Exploiting mmWave and Deep-Learning Models to Estimate People Count in Urban Scenarios

Girish Vaidya*†, Marco Zuniga*
* *Delft University of Technology, Delft, Netherlands*
† *Amsterdam Institute for Advanced Metropolitan Solutions, Amsterdam, Netherlands*
Email: {g.vaidya, m.a.zunigazamalloa}@tudeflt.nl

***Abstract*—Due to new regulations regarding privacy and the increasing reservations of citizens about surveillance systems, cities are exploring privacy-friendly technologies for crowd monitoring. An alternative gaining significant interest is mmWave radar for people counting applications. However, most studies in mmWave monitoring primarily focus on indoor or outdoor deployments under *ideal conditions*. Working with the municipality of Amsterdam, we propose a method for counting people with mmWave in a *real outdoor scenario*. These realistic scenarios pose two significant difficulties. First, it is hard to estimate the number of people when they walk together because the radar captures the group as a single-point cloud. Second, the dynamic environment adds random points (noise) that may get reflected as false counts. To tackle these challenges, we optimize a state-of-the-art deep learning model to attain *noise filtering* and *cluster disaggregation*. We deploy the radar across a street and measure its performance under real traffic conditions. Our system achieves an accuracy of 96% for *Presence detection* and 69% for *People counting*. Over the long term, the system can monitor the crowd level with an error below 2.5%. Furthermore, considering the needs of our application, we reduce the complexity of the original SoA model by 93%. Our results indicate that mmWave is a promising solution for privacy-friendly people counting in urban scenarios and exposes unique challenges that have not been reported or considered in the SoA.**

I. Introduction

A. Motivation

Need for people counting: We carry out this work together with the municipality of Amsterdam. The city requires an estimate of 'People Count' to meet various objectives. Depending on how crowded an area is, the city decides to deploy additional officers or to regulate the traffic movement. Similarly, the department governing the parks in the city is interested to know which areas and playground equipment are used more often by the citizens. This helps them better plan their future investments. The sports department likewise is interested to know the occupancy of open courts for badminton and soccer. Frequently, these fields are reserved by sports clubs but are not utilized. Thus, while the records portray that the sporting infrastructure is insufficient, the ground reality indicates it is underutilized. These three examples derived from our interaction with the city officials reinforce the need for people counting in urban infrastructure. However, the technological solutions adopted by the city must meet privacy mandates dictated by the legal framework and citizen concerns.

Mandate for privacy: Through various manifestos, the city declares the ethical guidelines governing the implementation of a technology. The TADA values -a manifesto containing shared values for a responsible digital city, states that it is important to use technology and data for the benefit of residents without having a negative impact on the privacy and freedom of citizens [1]. On a wider scale, across Europe, the General Data Protection Regulations (GDPR) provide a legal framework for sensing, communication, usage and storage of personal data. These new regulations state that personal data shall be adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed ('data minimization'). Thus, the privacy of a citizen is given utmost importance in the framework of urban governance. More significantly, beyond the legal mandate, citizens are becoming increasingly aware of their right to privacy and are asserting their reservations against any technology that may invade their personal space. For example, the city of Amsterdam deployed multiple cameras in the past, but now it is performing a pilot to deploy shutters to cover the cameras due to citizen demands. Furthermore, two universities were compelled to withdraw the installation of crowd-counting cameras due to protests from students [2], [3], emphasizing that these devices have not been able to invoke sufficient trust in people. Hence, a new legal framework and civic awareness are driving the sensing of urban parameters to be more privacy-friendly.

Ground reality: Despite the aforementioned guidelines, crowd counting cameras are the most adopted solution for people counting. Several products from companies such as Hikvision [4] and Xovis [5] are commercially available for people counting. Such cameras capture the images, identify people through algorithms, count them and communicate this count. However, besides being considered intrusive devices, the cameras need to be configured through various software updates and maintenance cycles. An incorrect configuration could render the privacy protection inadequate. Thus, while there is a strong requirement for people counting, simultaneously there is an expectation to perform it in a privacy-friendly manner. This has led the city to explore alternative technological solutions. One such alternative is *data-minimization*, i.e., limit as much as possible the collection of personal data. *Nothing can go wrong with the data that is not collected: it cannot be abused, it cannot be misused, or get leaked accidentally* [6]. For crowd monitoring, this means that the sensor should not be able to gather personalized data in the first place, which indicates that cameras cannot be used.

979-8-3503-7025-6/24/$31.00 ©2024 IEEE
DOI 10.1109/IoTDI61053.2024.00011

B. Millimeter Wave (mmWave) radar for people counting

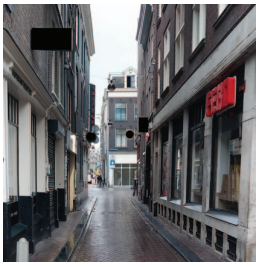
Millimeter wave (mmWave) radars have the innate advantage that they sense objects through a cluster of points, called 'point clouds', thus minimizing the exposure of personal data. Through radar parameter settings, the density of point clouds can be configured such that it indicates the people count but the sparsity makes it difficult to identify the individuals. Thus, the risk of leakage of personal data during a security breach is minimized. Also, they offer an added advantage: adverse weather conditions such as rain or fog have little impact on their performance [7]. This makes mmWave a promising candidate for privacy aware people counting.

In recent years, there has been an increased interest to explore the usage of these radars across diverse applications such as gesture sensing [8], [9], gait estimation [10] and people counting [11]. However, most of the work focuses on indoor scenarios under controlled conditions. Outdoor deployments for people counting pose two significant difficulties. Firstly, it is non-trivial to estimate the correct count of people when they walk together in a group because the radar captures the group as a *single* point cloud. Secondly, the dynamic environment adds random points (noise) that may get reflected as false counts. Few studies use mmWave sensor for outdoor people counting. In our earlier work, we discuss the deployment of a mmWave radar in a university campus for people counting during three months [12]. However, that study does not identify the number of people in a single cluster, leading to potentially severe undercounting. Another study proposes a method to estimate the number of people in a single cluster by extracting features from the point cloud and using a statistical method for classification [13]. However, the implementation uses raw radar data (radar cube) as input which is much larger in size compared to a point cloud. Further, the evaluation is done under controlled conditions.

In this work, we attempt to overcome these limitations by proposing a two-step approach consisting of *Cluster creation*, i.e., creating clusters of point clouds that are free of noise, and *Cluster identification*, i.e., identifying the people count within each cluster. We assess the performance of our implementation under real-life conditions through a rigorous evaluation. In particular, the main contributions of our work are:

- We implement an optimized method for point cloud data that effectively filters the noise and reduces false counts (Section IV). The efficacy of the noise rejection is shown through a high accuracy of presence detection.
- We optimize a state-of-the-art neural network to 7% of its original size and apply it for cluster identification (Section V). The light-weight network model classifies a single cluster to predict the count of people.
- We perform a rigorous evaluation to benchmark the performance of our methodology (Section VII). The results show that our network model can correctly predict the number of people in a single group with 83% accuracy.
- We evaluate the system with real traffic data by setting a deployment across a street (Section VIII). Our system can accurately identify 'Presence detection' with 96% accuracy, and 'People count' with 69% accuracy.

