18–55, weight 50–110 kg, and height 1.55–2 m for data collection for training, validation and test sets. They were unpaid volunteers recruited via mailing lists, external advertisements, news boards, and flyers distributed among several of our university's departments.

Before starting with the data collection, the experimenter demonstrated how each gesture should be performed through a video. Then, while the participants performed each gesture, the experimenter verified that it was articulated successfully by visualizing the point cloud in a nearby monitor or laptop (see Figure 6. In case of a wrongly executed gesture, the participant was asked to repeat it again.

Our final dataset comprises 22291 articulation samples in total. Users articulated the gestures in 1–3 s on average, so that our dataset accounts for 18.5 h of recording time. After performing all trial executions of each gesture class, participants could have a rest (if desired) before advancing to the next gesture.

5.5 Evaluation Metrics

We evaluate the performance of Pantomime using recognition accuracy (Acc.) and the area under the ROC curve (AUC), which quantifies the discriminatory power of the classifier. We also report the confusion matrix to analyze performance further and inspect conflicting gestures. By way of comparison, a random classifier would achieve $1/21 = 4.76\%$ classification accuracy and 50% AUC on the All gesture set. Our model was trained on a single Tesla V100 GPU in about 5 h.

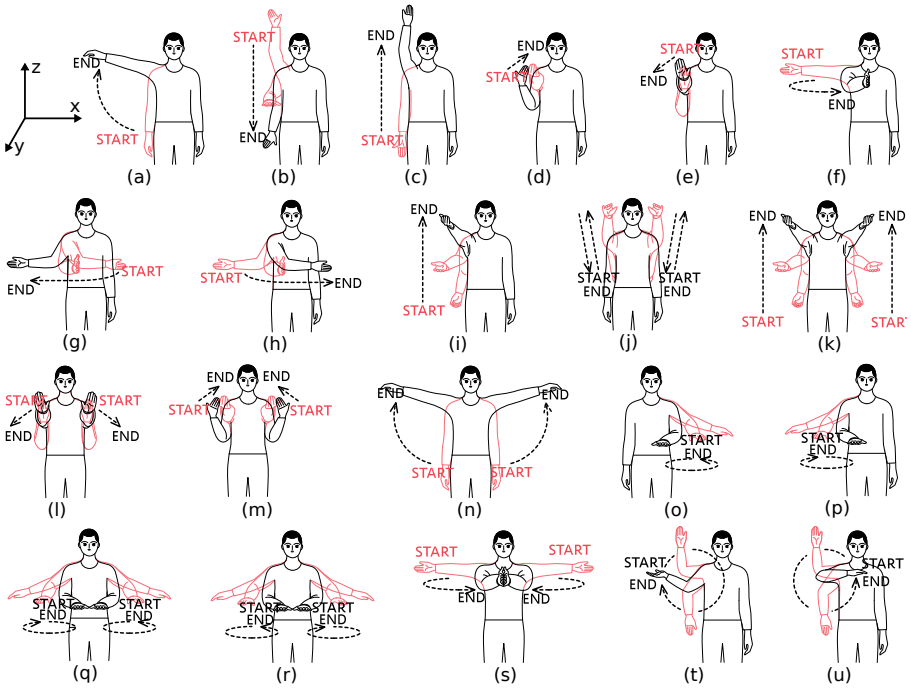


Fig. 7. Gesture set used in our experiments. Easy set: (a) 'lateral raise', (b) 'push-down', (c) 'lift', (d) 'pull', (e) 'push', (f) 'lateral-to-front', (g) 'swipe right', (h) 'swipe left', (i) 'throw'. Complex set: (j) 'arms swing', (k) 'two-hand throw', (l) 'two-hand push', (m) 'two-hand pull', (n) 'two-hand lateral-raise', (o) 'left-arm circle', (p) 'right-arm circle', (q) 'two-hand outward circles', (r) 'two-hand inward circles' (s) 'two-hand lateral-to-front', (t) 'circle clockwise', (u) 'circle counter-clockwise'.

5.6 Model Fine-tuning

Pantomime has three main components that should be optimized: the number of parallel input frames, the number of set abstraction levels in PointNet++, and the number of LSTM layers. To decide the best model architecture, we begin with 2 parallel frames, 2 set abstraction layers, and 2 stacked LSTMs. Then we increase each parameter while keeping the others constant, until the overall accuracy starts to decrease for gestures in the All group. We empirically set the number of points in each frame to 64.

We split all the recorded 7251 gestures from Open (2631 samples) + Office (4620 samples) into train/validation/test partitions using a user-dependent 60:20:20 ratio, i.e. 60% and 20% of the data (data from 33 users) is used for training and validation and the remaining 20% of the data (data from 8 users unseen by the training and validation sets) is used for testing. As shown in Table 2, the combination that achieves the highest accuracy (95%) on the validation set is 8 parallel frames as input, 2 set abstraction levels in PointNet++, and 2 stacked LSTM layers. After tuning the model hyperparameters, we used the whole training and validation sets for training the final model, and evaluate it on the held-out test set.

During the training phase, we augment the point cloud on the fly using three geometric transformations as mentioned in Section 4.3. The effect of different augmentation schemes is shown in Table 3. As observed, when both training and test data are not augmented, the model achieves a 93.89 % accuracy. The model performs worst when tested on augmented data without training on the augmented, which is a mere 31.2% accuracy. This justifies the selection of data augmentation which helps to generalize for deviations that occur in practical situations such

Table 2. Parameter Optimization of Pantomime.

No. of frames	2	4	8	10	8	8	8	8
No. of set abstraction layers	2	2	2	2	1	3	2	2
No. of LSTM layers	2	2	2	2	2	2	1	3
Accuracy (%)	93	94	95	93	94	94	93	94

as translation or disorientation of the participant from the reference point and for different body sizes of the participants. The highest accuracy is achieved when only the training set is augmented, where the accuracy is almost 1.1% higher than the model without any augmentation. This shows that data augmentation makes the model invariant to various geometric transformations during experiments, a desirable property when handling point clouds for gesture recognition.

To evaluate the effect of outlier removal, we considered three scenarios. In the first scenario, we train the model on the training set that has outliers (w/ outlier) and evaluate it using the test set that has outliers and achieved an accuracy of 88.74%. Next, we use the pre-trained model on the data after applying outlier removal (w/o outlier) to evaluate the test set with outliers, and achieved an accuracy of 59.2%. Finally, we compare these accuracies with the case where outliers are removed in both training and test sets, which is 95%, a 7% improvement from the first scenario.

Table 3. Effect of data augmentation.

