# Benchmark Data and Method for Real-Time People Counting in Cluttered Scenes Using Depth Sensors

Shijie Sun, Naveed Akhtar, Huansheng Song, Chaoyang Zhang, Jianxin Li, and Ajmal Mian

***Abstract*— Vision-based automatic counting of people has widespread applications in intelligent transportation systems, security, and logistics. However, there is currently no large-scale public dataset for benchmarking approaches on this problem. This paper fills this gap by introducing the first real-world RGB-D people counting dataset (PCDS) containing over 4500 videos recorded at the entrance doors of buses in normal and cluttered conditions. It also proposes an efficient method for counting people in real-world cluttered scenes related to public transportations using depth videos. The proposed method computes a point cloud from the depth video frame and re-projects it onto the ground plane to normalize the depth information. The resulting depth image is analyzed for identifying potential human heads. The human head proposals are meticulously refined using a 3D human model. The proposals in each frame of the continuous video stream are tracked to trace their trajectories. The trajectories are again refined to ascertain reliable counting. People are eventually counted by accumulating the head trajectories leaving the scene. To enable effective head and trajectory identification, we also propose two different compound features. A thorough evaluation on PCDS demonstrates that our technique is able to count people in cluttered scenes with high accuracy at 45 fps on a 1.7-GHz processor, and hence it can be deployed for effective real-time people counting for intelligent transportation systems.**

***Index Terms*— People counting, intelligent transportation, computer vision, large-scale data, cluttered scenes, RGB-D videos.**

## I. Introduction

AUTOMATIC people counting in real-time has multiple applications in intelligent public transportation systems [1], [2], [3], security, surveillance, logistics and resource management [4]. One effective method to reliably accomplish this task is to directly analyze continuous video stream of the region of interest, and perform automatic counting of people in those videos. For intelligent public transportation systems, such as buses with on-line monitoring, knowing the number of people entering and leaving the transport can be used in e.g. dynamic planning to avoid congestion. Public Transport Authorities can exploit the fundamental information of passenger congestion to build optimal bus scheduling models [5]. It also promises significant economic benefits by improving transportation scheduling in accordance with human traffic on stations at different hours of operation.

Computer Vision techniques are well-suited to the problem of automatic people counting for public transportations. However, using conventional RGB videos for this purpose is challenged by multiple issues resulting from real-world conditions such as clutter, occlusions, illumination variations, handling shadows etc. In comparison to the conventional video systems, RGB-D cameras (e.g. Kinect V1 [6], Prime Sense Camera [7]) can mitigate these issues by providing 'depth' information of the scene in addition to its color video. Nevertheless, effective people counting in real-world conditions using depth information still remains a largely unsolved problem due to noise and occlusion [8].

Vision-based people counting is a comprehensive task that involves object detection, recognition, and tracking. Existing approaches in this area can be broadly categorized into three classes: (a) regression-based methods, e.g. [9], [10] (b) clustering-based methods, e.g. [11], [12], and (c) detection-based methods, e.g. [13], [14]. Regression-based methods aim at learning a regression function using features of detection regions and exploit that for counting. Clustering-based methods track a set of features of target objects, and cluster their trajectories for counting them. Detection-based methods have a common pipeline, comprising foreground extraction, target localization, tracking, and trajectory classification. We can further divide these methods based on the data types they use e.g. color/depth/hybrid video methods (see Sec. II for the details). Although useful, the above mentioned approaches face some common problems while counting people under practical conditions in real-time, which include; restriction of camera angles [15]–[17]; computational inefficiency [18], and failing to handle cluttered scenes [19]. In this work, we propose a novel method for counting people using depth sensors that addresses these issues.

Moreover, to the best of our knowledge, there is no large-scale public dataset currently available to benchmark methods for real-world people counting. Hence, this paper fills this gap

Manuscript received July 25, 2018; revised December 3, 2018 and February 10, 2019; accepted March 31, 2019. Date of publication April 23, 2019; date of current version October 2, 2019. This work was supported by the Australian Research Council (ARC) under Grant DP160101458, in part by the National Natural Science Foundation of China under Grant 61572083, in part by the Joint Fund of Ministry of Education of China under Grant 6141A02022610, in part by the Fundamental Research Funds for the Central Universities under Grant 310824171003, and in part the China Scholarship Council (CSC). The Associate Editor for this paper was Z. Duric. *(Corresponding author: Naveed Akhtar.)*

S. Sun, H. Song, and C. Zhang are with the School of Information Engineering, Chang'an University, Xi'an 710000, China (e-mail: shijiesun@chd.edu.cn; hshsong@chd.edu.cn; zhaoyang_zh@chd.edu.cn).