(a) Crowd counting camera mounted on one of the streets.



(b) A typical pavement observed by a crowd counting camera.

Fig. 1: Crowd counting camera deployment.

Sr No	Parameter	Target Specification
1	Range	10 m
2	Area	100 sq m
3	Traffic	Slow moving
4	Lighting Conditions	Should operate under varying lighting conditions
5	Responsiveness	Few minutes
6	Power	Through electricity mains
7	Miscellaneous	Continuous mode of operation

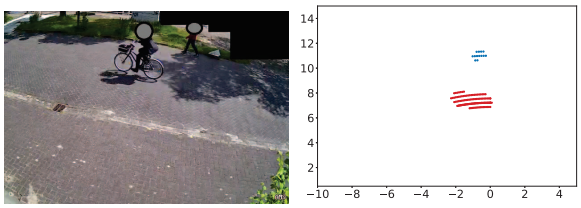
TABLE I: Requirements of a people counting system.

II. Requirements of a people counting system

We discussed three potential applications for people counting in Section I-A. Now, we present our work in the context of the first application, about people counting on streets, with the possibility of generalization for other applications.

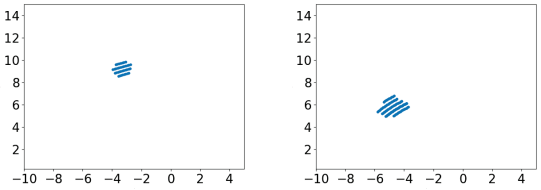
Application scenario: Figure 1 shows a crowd counting camera deployed in the city. Table I summarizes the requirements for such systems. In the current system, each crowd-counting camera captures the scene every minute, computes the count at the edge and relays the outcome to the server. The people count is processed to achieve two objectives. *Firstly*, the data is aggregated and relayed every fifteen minutes to the ground staff responsible for making interventions. The ground staff performs specific actions such as implementing blockades, depending upon the aggregate data. An estimate of how crowded an area is, rather than a precise count, is sufficient to perform these interventions. *Secondly*, the data is used to create various analyses to plan the city infrastructure. For example, the city plans to increase public transport in different areas based on an estimate of people visiting those areas. For the system to be trustworthy, it is imperative that the sensor is reliable and agnostic to ambient lighting.

Table I summarizes the requirements for a people counting system. We evaluate the accuracy of our system under these requirements. We use the data provided by the camera as ground truth. Our results show that if the count is aggregated over time, the system provides better accuracy.



(a) Sample camera picture (b) Equivalent point cloud

Fig. 2: The mmWave point cloud representation shows clusters for a person and a bicycle.



(a) Point cloud for one person. (b) Point cloud for five people.

Fig. 3: The shape and size of a point cloud varies with the number of people and their relative position from the radar.

III. System Background and Insights

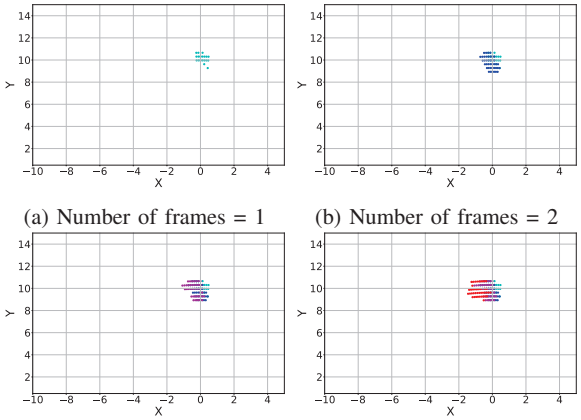
A. mmWave radar sensor

We use the IWR6843-ISK [14], a commercially available mmWave radar evaluation board from Texas Instruments for our application. The radar works in the frequency range of 60 GHz - 64 GHz and belongs to a special class of mmWave technology known as Frequency Modulated Continuous Wave (FMCW) radars. A FMCW radar emits linearly changing electromagnetic signals called chirps that are reflected by the objects in their path. The radar system compares the reflected chirps with the transmitted chirps, to determine the range, velocity and angle of the objects [15]. The object is represented as a cluster of points known as *point cloud*. Figure 2 shows a scene captured by a camera and the corresponding point cloud created by the mmWave radar. The comparison demonstrates the privacy-friendly property of mmWave sensing. In the context of our application, each point cloud may represent a single person or multiple people. Each point in the point cloud is characterized by its position (range and azimuth angle), velocity and signal strength. The range and azimuth are translated from Polar coordinates to the Cartesian space. We use the velocity information to filter and further process only those points that represent moving objects.

The radar encapsulates the point cloud information within a frame together with header information such as frame number and checksum. Each frame captures an instant of time.

B. Insights from the point cloud

From Figure 2, we see that distinct clusters are formed by objects or people far from one another. Thus, people counting algorithms first perform clustering to identify the number of people, but this assumes that every cluster contains only one person [11], [13], [16]. We follow a similar approach but with an improved clustering implementation, as shown in Section IV. The next step is to identify the number of people in each cluster. Figure 3 plots the point clouds for a single person and for five people. We observe that the shapes and sizes of the two clusters are different. This is because the contour of the point cloud depends upon the radar cross-section offered by the object. Thus, the point clouds also depend upon the relative position of the objects from the radar. Further, the signal strength and velocity of each point inside a cluster provide additional information about the object. For example, the signal strength of points reflected from a metal object such as a vehicle will be higher than those reflected by a non-metallic object. These point cloud properties are exploited to derive the number of people in each cluster. We discuss our Cluster identification methodology in Section V.



(a) Number of frames = 1 (b) Number of frames = 2

(c) Number of frames = 3 (d) Number of frames = 5

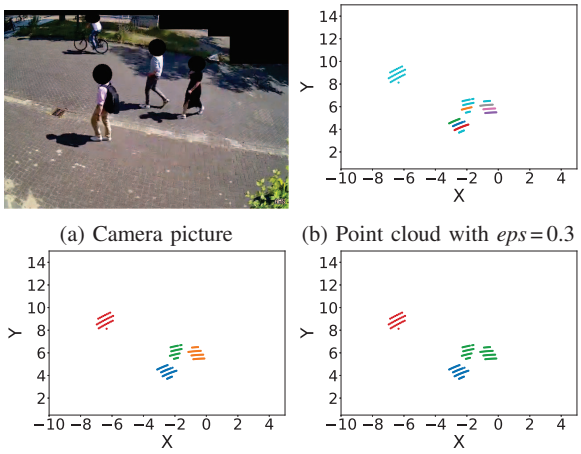
Fig. 4: Multiframe representation of point clouds. Each figure captures a different number of *consecutive* frames.

MF	1	2	3	5
Average cluster size	70.80	134.15	198.15	324.87

TABLE II: Average cluster size for different values of MF.

Multiframe Factor: The mmWave point cloud is sparse in nature providing little information in a single frame. Many gesture recognition algorithms, such as PointLSTM [17] and Pantomime [8], combine multiple consecutive frames to derive more meaningful information. This inspires us to process multiple successive frames as well. We set our system to capture three frames per second. The number of successive frames, combined for processing, is denoted by the term *Multiframe factor, MF*. Figure 4 shows a scene represented by a different number of successive frames. For example, Figure 4c plots three successive frames, in this case, we say that MF=3. Table II tabulates the average size of clusters across different MF values for our training data, reaffirming the increase in information as MF increases.