	Data augmentation			
Training set	✗	✗	✓	✓
Test set	✗	✓	✓	✗
Acc. (%)	93.89	31.20	94.4	95.01

Table 4. Effect of outlier removal.

Train	w/ outlier	w/o outlier	w/o outlier
Test	w/ outlier	w/ outlier	w/o outlier
Acc. (%)	88.74	59.20	95.01

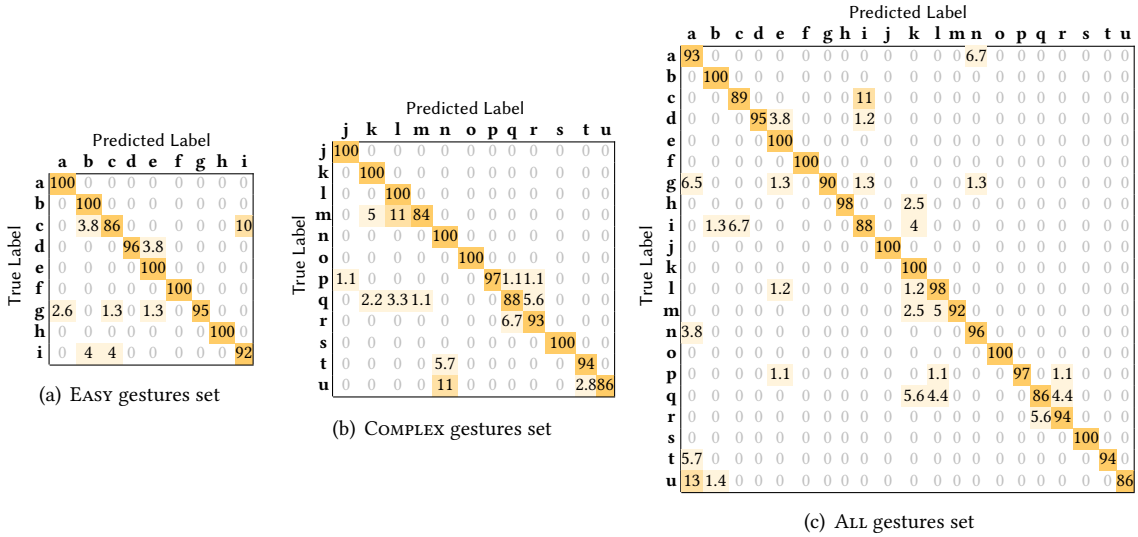
6 RESULTS

We evaluate the performance of Pantomime on our three gesture sets (Easy, Complex, and All) with increasing articulation difficulty. We train and test our model on the same data splits as used in the previous fine-tuning experiments (33 train users, 8 test users) to avoid any data leakage. Note that this procedure is more realistic and challenging than shuffling all the data and creating the train/test partitions later [17].

6.1 Overall Performance

Figure 8 shows the confusion matrices for our three gesture sets. As can be observed, Pantomime is remarkably accurate. Regarding the Easy gesture set, it achieves 96.6% accuracy. According to the confusion matrix in Figure 8(a), for 8 out of 9 gestures, it achieves an accuracy over 92%. The few gestures having conflicts with other gestures have unsurprising traits, e.g. (b) 'push-down', (c) 'lift', and (i) 'throwing' are performed in the YZ-plane; in (a) 'lateral raise' and (g) 'swipe right' the point cloud moves from left to right; and both (d) 'pull' and (e) 'push' operate along the Y axis but in opposite directions.

Regarding the Complex gesture set, Pantomime achieves 95.1% accuracy, which is, surprisingly for us, just 1.5% lower than the performance observed in the Easy gestures. According to the confusion matrix in Figure 8(b), for 9 out of 12 gestures Pantomime achieves an accuracy over 93%. Again, the conflicting gestures have common



(a) Easy gestures set

(b) Complex gestures set

(c) All gestures set

Fig. 8. Confusion matrices. Each cell denotes classification accuracy, in percentage.

characteristics; e.g. (k) 'two-hand throw', (l) 'two-hand push' and (m) 'two-hand pull' are similar types of gestures where the arms move either away from the body (k and l) or towards the body (m); Similarly, (q) 'two-hand inward circles' and (r) 'two-hand outward circles' are virtually the same gesture but operating on opposite directions; and (t) 'circle clockwise', (u) 'circle counter-clockwise' and (n) 'two-hand lateral raise' are circular gestures that operate in the XZ-plane.

Finally, Pantomime achieves 95% accuracy for the All gesture set, which is just 0.1% lower than the accuracy achieved on the Complex set, indicating the robustness of Pantomime for increased gesture complexity and number of gestures. The confusion matrix in Figure 8(c) is consistent with the confusion matrices of the Easy and Complex sets, where few similar types of gestures conflict with each other. One characteristic that is unique to the All gesture set is that the conflicting gestures (a, c, i, k, l and n) operate along the Z axis. We suspect that this is caused by the low resolution in Z axis of the radar, due to having two antennas for elevation angle resolution.

6.2 Comparison with the Related State-of-the-art Classifiers

We compare the performance of Pantomime on Easy, Complex, and All gesture sets against four state-of-the-art point cloud classifiers: two *static* point cloud classifiers: PointNet [40] and PointNet++[41], and three *dynamic* point cloud classifiers: Owoyemi and Hashimoto [35] (O&H), a standard PointNet++ and LSTM architecture [47] and RadHAR [51]. The main difference of PointNet and PointNet++ is that the former does not capture the local structure in point clouds and, consequently, it does not contain set abstraction layers. To ensure a fair comparison with PointNet and PointNet++, we aggregate all frames corresponding to the gesture and feed the aggregated point cloud to them. In this way, the two neural networks operate on the 3D shapes of the whole gestures.

O&H is a hand motion gesture recognition system that transforms the point cloud acquired from Microsoft Kinect into a dense occupancy grid by voxelizing the points. Motion dynamics are captured by constructing a 4D tensor with different time instances of the occupancy grid as the 4^{th} dimension. The tensor is used as the input to a 3D CNN which learns the spatio-temporal features in the data without explicit modeling of gesture dynamics. The O&H architecture comprises 4 convolution layers and 2 fully connected layers followed by an output layer.

Table 5. Comparison with the state of the art. Both Accuracy and AUC are reported in percentages. The best results are denoted in bold typeface.