N. Akhtar, J. Li, and A. Mian are with the School of Computer Science and Software Engineering, The University of Western Australia, Crawley, WA 6009, Australia (e-mail: naveed.akhtar@uwa.edu.au; jianxin.li@uwa.edu.au; ajmal.mian@uwa.edu.au).

Digital Object Identifier 10.1109/TITS.2019.2911128

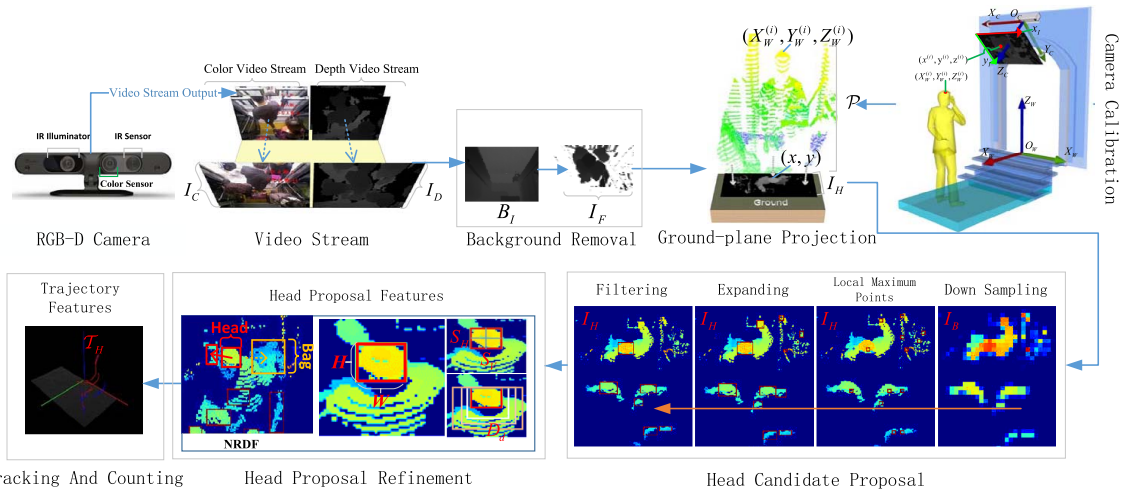


Fig. 1. Schematics of the proposed approach. The RGB-D camera provides color and depth video streams. The proposed method uses a depth frame ($I_D$) to extract the foreground for which 3D-point cloud is computed. The point-cloud is orthogonally projected onto the ground plane to generate a height image. Multiple potential head locations in the height image are computed by analyzing object features. These potential head proposals are refined and tracked continuously to count people entering or exiting a bus door.

by introducing the first large-scale dataset for counting people in real-world scenes of bus entrance/exit doors. The dataset, called **P**eople **C**ounting **D**ata**S**et (PCDS) contains 4,689 videos acquired with the Kinect V1 camera [6] that contains RGB and depth sensors. The dataset can be publicly downloaded using the following URL (https://github.com/shijieS/people-counting-dataset.git). Each video in the dataset is labeled with the number of people entering or exiting the bus door. The data has been collected on three different bus routes at different times of the day up to 6 different days, and presents large variations in terms of illumination, occlusion, clutter and noise. Our real-time people counting technique is thoroughly evaluated using the PCDS. The proposed method capitalizes on the depth video stream of the dataset to solve people counting problem. It is emphasized that whereas our method exploits the depth data for its inherent robustness against the real-world noise in terms of e.g. shadows, illumination variations; PCDS provides complete RGB-D videos for the broader research community.