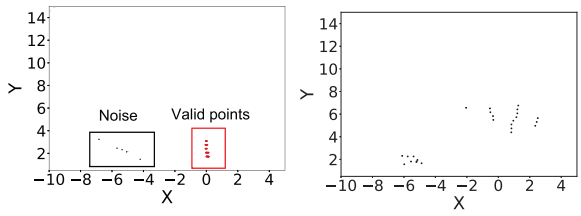
Processing consecutive frames provides multiple advantages. Firstly, if one frame is impacted by noise, the information can be retrieved from points in other frames. Secondly, if a person is occluded in one frame, she may be visible in



(a) Camera picture (b) Point cloud with *eps* = 0.3

(c) Point cloud with *eps* = 0.45 (d) Point cloud with *eps* = 0.6

Fig. 5: Impact of *eps* on clustering. Different colors indicate separate clusters. An optimum value of *eps* (0.45) helps create accurate clusters.



(a) Noise filtering in the presence of valid points. (b) Noise filtering for scattered noise.

Fig. 6: Noise filtering through DBSCAN.

another. Thirdly, the sequence of frames allows for inferring the direction of movement (if desired). Thus, besides using the position, velocity, and signal strength of points as inputs, our model will consider the sequence number as an additional point cloud feature.

Our approach processes multiple frames but we need to derive the optimum MF value. It may seem that a greater MF would be advantageous, but there are tradeoffs during Cluster Creation and Cluster Identification. We investigate these tradeoffs in Sections IV and V.

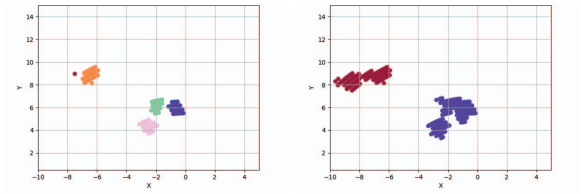
IV. Cluster Creation

Before estimating the number of people in the scene, we need to identify and isolate the clusters of points. The clustering algorithm considers only the spatial coordinates of the points and ignores the SNR and velocity. Across the literature, the preferred clustering method for radar point clouds is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [8], [18]. DBSCAN has several desirable properties [19]: (a) It is a density-based algorithm, effective for clustering points formed by people. (b) The number of clusters does need to be known beforehand. It is capable of creating a variable number of clusters, each with possibly a different number of points. (c) It performs noise rejection through outlier removal. The work in [20] combines multiple frames and implements DBSCAN on the merged point clouds for better noise filtering in an automatic driving application. We follow a similar approach in our application.

Optimization of parameters for noise rejection: The presence of scattered noise in the frame may lead to incorrectly counting a person when there is none. During data collection, we observed that any frame representing a person had a minimum of twenty points. Thus, we perform clustering only if the number of points present in a frame is greater than twenty. If less, the points are rejected as noise. Next we carefully set the two important parameters of DBSCAN, viz., *min_samples* and *eps* for optimizing the noise rejection and cluster formation. The point cloud resolution corresponding to our radar configuration is 0.37 m. This implies that the points within the same cluster may be as far as 0.37 m from one another. If the value is much larger than 0.37, points corresponding to different people may combine together into a single cluster or points corresponding to noisy objects may not get filtered. Thus, we set the *eps* value to 0.45. By deriving the *eps* based on the radar configuration, we arrive at an optimum value that prevents noise being labeled as valid points. Figure 5 shows the impact of different values of *eps* on cluster formation. Next, we choose *min_samples* to ten. Thus, even if a frame has more than twenty points, unless ten of them are together, they are rejected as noise. This helps reject scattered noise points. Figure 6a shows the filtering of noise in the presence of valid points. The algorithm marks the points between X= -8 and X= -4 as noise while putting the points around X= 0 into a single valid cluster. In Figure 6b all the points are marked as noise demonstrating an example of scattered noise rejection. One of the ways to measure the effectiveness of noise rejection is to check whether frames with no people have been correctly recognized with a zero count. We evaluate this metric in Section VIII-B.

MultiFrame factor tradeoff for Clustering: We have discussed in Section III-B that increasing the multiframe factor (MF) increases the number of points processed simultaneously. This may result in smaller separation between clusters, causing the smaller distinct clusters to merge together and form a single large cluster. Figure 7 compares the cluster formation for MF= 1 and MF= 3 for the same scene. Distinct clusters are indicated by different colours in the figure. We observe that MF= 1 leads to four distinct clusters. However, in case of MF= 3, three of the small clusters merge together to form one single cluster. In Section VII we show that the identification accuracy is higher when the cluster size is smaller. This implies that there is a tradeoff in choosing the value of MF, a higher value contributes to dense more meaningful frames, however also leads to merging of distinct clusters that may adversely impact accuracy.

At the output of Clustering stage, we have marked the points in the point cloud into distinct clusters, each representing one or more people. In the next section, we describe the methodology to count the number of people in each cluster.



(a) Four clusters with MF = 1 (b) Two clusters with MF = 3

Fig. 7: Impact of MF on cluster formation. Three small clusters merged into a single cluster when MF changes from 1 to 3.

V. Cluster Identification

After the clusters are formed, the next step is to identify the number of people in each cluster. We study the state of the art to identify a suitable neural network architecture that can process the point clouds and provide the corresponding people count. We approach this as a multi-class classification problem. The network takes the points in a cluster as input and classifies it into one of the following six classes: 1-person, 2-people, 3-people, 4-people, 5-people and bicycle. There are two underlying assumptions. Firstly, we assume that the noise filtering has happened in the Cluster creation stage (refer to Section IV). Thus, during cluster identification, the point clouds represent only valid classes. Secondly, we limit the maximum number of people in a single cluster to five. The studies related to the walking behavior of pedestrian social groups confirm that the group size is limited to less than five in 99% of the cases [22], [23]. Further, as the number of people in a single cluster increases, the number of possibilities in which they can form the cluster increases exponentially. For example, a group of four people can walk in a single row of four people, in two rows of two, in two rows of one and three, or in three rows with one of the rows having two people walking together. Due to the increasing combinations, the amount of data needed to train the neural network for a larger cluster increases significantly, making it harder to obtain.

A. State-of-the-art

Early works on image classification such as Subvolume and MVCNN focus on Convolutional Neural Networks (CNNs) [24], [25]. These approaches have been demonstrated on images and dense point clouds generated through Lidars for objects such as bathtubs, sofas, and beds. However, the computational cost for CNN-based approaches is high [21]. Texas Instruments itself provides a people counting demo under the Industrial toolbox [26]. In this approach, the moving targets are continuously tracked until they leave the scene. However, recent studies have shown that the approach fails to predict the count accurately when people are walking in close groups [13]. That same study [13] proposes a method of counting people in a group by extracting features from the radar cube and using a statistical method for classification. A *radar cube* is a raw representation of the data captured by the radar and is significantly bigger compared to point clouds. As a result, the methods based on raw radar cubes have higher memory and processing demands, potentially requiring complex algorithms and expensive hardware. The memory footprint for a single point cloud in our case is approximately 6 KB whereas the size of a radar cube is in the range of 500 KB [27]. Further, a statistical classifier must define and capture all the features precisely. On the other hand, learning-based models have the adaptability to learn the most relevant features on their own. We thus prefer a learning-based approach with point clouds as input.