	Easy		Complex		All	
Model	**Acc.**	**AUC**	**Acc.**	**AUC**	**Acc.**	**AUC**
PointNet	79.7	98.4	82.5	98.7	81.6	99.4
PointNet++	79.7	98.1	84.9	99.0	83.6	99.4
O&H	77.7	96.0	83.2	98.1	79.1	97.9
Std. Arch.	83.3	98.5	88.4	99.5	86.3	99.4
RadHAR	91.6	98.9	94.3	99.6	89.9	99.5
Pantomime (ours)	**96.6**	**99.8**	**95.1**	**99.8**	**95.0**	**99.9**

To increase the amount of training data, O&H uses data augmentation, so that we augmented accordingly our dataset with 0, 10, and 20 cm jitter and report the best results, which occurred when using 10 cm jitter.

The standard PointNet++ and LSTM architecture (Std. Arch.) uses parallel PointNet++ blocks for spatial feature extraction (without the aggregated frames) and each PointNet++ block is connected to two parallel LSTM layers.

RadHAR is a whole body activity recognition system that voxelizes the input point cloud from a mmWave radar sensor but uses a time distributed CNN and Bi-directional LSTM to capture spatio-temporal features. The architecture consists of 3 time-distributed CNNs (convolution layer + convolution layer + maxpooling layer) followed by the Bi-LSTM layer and an output layer. Even though RadHAR does not include data augmentation as part of its architecture, we perform this to be consistent with the other models. Accordingly, we perform data augmentation similar to O&H, as RadHAR is also a voxelization-based approach, and report the best results, which occurred with 10 cm jitter.

Table 5 shows the accuracy and AUC values for the three datasets for the 6 models. Importantly, Pantomime achieves the best performance, exceeding the accuracy of the next best classifier by 5.6%, 0.85%, and 5.7% for Easy, Complex and All gestures, respectively. It is interesting to note that the results of Pantomime are consistent throughout the three gesture sets with just 1.5% difference between the best and the worst results, whereas the accuracy differences of other classifiers fluctuate between 3.5% and 7% (worst and best results, respectively).

Interestingly, although our model has slightly higher accuracy for Easy than Complex gestures, this is not the case for the other models, which performed better on the Complex gestures (but not better than Pantomime in any case). This suggests that even though we selected the Easy gestures to be easy for humans to perform, it does not necessarily imply that those gestures are easy for the classifiers to recognize. In the remainder of this section we evaluate the performance of Pantomime using the All gesture set.

6.3 Effect of the Environment

Generally, in RF sensing systems the environment plays an important role on the accuracy due to noise caused by multipath effects. Table 6 evaluates the effect of the training and testing environment on the set of All gestures. As shown in the table, we use 7 different scenarios of Open and Office data for training and testing. In the first scenario, we train only using the Office data and test on the Open data: in this case, we split the Office dataset in 80% for training and 20% for validation. The validation set consists of unseen users (not in the training set) and the test set is from the Open dataset, which is a completely different data set. In the second column of the table we repeat this experiment but swap both environments. For training with Open+Office data, we use a 60:20:20 user-dependent ratio such that 60% is training data, 20% is validation data and the remaining 20% is used

for testing. Again, the test set consists of data from 8 unseen users in the training and validation sets. The same reasoning applies to the remaining columns of the table.

As expected, training on both environments achieves the highest accuracy irrespective of the testing environment. This indicates that training on both environments leads to better model generalization. Similarly, irrespective of the training environment, testing on Open achieves a higher accuracy than Office since Open has the least effect from environment noise. On average, training and testing on two different environments achieves an accuracy of 90.74%, which represents a 4.24% reduction compared to the average accuracy of training in both environments.

The table further illustrates the results of Restaurant (1680 samples from 4 participants) and Factory (1680 samples from 4 participants). For these two datasets, we use a pre-trained model trained on the data from Open+Office. Pantomime achieves 81% accuracy and 98.84% AUC on Restaurant, and 89% accuracy and 99.79% AUC on Factory. Understandably, performance is lower in these environments than in Open and Office, which we attribute to the fact that both environments are multipath-heavy and the model was not trained on either of these environments. Yet, accuracy in both environments is above 80%.

Table 6. Effect of environment on recognition performance.

Train	Office	Open	Open+Office	Open+Office	Open+Office	Open+Office	Open+Office
Test	Open	Office	Open	Office	Open+Office	Restaurant	Factory
Acc. (%)	93.96	87.51	96.12	93.40	95.01	81.13	89.11
AUC (%)	99.87	99.35	99.94	99.86	99.91	98.84	99.79

6.4 Effect of Multiple People and Occlusion

To study the effect of multiple people, the OPEN environment was used with 4 people within a 2 m radius including the person performing the gestures. Each participant performed 5 trials of each of the 21 gestures. Two people were sitting while working with their laptops and one person was walking around. Three of them were staying in less than 1 m distance from each other, in the field of view of the radar sensor, however not blocking the person conducting the gestures. In the multiple-people setting, we used additional processing to remove the effects in the point cloud from the reflections of additional participants. More specifically, we used a $1 \times 1 \times 2\,m^3$ bounding box to isolate the person conducting the gesture from the rest of the participants. Then, we used the model trained on Open+Office datasets and applied on the data collected from multiple people. Pantomime achieved an average accuracy of 93.33% and an average AUC of 99.95%.

For Through-wall, we obtained measurements from 4 participants, located within 2 m to the radar, and each participant performed 10 trials of each gesture. We used the same model trained on Open+Office, and Pantomime achieved an accuracy of 64.43% and an AUC of 97.24%. We would like to highlight that especially in the through-wall setting we were able to capture the point clouds of all gestures, even the motion patterns were similar to other environments, though with a sparser point cloud. The key difference compared to other environments, which caused the degradation in the classification accuracy, was that the static points of the point cloud that are usually associated with the participant's body are coming off the wall in the Through-wall case.

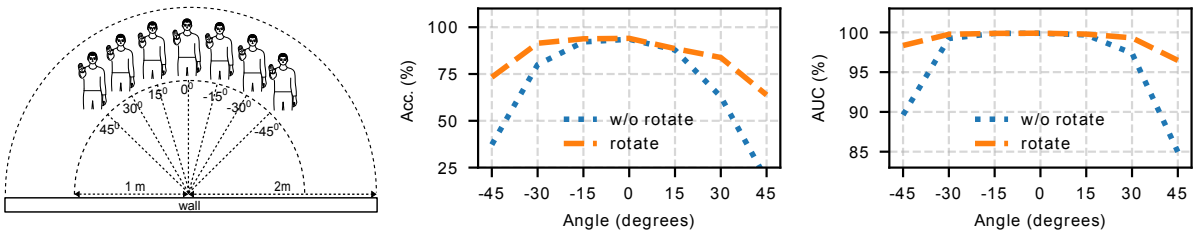
6.5 Effect of the Articulation Angle

RF gesture recognition systems in the literature are usually evaluated with respect to their boresight angle (i.e. 0°, in front of the radar) since this achieves best recognition performance. We consider angles between −45° to +45° in increments of 15° with regard to the boresight angle. We selected this set of angles because the field of

view of the radar ranges between −55° and +55° azimuth angles. Therefore, within ±45° angles it is possible to capture the whole body of the participants within the field of view of the radar. Under this scenario, we collected 2940 gestures (420 samples for each angle) from 8 participants that were used as test data. The results are shown in Figure 9.

As can be observed, the results indicate excellent recognition performance (> 89%) when the user is located within the range of ±15°. Then, recognition accuracy drops to 84% for angles comprised between −30° to +30° and there is a decrease of about 20 % at ±45°. This degradation is mostly caused by self-shadowing of one of the participant's arm, due to the orientation of the body, as it was parallel to the radar at all times. So, for extreme angular values the radar simply cannot fully sense the participant's moving arms.

Another important observation is that accuracy in positive angles is slightly lower than the accuracy recorded in negative angles. Since our single-handed gestures are predominantly right-handed, we conjecture that this differences may be a combination of a calibration error of the radar and the right hand of the participants reaching beyond the +55° field of view while performing gestures.



(a) (b) (c)

Fig. 9. Performance analysis as a function of the angle between the person and the radar. (a) Participant's position (1–2 m) and orientation (parallel to the wall)., (b) Accuracy vs angle. (c) AUC vs angle.. Note: 'rotate' denotes the participant's point cloud is rotated to 0° angle and 'w/o rotate' denotes no rotation of the point cloud.

6.6 Effect of the Articulation Speed

To evaluate the effect of various velocities on the gesture recognition performance, we recorded the set of All gestures in *slow*, *medium*, and *fast* speed from two participants at 0° at a 1.5 m distance. Participants were allowed to perform the gestures according to their perceived pace rather than setting a strict timing on the three different speeds. Under this scenario, we collected 1260 instances of gestures (420 for each speed) that were used as test data. The results are shown in Table ??.

The results show that accuracy is consistently high for medium (94.0%) or fast (92.14%) articulation speed, whereas accuracy decreased by 9% for slow gestures. We note that this drop is attributed to the fact that the CFAR algorithm [44] is sensitive to the speed of the gesture, which produces sparser point clouds at lower articulation speeds.

6.7 Effect of the Articulation Distance

To analyze the impact of the distance of the participant conducting the gesture, we collected further measurements for 3, 4, and 5 m distances at 0°. Since articulation distance is a critical operational parameter, we recorded 1890 instances (630 gestures for each distance) from six participants. Out of these, 1260 gestures (210 instances for each distance) from four participants were used for model training and the remaining gestures from two participants were used for model testing. Note that, unlike in the previous speed and angle related experiments, here we train

Table 7. Effect of articulation speed on recognition performance.

	Slow	Medium	Fast
Acc. (%)	85.00	94.05	92.14
AUC (%)	99.33	99.90	99.68

a new model using the gestures from 1.5 m and the newly collected gestures for other three distances. As the gesture samples from 1.5 m had 8519 instances, data augmentation is performed on the newly collected data to increase the sample size to 5040 gestures and thus ensure balanced splits of the data. Data augmentation was performed according to the approach mentioned in Section 4.3. The results are shown in Table 8.

We compare the results trained from all the distances (denoted as 'All' columns in the table) against training only from 1.5 m distance data, which is the original Pantomime model that outperformed the state-of-the-art classifiers in Section 6.1. We can see that training with samples from all distances is highly beneficial and achieves almost twice the accuracy than when testing our original model (trained only at 1.5 m). The best performance is achieved when gestures are performed at 3 m, corresponding to an accuracy of about 90% which is just a 6% decrease from the accuracy achieved in the original case. As expected, accuracy decreases with increasing distance, with the lowest accuracy of 66.2% achieved at a distance of 5 m. Several factors contribute to this performance degradation, such as changes in chirp configuration affecting the point cloud distribution, increasing sparsity of the point cloud, and smaller body parts being detected because of a smaller radar cross-section, which together degrade the spatial structure of the gesture.

Table 8. Effect of distance on recognition performance. Note: The 'All' columns denote training with gestures performed at 1.5, 3, 4 and 5 m.

Test distance (m)	1.5	3		4		5	
Train distance (m)	1.5	1.5	All	1.5	All	1.5	All
Acc. (%)	95	47.6	89.1	40	78.1	30	66.2
AUC (%)	99.9	95.4	99.6	91.8	98.8	87.8	96.2

In addition to the above results, we note that we also trained three separate models for each of the 3, 4, and 5 m distances (with augmented data for each specific distance). However, the performance of the distance-specific models achieved lower accuracy compared to the model trained on all distances. The main reason is that, although we augmented data for each of these models, the number of training samples we had for each distance was not enough. Given the number of hyperparameters, the distance-specific models overfitted the training data. Nevertheless, when we train a model for all distances, despite differences in their data distribution, their similarities commit to each other's performance, so that the model trained on all distances eventually outperformed the distance-specific models by a large margin.

6.8 Challenging Articulation Combinations

So far we have evaluated three key impact factors of gesture articulation, in terms of angle, speed, and distance, however they have been investigated individually. Since practical usage scenarios usually contain combinations of these factors, we present some results under challenging circumstances with different combinations of these three factors. Such combinations were decided at random. The results are reported in Table 9.

Table 9. Random articulation combinations in terms of speed, angle, and distance.

Distance	Angle	Speed	Accuracy (%)	AUC (%)
3m	0	fast	87.61	99.53
3m	15	fast	64.29	95.95
5m	0	normal	62.86	96.37
4m	-30	slow	46.98	91.46
3m	30	fast	45.24	90.48
2m	45	fast	42.86	89.35
3m	-45	fast	36.19	89.51
3m	-45	normal	31.43	83.41
3m	-45	slow	23.81	71.06
2m	45	normal	21.43	73.87

It is important to note that Pantomime was trained for 0° angle and 'normal' speed; see Section 6.5 and Section 6.6. However, for the 1.5, 3, 4 and 5 m distances, we used the same models trained in Section 6.7 where the 1.5 m distance gestures were infused with data from 2, 3, 4 and 5 m gesture samples. Therefore, it was expected that Pantomime would not achieve as high performance as in the designed operational conditions, i.e. when gestures are performed in front of the sensor and at a normal articulation speed. We can see that, all other factors being equal, large variations in the articulation angle contribute more to degrading recognition performance. Nevertheless,s Pantomime still achieves a good performance for completely unseen test cases, highlighting thus the versatility of our system to handle out-of-domain cases. (We expand a bit more on this out-of-domain notion by investigating unseen gesture classes in the next section.) Ultimately, these experiments should be seen as challenging examples of Pantomime's performance under more extreme operational conditions, for which it was not designed.