Blocks	PointNet	Our Implementation
Input Transform & Feature Transform	Spatial Transformer	Block not implemented
Segmentation Network	Local and global feature aggregation followed by mlp	Block not implemented
Information aggregation stage	mlp (64, 128, 1024)	mlp (64, 128, 512)
Class probability prediction	mlp (512, 256, k)	mlp (256, 128, 6)

TABLE III: Comparison of PointNet with our implementation.

PointNet [21] proposes a lightweight deep learning-based architecture for processing point clouds. The number of parameters in PointNet is 3.5 M as compared to 16.6 M for Subvolume and 60 M for MVCNN [21]. PointNet extracts the features from a point cloud and uses them for multi-class classification as well as segmentation. PointNet classifies Lidar point clouds from sixteen object categories such as bag, car and lamp with 83.17% accuracy. Some recent works, such as PointNet++ [28], Pantomime [8] and Tesla Rapture [9], further extend this line of work for classifying gestures.

PointNet++ applies the PointNet approach recursively on nested input partitions and thus captures local features at different contextual levels. PointLSTM uses a Recurrent Neural Network (RNN) together with LSTM to capture the temporal evolution of frames. Similarly, Pantomime uses the PointNet++ architecture with LSTM to exploit the temporal and spatial correlation of frames. These architectures are shown to provide better performance for a structured sequence of frames. In the case of gestures, the point clouds evolve in a structured manner over the sequence of successive frames. In our case, the sequence of frames may not lead to a well-defined pattern of point clouds. A person may follow different paths causing varying trails of point clouds. This makes it difficult to exploit the temporal evolution. Thus, we prefer the simple architecture of PointNet for our application. Next, we discuss the PointNet architecture and its adaptation to our application.

B. PointNet architecture and model adaptation

Figure 8 shows the block diagram of the PointNet architecture. The input to the neural network is a $n \times m$ datastructure, where n indicates the number of points and m indicates the number of channels. In our case, n represents the number of points across all the aggregated frames and depends upon the MF; and m represents the number of features determined from the point clouds. We evaluate the performance for four different values of n and three different combinations of m. We discuss the choice for n and m in Section VII.

Input and Feature transforms: PointNet utilizes input data dependent spatial transformer in two stages represented

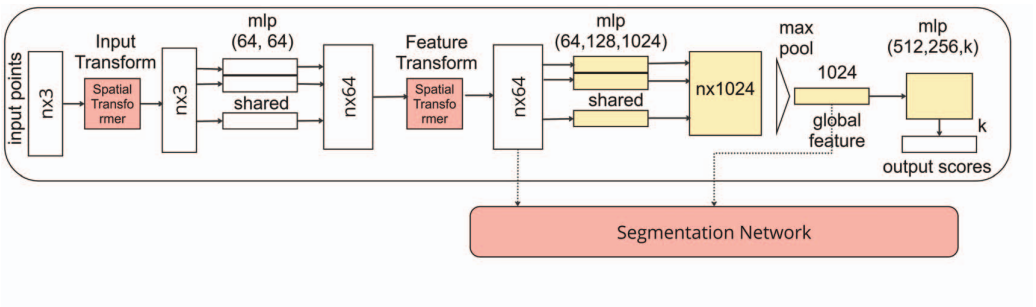


Fig. 8: PointNet architecture [21]. Blocks that we deleted for our implementation are indicated by red and blocks that we modified are indicated by yellow.

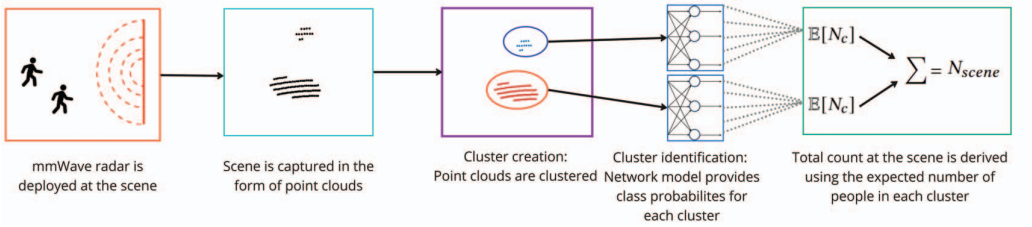


Fig. 9: Pipeline explaining different steps for computing people count at a scene.

as input and feature transforms. The spatial transformers are typically used to transform the inputs in a standard form before the neural network processes them. It normalizes the input and achieves invariance against transformations such as translation and scaling. In our application, the size and shape of the point cloud with respect to its distance from the radar provides significant information about the count of people. Thus, we have decided against using the spatial transformer. Even in the case of PointNet, the improvement in performance through the spatial transformer is not significant (89.2% Vs. 87.1%) [21]. Eliminating the spatial transformer provides an additional advantage of significantly reduced model size, as discussed in Section VII.

Information aggregation stage: The MultiLayer Perceptron (mlp) with a configuration of (64,128,1024) and the subsequent max pooling stage performs information aggregation and extracts the global features of the point cloud. The max pooling function acts as a symmetric function for the unordered set of points, i.e., it extracts the features independent of the order of points. The output of this stage is a 1024-dimensional feature vector. In our case, the mmWave data is sparse. Thus, we reduce the mlp size to (64,128,512) effectively reducing the feature size to half.

Class probability prediction: The next stage is also a fully connected mlp network with a (512,256,k) configuration, where k indicates the number of output classes. In case of PointNet, k corresponds to sixteen. In our implementation, we perform a six-class classification (k=6). The mlp stage is accordingly resized to (256,128,6). The last stage uses a log-softmax function that is used to derive the class probabilities.

Segmentation network: The PointNet architecture combines local and global features to achieve segmentation. However, our application only requires classification. Thus, we ignore the local features and exclude the segmentation network.

Table III summarizes the changes done to PointNet to suit our application. We eliminate blocks that are not relevant to our application and scale down the configuration of the remaining blocks. This results in a lower memory footprint and computational requirement. We discuss the impact of our model adaptations in Section VII-C.

C. Computation of people count

Having discussed the two steps, viz. Cluster creation and Cluster identification, we now explain the steps for computing the count of people at a scene.

Firstly, using the class probabilities from the Cluster identification stage, we compute the expected number of people in a cluster using

$$\mathbb{E}[N_c] = \sum_{i=1}^{i=6} \mathbb{P}_i \times N_i. \tag{1}$$

In this equation, N_c represents the number of people in the cluster, $\mathbb{P}_i$ represents the classification probability of the cluster being classified as N_i class. For $i = 1 \ldots 5$, the value of i indicates the corresponding number of people. For $i = 6$ (bicycle class), we use a count of one, i.e. $N_6 = 1$.

The total number of people in a scene (N_{scene}) is derived by adding the expected count from all the clusters present in the scene:

$$N_{scene} = \sum_{j=1}^{j=n} \mathbb{E}[N_c]. \tag{2}$$

where n represents the total number of clusters after the cluster creation step. Figure 9 captures the step-by-step process of the entire methodology.