6.9 Effect of Unseen Gestures

We have conducted additional experiments to show how Pantomime can handle unseen (aka out-of-domain) gestures. We have redesigned our system's pipeline as shown in Section A.3, to perform *zero-shot learning* based on the latent space of motion gestures.

Figure 10 shows the accuracy of Pantomime (overall accuracy of seen and unseen gestures) as well as the accuracy of the unseen gesture classes as the number of unseen gesture classes increase. It can be noted that both accuracies decrease with increasing number of unseen gesture classes. After adding 4 classes, the unseen gesture class accuracy drops below 50% and the overall system accuracy is around 75%, which could be seen as the "breaking point" of Pantomime; i.e. the system could handle up to three different unseen gestures with competitive accuracy. By way of example, a random classifier would be right in 1/(21+4) = 4% of all cases. Hence, the achieved 75% accuracy is still a remarkably good result. We further note that gesture accuracy and system accuracy are dependent on the selected threshold, Thr. In the current implementation, we selected Thr based on EER, as explained in Section A.3. Then, by adjusting this threshold based on the requirements of the gesture designer, a trade-off between the system and the unseen gesture class accuracy can be reached.

7 DISCUSSION

In this section, we provide a high-level discussion of our results, highlight the main limitations of Pantomime, and propose new ideas for future work.

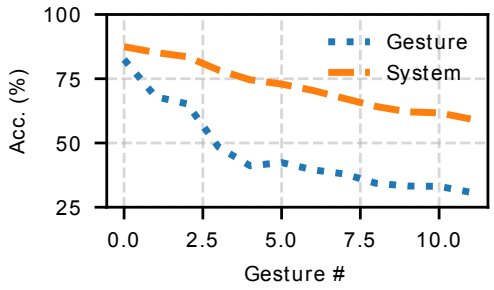


Fig. 10. The accuracy of recognizing an unseen gesture and the overall system accuracy with increasing number of unseen gesture classes.

7.1 Real-time Recognition

Pantomime is capable of recognizing gestures in real-time. We have tested it under two different settings on a single Tesla V100 GPU with 16GB of memory. Pantomime is able to recognize 16 and 10 gestures per second on average using a batch size of 16 and 1 gestures, respectively. (The latter illustrates the case where a single user executes a single gesture in real-time). Note that we consider here the case where a pretrained model performs *both* the preprocessing and classification of incoming raw point clouds.

7.2 Provision of the Radar Configuration

In Appendix A, we discuss how to setup the IWR1443 hardware for the generation of dynamic 3D point clouds for motion gesture recognition. By doing this, we were able to achieve full control over the radar signal and can optimize computational resources. For convenience, we provide our radar configuration files at [36].

7.3 Delimiter Gesture for Getting Pantomime's Attention

In practice, users generally do not inform the system when they start the gestures. This applies to any gesture-driven application, as noted by Ruiz & Li [46]. Even in voice-activated systems like 'Google Assistant', the system has to receive an activation command such as 'OK Google' from the user in order to obtain the attention of the system. Only then, the system records and interprets a user's utterance and executes the intended command. We propose the same approach for Pantomime where the person who wants to interact with the system uses a delimiter gesture to obtain its attention. Since Pantomime can detect a bounding box to single out the interacting user (if there are more people in the sensor's coverage area), it can follow that user and execute their commands for a fixed duration. We note that other gesture recognition systems have followed this approach, such as WiSee [39].

7.4 Limitations and Future Work

We have identified three key areas when it comes to improving our model's performance. In the following we highlight them and propose ways to mitigate their influence.

Sensor granularity. As mentioned in Section 6.1, Pantomime has lower accuracy on gestures performed along the Z axis due to the low elevation angle resolution. This could be solved by placing two radars: one resolving gestures in the azimuthal angle and the other resolving elevational angle. Another related issue is that gestures should not be performed at slow speed, since the resulting point clouds would be very sparse and therefore they would be much more difficult to recognize. However, users are known to operate at their own speed-accuracy tradeoff [64] so they are expected to articulate gestures at medium speed in practice.

Point intensity. While Pantomime currently achieves a recognition accuracy over 95% on average (see Table 5), we plan to include point intensity to improve performance further. The IWR1443 sensor provides intensity values for each point in the point cloud, however if we incorporate this information as an additional spatial feature (i.e., considering each point as {x,y,z,intensity} vector) then recognition accuracy does not improve. Therefore, we should explore other ways of incorporating this information effectively.

Temporal resolution. Using a higher number of frames improves temporal resolution and potentially also recognition accuracy, but it requires more processing power, both during training and at inference time. This is mainly why Pantomime is optimized to classify sparse point clouds with a limited number of frames. For future work, we plan to handle more frames more efficiently.

8 CONCLUSION

We have introduced Pantomime, a novel mid-air gesture recognition system exploiting spatio-temporal properties of mmWave RF signals via 3D sparse point motion clouds. Pantomime is positioned in a unique region of the RF landscape: mid-resolution mid-range high-frequency sensing, which makes it ideal for motion gesture interaction. In designing Pantomime, we followed a holistic approach, starting from selecting a miniaturized mmWave radar, optimizing it for high spatio-temporal resolution for complex motion gestures, and finally introducing a novel deep learning architecture to classify a challenging set of 21 gestures. We tested the robustness of Pantomime in three orthogonal motion directions with remarkable average performance (over 95% accuracy and 99% AUC on average), different environments, speeds, angles and distances. Ultimately, this work opens the door to using commercial radar devices for gesture interaction. Our models and dataset are publicly available at [36].

ACKNOWLEDGMENTS

We thank the anonymous referees for the constructive feedback provided. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813999 and ERANET-COFUND (H2020) CHIST-ERA III, RadioSense. The calculations presented above were performed using computer resources within the Aalto University School of Science "Science-IT" project.

REFERENCES

[1] ABB. 2020. YuMi - Creating an automated future together. (2020). https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi (accessed: 09.11.2020).

[2] Heba Abdelnasser, Moustafa Youssef, and Khaled A Harras. 2015. WiGest: A ubiquitous wifi-based gesture recognition system. In *Proc. INFOCOM*. 1472–1480.

[3] Fadel Adib, Chen-Yu Hsu, Hongzi Mao, Dina Katabi, and Frédo Durand. 2015. Capturing the Human Figure through a Wall. *ACM Trans. Graphics* 34, 6 (2015), 1–13.