Parameters	Designed values
Maximum range (m)	34.306
Maximum velocity (m/s)	7.969
Chirp Time (us)	36.411
Range resolution (m)	0.343
Velocity resolution (m/s)	0.125

TABLE IV: Radar configuration parameters.

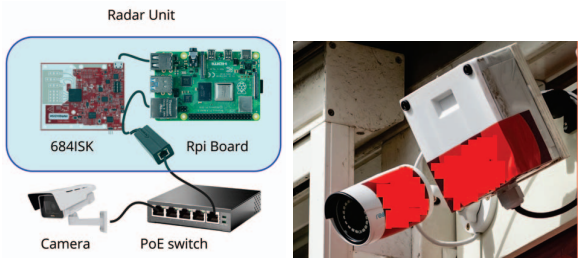
VI. Setup and Data preparation

A. Hardware Setup

We choose the IWR6843ISK [14], an off-the-shelf evaluation board from Texas Instruments (TI) as a radar sensor for our implementation. The board has a IWR6843 radar chip. This is a single chip solution from TI with 4 receivers and 3 transmitters, thus providing a fine angular resolution. It has an onchip ARM CPU and radar hardware accelerator, enabling complex computations in hardware. This provides us with the opportunity to perform machine learning at the edge during future enhancements.

Figure 10a shows the hardware connection setup. A Raspberry Pi (RPi) connected to the radar board, receives the point clouds and writes them into an output file every 1000 frames. The RPi is powered through Power over Ethernet (PoE) and in turn powers the mmWave sensor through USB. To protect the hardware from various weather conditions (rain, fog, wind), we place the RPi, mmWave sensor and the PoE splitter inside a radome (case). For the radome, we use the design suggested in our earlier work [12] because it is low cost and resilient to different types of weather. A camera is installed next to the mmWave radar to capture the ground truth. The time on the camera and Rpi are synchronized through the Network Time Protocol (NTP). Figure 10b shows the picture of the actual installation. The setup is mounted at a height of 3.4 m and looks into a street that is 12 m wide. The street is generally used by pedestrians and bicycles with a very little presence of motor vehicles, making it suitable for our crowd monitoring application. The setup is mounted such that generally the traffic moves sideways across it. Further, the mmWave sensor and the camera are tilted downwards and towards left with an azimuth angle of 20 degrees and an elevation angle of 30 degrees. Figure 5a shows a typical picture of the street captured by the camera with a visual perspective of the azimuth and elevation tilt.

Table IV lists the important radar parameters. To cover the road's width of 12 m at a range of 10 m (i.e., to ensure that the coordinate (-10, 12) is covered by the radar installed at (0, 0)), the radar must cover a diagonal distance of 16 m. Through experiments, we found out that our initial configuration achieved only half of the expected range. Our conjecture is that the radome and environmental parameters attenuate the signals more than what is reported in the SoA. This highlights the challenges for outdoor deployment with a protective enclosure. Thus, we re-configured the radome for a 34.3 m range and limited the velocity to 8 m/s considering the slow moving traffic. With these values, the range and the velocity resolutions are set to 0.343 m and 0.125 m/s, respectively.



(a) Block diagram. (b) Actual installation.

Fig. 10: Radar setup for capturing the point clouds.

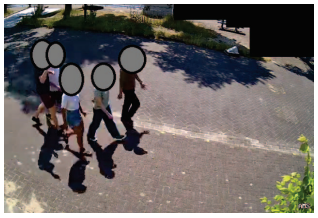


Fig. 11: Volunteers participating in the data collection.

Class	Sample size
1	4491
2	4958
3	3885
4	2924
5	3015
Bicycles	829

TABLE V: Sample size across different classes.

B. Data collection

We signed up a group of five volunteers for training our neural network model. The group consists of a mix of two females and three males, all in an age group between 21 years and 30 years and heights between 150 cm and 175 cm. The volunteers walked in different group sizes, starting with individual walks and then in groups of two, three, four and five. Further, for group sizes larger than two, the volunteers walked in different formations. For example, for a group size of four, they walked in formations of one and three, two and two as well as all four in a row. They walked at different distances, from the near-end to the far-end of the road, covering the entire field of view of the radar. The walks were recorded through the camera and the corresponding mmWave point clouds were captured. We used the video data to annotate the radar frames into six different classes indicating the count of people in the cluster. We created a separate class for bicycles since the point cloud created by a bicycle is distinct from that of a group of people. Figure 11 shows a frame from the video recorded during training. Table V provides the number of radar frames captured for each class. Altogether more than 20000 frames were captured during seven hours of walk by the volunteers. From the table, we also see that the sample size varies across different classes. To minimize the impact of class imbalance we perform data augmentation making to attain 5000 point cloud samples for each class. To generate the augmented data, we sequentially process the collected point clouds and add Gaussian noise with zero-mean to the x and y values. The variance of the x and y values in the point cloud is computed and 20% of that range is set as variance for the noise distribution.

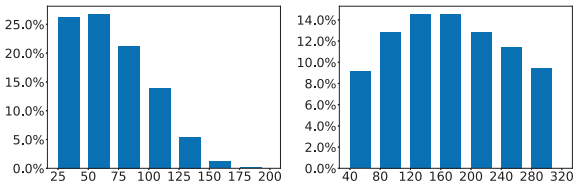
The point clouds captured by the radar as well as the augmented data have varying number of points. However,

MF	Input size n	Percentage of clusters within the input size
1	128	95
2	256	95
3	512	99
5	1024	100

TABLE VI: Input size to neural network for different values of MF. The input sizes have been chosen such that the majority of the clusters have size less than the defined input size.

Feature size m	Features
3	Position (x,y), Sequence
4	Position (x,y), Sequence, SNR
5	Position (x,y), Sequence, SNR, Velocity

TABLE VII: Combination of features considered for different values of m.



(a) Histogram with MF = 1 (b) Histogram with MF = 3

Fig. 12: Histograms of cluster size.

the neural network model expects a uniform input. Next, we describe the data preprocessing steps undertaken to achieve uniformity before it is given to the network model for classification.

C. Data preprocessing

The input data size to the network model depends on the multiframe factor MF and the combination of features. We evaluate the performance with four different values of MF listed in Table VI and three different combinations of features listed in Table VII. We are interested only in points corresponding to moving objects. Hence **first**, we filter points with velocity greater than zero and combine them from successive frames depending on the MF. For example, if MF= 3, the points from each frame are combined together with those from the next two successive frames to form a larger point cloud. However, while combining we add a new property, viz. *sequence*, to keep track of the order of original frames. Figure 12 plots the histograms of cluster sizes for MF=1 and MF=3. We see that the sizes of the point clouds vary significantly for both values of MF. Thus, **second**, we compare the resulting number of points in the combined point cloud with the target input size listed in Table VI. If the resulting point cloud size is smaller, we pad the point cloud with supplementary points through Agglomerative clustering [29]. Agglomerative clustering creates the specified number of centroids depending upon the location of input points and has been used for upsampling in the literature [8]. For example, if MF=2 and the resulting point cloud size is 200, then 56 supplementary points are added. In this case, 56 centroids are created through Agglomerative clustering based on the existing 200 points, and the coordinates of these centroids are considered as supplementary points. In case the resulting point cloud size is larger, we remove the points with the lowest SNR to meet the target input size. We prefer to perform padding (add points) as opposed to reduction, and we set accordingly the value of n to achieve that. For example, referring to Table VI, we see that for MF=3 only 1% of the clusters undergo reduction. **Third,** we select the corresponding features based on the parameter m. Table VII lists the selected combination of features for different values of m. We always use the coordinates and sequence values as features since they provide information regarding shape and size. We observed that the velocity information is noisy even when a single person is walking. This may be due to the fact that the trunk of a person has different motion characteristics as compared to the limbs of a person. Hence, we choose SNR ahead of velocity while selecting the features. **Fourth,** we add some amount of noise to the x and y values of the resulting point clouds. Similar to the case of adding supplementary point clouds, the added noise has Gaussian distribution with mean as 0. This last step is taken because the literature reports that adding noise to the training data makes the model resilient under noise and minor variations [30].