[4] Daniel Avrahami, Mitesh Patel, Yusuke Yamaura, and Sven Kratz. 2018. Below the Surface: Unobtrusive Activity Recognition for Work Surfaces Using RF-Radar Sensing. In *Proc. IUI*. 439–451.

[5] A. D. Berenguer, M. C. Oveneke, H. Khalid, M. Alioscha-Perez, A. Bourdoux, and H. Sahli. 2019. GestureVLAD: Combining Unsupervised Features Representation and Spatio-Temporal Aggregation for Doppler-Radar Gesture Recognition. *IEEE Access* 7 (2019), 137122–137135.

[6] Francisco Bernardo, Nicholas Arner, and Paul Batchelor. 2017. O soli mio: exploring millimeter wave radar for musical interaction. In *Proc. NIME*. 283–286.

[7] David J. Brenner, Richard Doll, Dudley T. Goodhead, Eric J. Hall, Charles E. Land, John B. Little, Jay H. Lubin, Dale L. Preston, R. Julian Preston, Jerome S. Puskin, Elaine Ron, Rainer K. Sachs, Jonathan M. Samet, Richard B. Setlow, and Marco Zaider. 2003. Cancer risks attributable to low doses of ionizing radiation: Assessing what we really know. *Proc. Natl. Acad. Sci. U.S.A* 100, 24 (2003), 13761–13766.

[8] Kelly Caine, Selma Šabanovic, and Mary Carter. 2012. The Effect of Monitoring by Cameras and Robots on the Privacy Enhancing Behaviors of Older Adults. In *Proc. HRI*. 343–350.

[9] Gilles Cohen, Mélanie Hilario, Hugo Sax, Stéphane Hugonnet, and Antoine Geissbuhler. 2006. Learning from imbalanced data in surveillance of nosocomial infection. *Artif. Intell. Med.* 37, 1 (2006), 7–18.

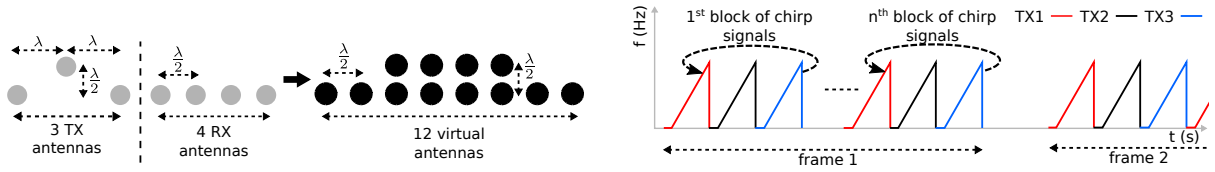
[10] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y Zeevi. 1997. The farthest point strategy for progressive image sampling. *IEEE Trans. on Image Process.* 6, 9 (1997), 1305–1315.
[11] Barrett Ens, Aaron Quigley, Hui-Shyong Yeo, Pourang Irani, Thammathip Piumsomboon, and Mark Billinghurst. 2017. Exploring Mixed-Scale Gesture Interaction. In *SIGGRAPH Asia Posters*. 1–2.
[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. KDD*. 226–231.
[13] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. 2018. Hand PointNet: 3D hand pose estimation using point sets. In *Proc. CVPR*. 8417–8426.
[14] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 2016. Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In *Proc. CVPR*. 3593–3601.
[15] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. 2017. 3D convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *Proc. CVPR*. 1991–2000.
[16] Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič, and Jaka Sodnik. 2014. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors* 14, 2 (2014), 3702–3720.
[17] Nils Y Hammerla and Thomas Plötz. 2015. Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*. 1041–1051.
[18] Otmar Hilliges, Shahram Izadi, Andrew D. Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. 2009. Interactions in the Air: Adding Further Depth to Interactive Tabletops. In *Proc. UIST*. 139–148.
[19] Cherdsak Kingkan, Joshua Owoyemi, and Koichi Hashimoto. 2018. Point Attention Network for Gesture Recognition Using Point Cloud Data. In *Proc. BMVC*. 1–13.
[20] Panayiotis Koutsabasis and Panagiotis Vogiatzidakis. 2019. Empirical Research in Mid-Air Interaction: A Systematic Review. *Int. J. Hum.-Comput. Interact.* 35, 18 (2019), 1747–1768.
[21] KUKA. 2020. Cobots in the industry. (2020). https://www.kuka.com/en-de/future-production/industrie-4-0/industrie-4-0-cobots-in-industry (accessed: 09.11.2020).
[22] Inwoong Lee, Doyoung Kim, Seoungyoon Kang, and Sanghoon Lee. 2017. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In *Proc. CVPR*. 1012–1020.
[23] Luis A. Leiva, Matjaž Kljun, Christian Sandor, and Klen Čopič Pucihar. 2020. The Wearable Radar: Sensing Gestures Through Fabrics. In *Proc. MobileHCI Extended Abstracts*. 1–7.
[24] Hong Li, Wei Yang, Jianxin Wang, Yang Xu, and Liusheng Huang. 2016. WiFinger: talk to your smart devices with finger-grained gesture. In *Proc. UbiComp*. 250–261.
[25] Tianhong Li, Lijie Fan, Mingmin Zhao, Yingcheng Liu, and Dina Katabi. 2019. Making the invisible visible: Action recognition through walls and occlusions. In *Proc. ICCV*. 872–881.
[26] 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 Trans. Graphics* 35, 4 (2016), 1–19.
[27] Roanna Lun and Wenbing Zhao. 2015. A survey of applications and human motion recognition with Microsoft Kinect. *Int. J. Pattern Recognit. Artif. Intell.* 29, 5 (2015), 1555008:1–48.
[28] Yongsen Ma, Gang Zhou, Shuangquan Wang, Hongyang Zhao, and Woosub Jung. 2018. SignFi: Sign language recognition using wifi. *ACM IMWUT* 2, 1 (2018), 1–21.
[29] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2014. Hand gesture recognition with leap motion and Kinect devices. In *Proc. ICIP*. 1565–1569.
[30] Jess McIntosh, Mike Fraser, Paul Worgan, and Asier Marzo. 2017. DeskWave: Desktop Interactions Using Low-Cost Microwave Doppler Arrays. In *Proc. CHI EA*. 1885–1892.
[31] Zhen Meng, Song Fu, Jie Yan, Hongyuan Liang, Anfu Zhou, Shilin Zhu, Huadong Ma, Jianhua Liu, and Ning Yang. 2020. Gait Recognition for Co-existing Multiple People Using Millimeter Wave Sensing. In *Proc. AAAI*. 1–8.
[32] Yuecong Min, Xiujuan Chai, Lei Zhao, and Xilin Chen. 2019. FlickerNet: Adaptive 3D Gesture Recognition from Sparse Point Clouds. In *Proc. BMVC*. 1–13.
[33] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, and Jan Kautz. 2016. Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural network. In *Proc. CVPR*. 4207–4215.
[34] A. E. Omer, G. Shaker, S. Safavi-Naeini, K. Murray, and R. Hughson. 2018. Glucose Levels Detection Using mm-Wave Radar. *IEEE Sens. Lett.* 2, 3 (2018), 1–4.
[35] Joshua Owoyemi and Koichi Hashimoto. 2018. Spatiotemporal learning of dynamic gestures from 3d point cloud data. In *Proc. ICRA*. 1–5.
[36] Sameera Palipana, Dariush Salami, Luis Leiva, and Stephan Sigg. 2020. Pantomime dataset and source. (April 2020). DOI: http://dx.doi.org/10.5281/zenodo.4459969
[37] Joseph Paradiso, Craig Abler, Kai-yuh Hsiao, and Matthew Reynolds. 1997. The Magic Carpet: Physical Sensing for Immersive

Environments. In *Proc. CHI EA*. 277–278.
[38] Pramod Kumar Pisharady and Martin Saerbeck. 2015. Recent methods and databases in vision-based hand gesture recognition: A review. *Comput. Vision Image Understanding* 141 (2015), 152–165.
[39] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home gesture recognition using wireless signals. In *Proc. MobiCom*. 27–38.
[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proc. CVPR*. 652–660.
[41] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017b. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. NeurIPS*. 5099–5108.
[42] Tauhidur Rahman, Alexander T. Adams, Ruth Vinisha Ravichandran, Mi Zhang, Shwetak N. Patel, Julie A. Kientz, and Tanzeem Choudhury. 2015. DoppleSleep: A Contactless Unobtrusive Sleep Sensing System Using Short-Range Doppler Radar. In *Proc. UbiComp*. 39–50.
[43] S. S. Rautaray and A. Agrawal. 2012. Vision based hand gesture recognition for human computer interaction: A survey. *Artif. Intell. Rev.* 43, 1 (2012), 1–54.
[44] Mark A. Richards, James A. Scheer, and William A. Holm. 2010. *Principles of Modern Radar: Basic Principles*. Scitech Publishing.
[45] Frank C Robey, Scott Coutts, Dennis Weikle, Jeffrey C McHarg, and Kevin Cuomo. 2004. MIMO radar theory and experimental results. In *Proc. ACSSC*, Vol. 1. 300–304.
[46] Jaime Ruiz and Yang Li. 2011. DoubleFlip: a motion gesture delimiter for mobile interaction. In *Proc. CHI*. 2717–2720.
[47] D. Salami, S. Palipana, M. Kodali, and S. Sigg. 2020. Motion Pattern Recognition in 4D Point Clouds. In *Int. Works. on Machine Learning for Signal Processing (MLSP)*. 1–6.
[48] C. Sandor and H. Nakamura. 2018. SoliScratch: A Radar Interface for Scratch DJs. In *Proc. ISMAR-Adjunct*. 427–427.
[49] Panneer Selvam Santhalingam, Al Amin Hosain, Ding Zhang, Parth Pathak, Huzefa Rangwala, and Raja Kushalnagar. 2020. Environment-Independent ASL Gesture Recognition Using 60 GHz Millimeter-wave Signals. *ACM IMWUT* 4, 1 (2020), 1–30.
[50] G. Shaker, K. Smith, A. E. Omer, S. Liu, C. Csech, U. Wadhwa, S. Safavi-Naeini, and R. Hughson. 2018. Non-Invasive Monitoring of Glucose Level Changes Utilizing a mm-Wave Radar System. *IJMHCI* 19, 3 (2018), 10–29.
[51] Akash Deep Singh, Sandeep Singh Sandha, Luis Garcia, and Mani Srivastava. 2019. RadHAR: Human Activity Recognition from Point Clouds Generated through a Millimeter-wave Radar. In *Proc. ACM mmNets Workshop*. 51–56.
[52] John A. Stine and David L. Portigal. 2004. *Spectrum 101: An Introduction to Spectrum Management*. Technical Report. MITRE Corporation.
[53] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3D shape recognition. In *Proc. ICCV*. 945–953.
[54] Ivan E. Sutherland. 1963. *Sketchpad: A man-machine graphical communication system*. Technical Report 296. Lincoln Laboratory, MIT.
[55] I. E. Sutherland. 1965. The ultimate display. In *Proc. IFIP Congress*. 506–508.
[56] Klen Čopič Pucihar, Christian Sandor, Matjaž Kljun, Wolfgang Huerst, Alexander Plopski, Takafumi Taketomi, Hirokazu Kato, and Luis A. Leiva. 2019. The Missing Interface: Micro-Gestures on Augmented Objects. In *Proc. CHI EA*. 1–6.
[57] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. 2014. Human action recognition by representing 3d skeletons as points in a lie group. In *Proc. CVPR*. 588–595.
[58] Raghav H Venkatnarayan, Griffin Page, and Muhammad Shahzad. 2018. Multi-user gesture recognition using WiFi. In *Proc. MobiSys*. 401–413.
[59] Aditya Virmani and Muhammad Shahzad. 2017. Position and orientation agnostic gesture recognition using WiFi. In *Proc. MobiSys*. 252–264.
[60] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan. 2011. Vision-based hand-gesture applications. *Commun. ACM* 54, 2 (2011).
[61] 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 *Proc. UIST*. 851–860.
[62] Teng Wei and Xinyu Zhang. 2015. mTrack: High-precision passive tracking using millimeter wave radios. In *Proc. MobiCom*. 117–129.
[63] Hui-Shyong Yeo, Gergely Flamich, Patrick Schrempf, David Harris-Birtill, and Aaron Quigley. 2016. RadarCat: Radar Categorization for Input & Interaction. In *Proc. UIST*. 833–841.
[64] Shumin Zhai, Jing Kong, and Xiangshi Ren. 2004. Speed-Accuracy Tradeoff in Fitts' Law Tasks: On the Equivalency of Actual and Nominal Pointing Precision. *Int. J. Hum.-Comput. Stud.* 61, 6 (2004), 823–856.
[65] Mingmin Zhao, Fadel Adib, and Dina Katabi. 2016. Emotion Recognition Using Wireless Signals. In *Proc. MobiCom*. 95–108.
[66] M. Zhao, T. Li, M. A. Alsheikh, Y. Tian, H. Zhao, A. Torralba, and D. Katabi. 2018. Through-Wall Human Pose Estimation Using Radio Signals. In *Proc. CVPR*. 7356–7365.
[67] Mingmin Zhao, Yingcheng Liu, Aniruddh Raghu, Tianhong Li, Hang Zhao, Antonio Torralba, and Dina Katabi. 2019a. Through-wall human mesh recovery using radio signals. In *Proc. ICCV*. 10113–10122.
[68] Peijun Zhao, Chris Xiaoxuan Lu, Jianan Wang, Changhao Chen, Wei Wang, Niki Trigoni, and Andrew Markham. 2019b. mID: Tracking and identifying people with millimeter wave radar. In *Proc. DCOSS*. 33–40.