Model Training: Across different combinations of m and MF, the input layer of the model is changed to accommodate the varying number of features. The architecture remains the same and in each case the number of features extracted remains 512. We randomly split the data into training, validation and test sets in a 70:15:15 ratio. At this stage, the data is ready for being given as input to the neural network model.

	MF=1	MF=2	MF=3	MF=5
m=3	0.71	0.76	0.80	0.79
m=4	0.77	**0.83**	**0.83**	0.83
m=5	0.77	0.81	0.82	0.82

TABLE VIII: Accuracy across different combinations of features and multi-frame factors.

VII. Evaluation of Neural Network

A. Accuracy across different configurations

We evaluate the performance of the network model for different combinations of features (m) and multi-frame factors (n) as shown in Tables VI and VII. The performance indicates the capability of the model to identify the number of people in different-sized clusters. Table VIII summarizes the accuracy across different combinations. We observe that the accuracy increases with MF but saturates at MF=3. Thus processing successive frames simultaneously indeed improves performance, but the law of diminishing return kicks in early. Similarly, the accuracy improves when we add SNR as a feature (MF=4) but not further when velocity is also considered (MF=5). Thus, the optimum accuracy is 83% when m=4 and MF is either 2 or 3. Table IX presents various performance metrics for these two combinations. We observe that the performance is balanced across both precision and recall.

Metric	F1 score	Accuracy	Precision	Recall
m=4 MF=2	0.8278	0.8306	0.8297	0.8306
m=4 MF=3	0.8348	0.8371	0.8387	0.8371

TABLE IX: Performance metrics for two different combinations of m and MF.

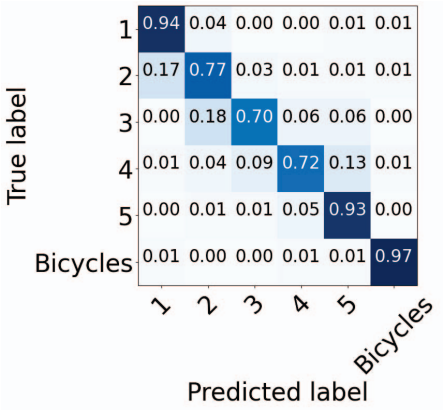


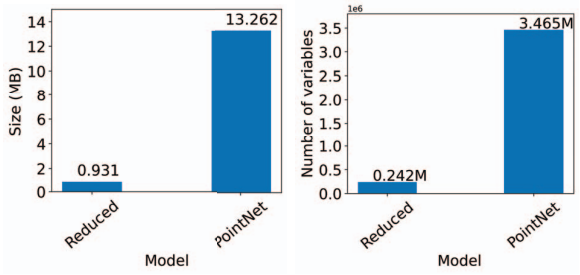
Fig. 13: Confusion matrix for m=4 and MF=3.

B. Observations from Confusion Matrix

Figure 13 plots the normalized confusion matrix for the case when m=4 and MF=3. We have a few critical observations from this confusion matrix. Firstly, we see that the clusters with a single person and bicycles are recognized better. In the case of bicycles, we believe this may be due to the distinct shape characteristics. Secondly, in case of misclassification, the probability that a cluster gets misclassified to an adjacent class is higher. Unlike classical multi-class classification, the adjacent clusters in our case are related and imply an error of one person in the count. In other words, if a cluster of three people gets misclassified, there is a higher chance that it is counted as a cluster of either two or four people. Additionally, we observe that for three and four-people clusters, the errors are balanced. For example, a three-people cluster gets undercounted 18% of the time and overcounted 13% of the time. Similarly, a four-people cluster gets undercounted 13% and overcounted 15% of the time. This observation is significant for people counting applications. In Section II, we mentioned that current people counting systems aggregate data over fifteen minutes and then transmit it. Our results imply that as we aggregate people count over a longer duration, the errors will tend to cancel each other out.

C. Model size

As discussed in Section V, we scaled down the PointNet model to adapt to the people counting application. Figure 14a and Figure 14b compare the size and the number of variables of the scaled-down model with the original PointNet model. We observe that we have been able to achieve a 93% reduction in model size. This lightweight model could facilitate execution from the edge in future enhancements.



(a) Model size. (b) Number of variables.

Fig. 14: Comparison of the reduced model with the original PointNet model.

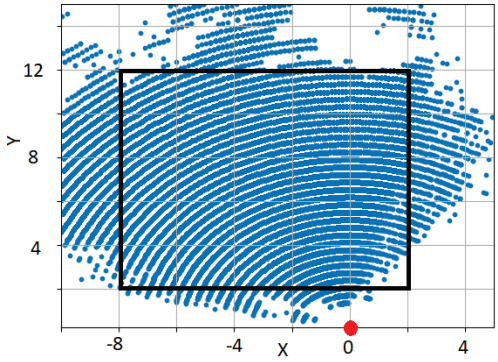


Fig. 15: Radar coverage. The red dot depicts the radar's location.

D. Range and coverage of the radar

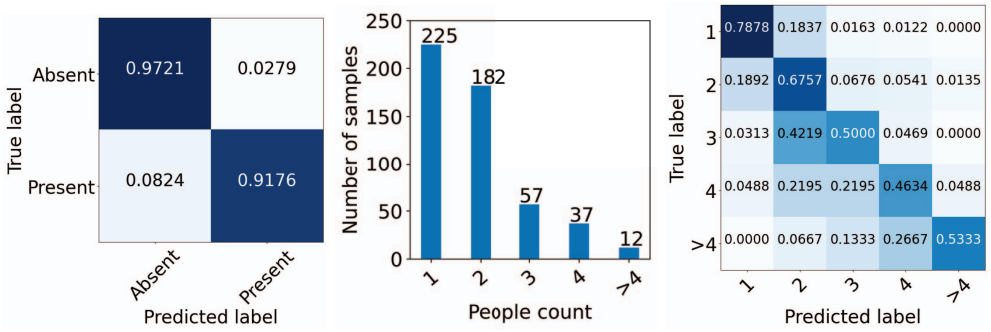
Figure 15 depicts the coverage of the mmWave radar. In this figure, we plot all the point clouds collected over 2 hours, while the volunteers walked during data collection. From the figure, we observe that the radar covers reasonably well the area between x= -8 and x= 2 and y= 2 and y= 12, an area greater than 100 m^2, effectively monitoring the entire road width. The coordinate (0, 0) indicated by the red dot in the figure points to the location of the radar. More significantly, the coverage area is conical with a field of view of 120 degrees, with a maximum range along the cone's axis. This information can help in planning future deployments.

VIII. Performance during Deployment

Having evaluated the neural network on the data gathered during training, we evaluate the performance of the system in a live deployment with real traffic. We have used the model trained with $m = 4$ and MF=3 during our deployment.

A. Data collection and processing

Figure 10 shows the setup with the camera installed next to the mmWave sensor. The mmWave point cloud data and the camera images have been collected over a continuous duration of 72 hours. The data collection was done in the second week of September in Amsterdam with temperatures varying between $15°C$ and $29°C$. We configured the camera to capture one picture every minute. Thus, a total of 4320 pictures were collected. Sixty-five of these pictures (equivalent to one hour of data) had to be discarded due to insufficient clarity.



(a) Presence detection performance. (b) Number of samples per count. (c) People counting performance.

Fig. 16: Performance during deployment.

For the remaining 4255 pictures, we compute the count in each picture manually with the help of a group of volunteers. We consider this camera count as the true count. The mmWave radar continuously collects and stores the point clouds during this period. The time stamps of the camera pictures are recorded and the radar frames corresponding to these time stamps are extracted. The point clouds undergo clustering, cluster identification and count aggregation to finally provide a count of people corresponding to the scene at each time stamp as explained in Section V-C. The camera timestamp has a resolution of one second. In cases where multiple mmWave frames correspond to the same timestamp, we compute the number of people for each mmWave frame and consider their average as the count for that time instant. The estimated count is rounded to the nearest integer and compared with the count obtained from the camera pictures. We observed that nearly 88% samples show zero count. To avoid skewing our results by these samples, we separately analyze scenes where either the camera or mmWave count zero people. Thus, our approach could be divided into two stages. First, presence detection functionality, where we identify if the scene is empty or has at least one subject. Second, people count functionality, where we estimate the number of people on scenes where both sensors indicate presence.

B. Presence detection

Figure 16a plots the confusion matrix for presence detection. We observe that the accuracy of presence detection is greater than 96%. This implies that there are no significant false detections by mmWave confirming the efficacy of the noise rejection mechanism during clustering.

C. People count

We analyze the 12% samples (513 images) where the number of people counted by the camera and mmWave sensor is non-zero. Figure 16b plots the number of samples for each of the counts and Figure 16c plots the normalized confusion matrix. The overall accuracy is 68.62%. Further, we observe again that in the case of misclassification, the probability that a cluster is misclassified to an adjacent class is higher. Also, the true people count across all the scenes is 1005, and the number estimated by the mmWave sensor is 980 - an undercounting of only 2.48%. This reaffirms that the errors are balanced. Next, we compare the camera images with the corresponding point clouds to investigate the reasons for misclassification.

(a) 3 people with a trolley counted as 4. (b) 1 person with a pet counted as 2.

Fig. 17: Overcounting due to additional objects the model did not encounter during training.

(a) 2 people walking together counted as 1. (b) 3 people walking together counted as 2.

Fig. 18: Undercounting due to occlusion.

D. Analysis and Future enhancements

To the best of our knowledge, this is the first work that deploys an mmWave sensor in real-life conditions and analyzes the errors observed in the deployment. This provides meaningful insights for the further development of mmWave sensing for people counting. We derive future directions based on these analysis.

Overcounting due to new objects: During the actual deployment, the system encountered many objects that it has not been trained on. Figure 17 shows two such examples where the presence of a trolley and a pet led to overcounting. Similarly, people carrying garbage bins, and bicycles with child trolleys caused overcounting. This suggests the need for more training under diverse deployment conditions.

Undercounting due to occlusion: Figure 18 shows two examples where occlusion has caused undercounting. In such

(a) 6 people standing together counted as 2.

(b) Sitting and standing subjects together counted as 1.

Fig. 19: Undercounting due to no movement.

cases, one or more people are blocked at the time the camera and mmWave sensor record the data. However, it is likely that they appear unoccluded in a frame a few seconds later. This provides us a future direction to consider the temporal information while counting. Machine learning frameworks such as LSTM could be explored to sense the scene over a longer duration leading to more accurate inference.

Undercounting due to lack of movement: As discussed in Section VI-C, for people counting, we only consider frames with points that have velocity greater than zero. Figure 19 shows scenarios where undercounting occurs because people do not move. The implementation from TI [26] partly addresses this problem through an algorithm that tracks the clusters and holds the count for a certain time if the objects turn static. We plan to integrate a similar approach to enhance the correctness of our count.

The above scenarios from the real deployment are insightful but have not been considered or reported in the related literature [13]. These inisghts suggest that the performance of the radar sensor could be further improved by focusing on the limitations encountered under practical situations.

E. Comparison with count from image recognition algorithm:

The images captured by the camera are processed by an open source crowd monitoring algorithm developed by the city [31]. We calibrated the algorithm after the camera was deployed with around 100 images across diverse lighting conditions. Figure 20 shows the comparison of errors across the three days. The average error for the camera is 0.3, higher than the -0.02 error reported by the mmWave. The corresponding standard deviations are 0.7 (for the camera) and 0.33 (for the mmWave). We see that especially during nights, the mmWave radar performs better than the camera. While it may be possible to minimize the camera errors by using better state of the art vision algorithms, our work highlights the challenges faced by camera based systems under low light conditions. Vision based algorithms must be trained across a large sample size and more diverse lighting conditions for accurate estimation. On the other hand, the performance of mmWave is not affected by light, lowering the demand for samples under different lighting. In our case, the model was trained with samples collected only during the day, however it performed equally well during the night.

IX. Related Work

Based on our literature survey, there are not many studies evaluating the performance of mmWave for outdoor people

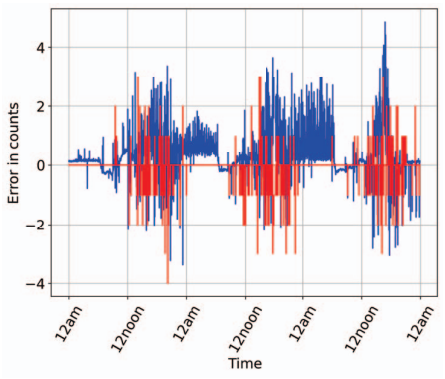


Fig. 20: Error comparison for the image recognition algorithm and mmWave radar. Blue indicates the error for the camera and red the error for the mmWave error.

counting. Table X summarizes the most relevant works we came across in this domain. Most studies [11], [18], [26], [32] demonstrate people counting with mmWave in indoor deployments. Since indoor scenarios have minimal background noise and do not require placing a radome around the sensor (no extra attenuation), their approach cannot be mapped directly to our application. Further most of these works are carried out in controlled scenarios.