A RADAR OVERVIEW AND CONFIGURATION

A.1 Signal processing

As shown in Figure 11(a), the IWR1443 sensor has three Tx antennas, separated by one wavelength (λ), and four Rx antennas, each $\lambda/2$ apart. This configuration yields 12 virtual antennas [45]. During each time slot, the corresponding antenna sends a chirp signal, which is a frequency ramp for the FMCW operation; cf. Figure 11(b).



(a) Virtual antenna array configuration.

(b) Chirp signal transmission. A block of chirps completes one transmission loop of the three Tx antennas.

Fig. 11. Basic configurations of the radar sensor.

Figure 12 illustrates the signal processing pipeline of the radar sensor, starting with analog-to-digital conversion (ADC) and ending with the computed point cloud:

(1) *Range-FFT*: The radar sends a chirp signal and the mixing operation between the transmitted and received chirps produces an intermediate frequency signal. The range of the detected object is linearly proportional to the frequency of such signal, which is computed using the Fast-Fourier Transform (FFT) operation.

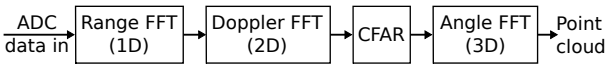


Fig. 12. Radar signal processing of the IWR1443 sensor.

(2) *Doppler-FFT*: At least two time-separated chirps are used to estimate the radial velocity of an object. The phase difference of two chirps at the range-FFT peak is proportional to the radial velocity of the detected object. An FFT operation on the signal range (i.e., a 2D-FFT or a Doppler-FFT) produces a peak at the velocity of the object.

(3) *CFAR*: The summation of the Doppler-FFT matrices creates a pre-detection matrix for each virtual antenna. The constant false alarm rate detection (CFAR) algorithm [44] identifies peaks in the pre-detection matrix that correspond to the detected objects.

(4) *Angle-FFT*: For each object, an FFT of the angle is performed on the corresponding CFAR peaks across multiple Doppler-FFTs. Velocity-induced phase changes are Doppler-corrected before computing the angle-FFT.

A.2 Radar Configuration

We configure the radar's chirp and frame parameters to achieve a fixed frame rate of 30 fps, high range and velocity resolutions, a maximum velocity of 7 m/s and a maximum range of 5m. A chirp is essentially a linear frequency ramp defined by: starting frequency, slope, and duration. A frame comprises of a sequence of chirps and is determined by the frame periodicity T, the number of chirps in a frame (N_c) and the frame duty cycle. Figure 11(b) shows the relationship between chirps and frames.

We configure the chirp duration (T_c) based on the maximum velocity as follows:

$$v_{\max} = \frac{\lambda}{4T} \tag{9}$$

where λ is the wavelength of the radar signal and $T = T_{\text{idle}} + T_c$. The predefined velocity resolution dictates the number of chirps in a frame according to

$$v_{\text{res}} = \frac{\lambda}{2N_c T_c}. \tag{10}$$

The maximum range is related to the chirp slope (s) by

$$R_{\max} = \left(\frac{0.9r}{2}\right)\frac{c}{s} \tag{11}$$

where r is the sampling rate in the analog-to-digital conversion and c is the speed of light. Finally, the range resolution (R_{res}) is determines

$$R_{\text{res}} = \frac{c}{2B} \tag{12}$$

where $B = \frac{r}{sN_s}$ is the bandwidth (B) of the signal and N_s is the number of samples per chirp.

As evidenced from the equations above, parameters such as T_c, N_c, s, r and N_s are inter-related and thus require several iterations with trial and error to configure them properly in order to achieve the required performance in v_{max}, v_{res}, R_{max} and R_{res}.

A.3 Unseen Gestures Work Flow

Figure 13 shows how Pantomime's class scores are used to determine if a gesture is known to the system or not. First, we compute the class scores (s_c, the output of the last layer of Pantomime without applying the argmax operation). If the maximum class score is less than a threshold (Thr), we assume the gesture belongs to an unseen class. If the maximum class score is higher than Thr, we use the label with the highest class score (after the argmax operation) as the predicted label which is the prediction of Pantomime. If the maximum class score is lower than Thr, we extract the output of the last latent feature vector of Pantomime (with dimension 256) and perform dimensionality reduction to 3 dimensions using the TSNE algorithm. We then use k-means clustering to cluster the unseen gestures. We experiment with an increasing number of unseen gestures, up to 11 different gestures classes. We set the number of clusters to be equal to the number of unseen classes in each case (1..12).

Selecting the threshold. The threshold, Thr, we used is the Equal Error Rate (EER) of the seen gestures from Open+Office. We determine Thr in which the false acceptance rate (the ratio of the total number of maximum class scores above Thr with correct prediction to the total number of maximum class scores above Thr with incorrect prediction) is equal to the false rejection rate (the ratio of the total number of maximum class scores with correct prediction below the threshold to the total number of maximum class scores with incorrect prediction below the threshold). In our experiments the EER is equal to 7.89, for which the false rejection and false acceptance rates are 20%.

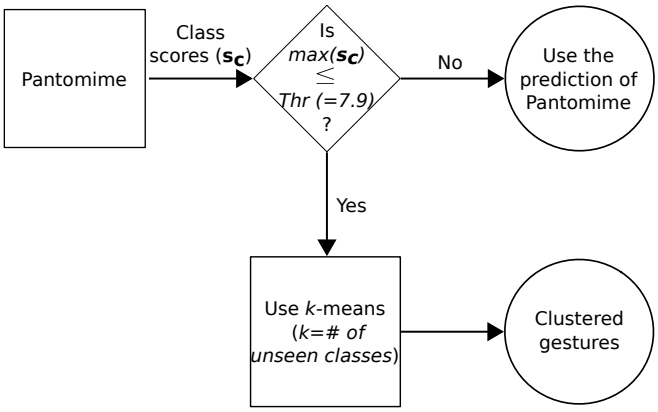


Fig. 13. The decision making process to determine if a gesture is known to the system.