The two most relevant works, evaluated outdoors, are [12] and [13]. Our earlier work [12] deploys the mmWave radar enclosed in a radome at a university campus. However, due to coverage inconsistencies between the camera system and the mmwave sensor, the accuracy could not be measured in an reliable manner. In this work, we deploy a more robust setup to compare against camera systems. Further, we achieve better presence detection (96% in our case as compared to 85% for [12]) and people counting (68% accuracy with no aggregation in our case as compared to 60% accuracy with 15-minute aggregation for [12]). Also, the work in [12] uses the algorithm provided by Texas Instruments for counting. That algorithm cannot estimate the number of people inside a point cloud cluster. The authors in [13] address this shortcoming by focusing on disaggregating the cluster. That work takes the radar cube as input and passes the features from the range-azimuth map together with cadence velocity diagrams to a statistical classifier for estimating the count in each group. The evaluation was performed in an outdoor scenario but under controlled settings and without a radome. As discussed in Section VI, the radome is central in an outdoor deployment but causes attenuation, making the sensing process more challenging. Our work addresses the shortcomings of [12] and [13] by developing a network model to accurately identify the count in a group of people and investigating the performance of radars under real traffic conditions.

X. Conclusion

We demonstrate a novel method to count people in a privacy-friendly manner using an mmWave sensor. We carefully craft the parameters of a clustering algorithm and train a deep-learning model for better noise rejection and accurate

Title	Environment	Input	Deployment	Approach
TI lab for people counting [26]	Indoor	Point cloud	Controlled environment	Group tracker approach tracking a group of point clouds
A Field People Counting Test Using Millimeter Wave Radar in the Restaurant [11]	Indoor	Point cloud	Real deployment in a restaurant	Not stated (Perhaps default TI algorithm)
mid: Tracking and identifying people with millimeter wave radar [18]	Indoor	Point cloud	Controlled environment	Deep learning based
Improved People Counting Algorithm for Indoor Environments using 60 GHz Radar [32]	Indoor	Radar cube	Controlled environment	Vital sign verification using micro doppler signatures.
A Long-Term Study Of mmWave Sensing In An Outdoor Urban Scenario [12]	Outdoor (with radome)	Point cloud	Real deployment in campus	Default TI algorithm
Grouped People Counting Using mm-wave FMCW MIMO Radar [13]	Outdoor (without radome)	Radar cube	Controlled environment	Statistical clustering with range-azimuth and cadence velocity diagrams
Our current work	Outdoor (with radome)	Point cloud	Real deployment across a street	Clustering with DBSCAN and neural network for group size estimation.

TABLE X: Summary of related works.

counting inside a point-cloud group. After deploying the radar across a street and evaluating its performance, our results show that our system performs better than the camera system currently used as a pilot by the city of Amsterdam. This outcome shows that an mmWave system can not only be more privacy-friendly but also more accurate. Our real evaluation also exposes new challenges that have not been reported in prior SoA studies, which are based on more ideal setups.

ACKNOWLEDGEMENTS

The authors would like to thank the members of the Responsible Sensing Lab at the Amsterdam Institute for Advanced Metropolitan Solutions (AMS) for providing the infrastructure and facilitating the permissions for the deployment. The research was supported by a grant from AMS.

REFERENCES

[1] "Tada principles," https://www.amsterdam.nl/innovation/digitalisation-technology/data/tada-principles/.
[2] "Leiden university students fight against classroom surveillance cameras," https://dutchreview.com/news/protests-cameras-leiden-university//.
[3] "Universiteit utrecht stopt met 'druktecamera's' in collegezalen na studentenprotest in leiden," https://www.rtvutrecht.nl/nieuws/3245218///.
[4] "Occupancy by the numbers," https://www.hikvision.com/sg/core-technologies/ai-analytics/people-counting///.
[5] "People counting sensors," https://www.xovis.com/technology/sensor.
[6] J.-H. Hoepman, "Privacy design strategies (the little blue book)," 2018.
[7] J. Guan, S. Madani, S. Jog, S. Gupta, and H. Hassanieh, "Through fog high-resolution imaging using millimeter wave radar," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
[8] S. Palipana, D. Salami, L. A. Leiva, and S. Sigg, "Pantomime: Mid-air gesture recognition with sparse millimeter-wave radar point clouds," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2021.
[9] D. Salami, R. Hasibi, S. Palipana, P. Popovski, T. Michoel, and S. Sigg, "Tesla-rapture: A lightweight gesture recognition system from mmwave radar sparse point clouds," *IEEE Transactions on Mobile Computing*, 2022.
[10] M. A. Alanazi, A. K. Alhazmi, O. Alsattam, K. Gnau, M. Brown, S. Thiel, K. Jackson, and V. P. Chodavarapu, "Towards a low-cost solution for gait analysis using millimeter wave sensor and machine learning," *Sensors*, 2022.
[11] S. Li and R. Hishiyama, "A Field People Counting Test Using Millimeter Wave Radar in the Restaurant," 2021.
[12] W. Wang, G. Vaidya, A. Bhattacharjee, F. Fioranelli, and M. Zuniga, "A long-term study of mmwave sensing in an outdoor urban scenario," in *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2023.
[13] L. Ren, A. Yarovoy, and F. Fioranelli, "Grouped people counting using mm-wave fmcw mimo radar," *IEEE Internet of Things Journal*, 2023.
[14] Texas Instruments, *60GHz mmWave Sensor EVMs*, May 2022.
[15] C. Iovescu and S. Rao, "The fundamentals of millimeter wave sensors," *Texas Instruments*, 2017.
[16] Z. Yang, Q. Cheng, P. Chu, H. Tan, and M. Zhou, "A bi-direction people counting method based on multi-scale clustering and multiple extended target tracking using mimo radar," *IEEE Sensors Journal*, 2023.
[17] X. Min, M. Jiang, L. Qiao, Y. Chen, and Z. Cao, "An efficient pointlstm for point clouds based gesture recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
[18] P. Zhao, C. X. Lu, J. Wang, C. Chen, W. Wang, N. Trigoni, and A. Markham, "mid: Tracking and identifying people with millimeter wave radar," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2019.
[19] "DBSCAN," https://scikit-learn.org/stable/modules/clustering.html#dbscan.
[20] M. Wang, F. Wang, C. Liu, M. Ai, G. Yan, and Q. Fu, "Dbscan clustering algorithm of millimeter wave radar based on multi frame joint," in *2022 4th International Conference on Intelligent Control, Measurement and Signal Processing (ICMSP)*, 2022.
[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.
[22] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PloS one*, 2010.
[23] J. James, "The distribution of free-forming small group size." *American Sociological Review*, 1953.
[24] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
[25] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015.
[26] "People counting demo," https://dev.ti.com/tirex/explore/node?a=VLyFKFf__4.8.0&node=A__APApso6UOq70CrevvkG5Sw__com.ti.mmwave_industrial_toolbox__VLyFKFf__4.11.0.
[27] "Size of radar cube," https://e2e.ti.com/support/sensors-group/sensors/f/sensors-forum/830400/iwr6843-size-of-radar-data-cube.
[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017.
[29] "Agglomerative clustering," https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.
[30] M. Zhou, T. Liu, Y. Li, D. Lin, E. Zhou, and T. Zhao, "Toward understanding the importance of noise in training neural networks," in *International Conference on Machine Learning*. PMLR, 2019.
[31] "Public eye: an open-source crowd monitoring solution," https://www.amsterdam.nl/innovation/mobility/public-eye/.
[32] J. Weiß, R. Pérez, and E. Biebl, "Improved people counting algorithm for indoor environments using 60 ghz fmcw radar," in *2020 IEEE Radar Conference (RadarConf20)*, 2020.