Fig. 1 illustrates the pipeline of the proposed method. Our approach assumes the setting where the camera is mounted on top of the area to be monitored (see 'Camera Calibration' in the figure). This is the most common setting in scenarios like bus doors, corridors, entrances/exits to market places etc. After retrieving the depth video stream from the input, we subtract the scene background using a proposed procedure for the real-time performance of our approach. A 3D-point cloud of the foreground is computed from the depth information and then re-projected orthogonally onto the ground plane for effective segmentation. For each video frame, we analyze its projected height image for the presence of potential human heads while employing a 3D-human model to refine those proposals. The refined proposals are tracked to compute the head trajectories that are further refined and continuously monitored in our approach to count people entering or leaving the buses. To achieve our objective, we also introduce two discriminative feature vectors for head detection in height images and trajectory tracking in frame sequences. Our approach is evaluated using PCDS, and achieves up to 92% accuracy for the real-world bus videos while enabling processing at 45 fps on a relatively less powerful 1.7GHz Intel processor with 2GB RAM. These results are significant since people counting is a challenging problem and our method can achieve real-time performance in practical conditions with minimal computational resources.

This paper is organized as follows. The related work is reviewed in Sec. II followed by Sec. III which describe the published dataset named PCDS. In Sec. IV, we introduce the proposed approach for people counting in cluttered environment. The experiments and results are provided in Sec. V. In Sec. VI, we draw the conclusion.

## II. Related Work

The problem of people counting is often seen from two different perspectives: (a) Region of Interest (ROI) counting [4], [20]–[22], and (b) Line of Interest (LOI) counting [23]. The former deals with counting people in specific regions (e.g. in playgrounds), whereas the latter aims at counting the number of people 'passing through' a certain region (e.g. through doorways). This work deals with the LOI counting. Many methods for LOI counting have been proposed, which can be divided into three major categories: 1) regression-based methods, 2) clustering-based methods, and 3) detection-based methods. Below, we review literature under these categories with emphasis on detection-based methods because of their close relevance to the proposed approach.

### A. Regression-Based Methods

The main objective of the regression-based methods is to learn a regression function as the representation of changes in a scene which indicates passing of a pedestrian. Under the

paradigm of regression-based approach, Barandiaran *et al.* [9] used a single RGB camera to count people by the state change of virtual counting lines. Del Pizzo *et al.* [14] proposed a method which divides the detection region into stripes and counts people by monitoring the change of state for these stripes without people head detection and object tracking steps. Fradi and Dugelay [10] used Gaussian Mixture Model (GMM) to extract the foreground and used Gaussian Process regression to learn the correspondence between frame-wise features and the number of persons. Benabbas *et al.* [24] proposed a method which accumulates image slices and estimates the optical flow. They applied a linear regression model to blob features which are extracted by an on-line blob detector to get the position, velocity, and orientation of the pedestrian. Cong *et al.* [23] estimated the number of pedestrians passing through a line by quadratic regression with the number of weighted pixels and edges which are extracted from the flow velocity field. Whereas useful, one common drawback of the above methods is that they place hard restrictions on camera installation angles and the scene itself, which compromises their practical value.

### *B. Clustering-Based Methods*

Clustering based methods simultaneously track multiple features of objects e.g. key points or people component, and subsequently count people by clustering feature trajectories. For instance, Antonini *et al.* [11] clustered trajectories of visual features and then used the number of clusters for counting people. Topkaya *et al.* [12] used features based on spatial, color and temporal information and clustered the detected feature trajectories by Dirichlet Process Mixture Models (DPMMs) [25]. They used Gibbs sampling to estimate an arbitrary number of people or groups in their approach. Brostow and Cipolla [26] proposed a method that first tracks simple image features and then probabilistically groups them into clusters based on space-time proximity and trajectory coherence through the image space. Rabaud and Belongie [27] used KLT tracker [28] to track feature points, and segmented the set of trajectories by a learned object descriptor.

### *C. Detection-Based Methods*

The approaches that fall under this category share a common sequential processing pipeline which goes as follows. First, foreground is extracted from the video stream, then the objects of interest are detected and tracked. The tracked trajectories are subsequently classified to count the objects of interest. The detection-based methods can be further divided into three different groups based on the underlying data modalities, namely 1) RGB video methods, 2) Depth video methods, and 3) Hybrid methods. We also include an additional category in our review that includes approaches employing the fast emerging deep learning framework.

*1) RGB Video Methods:* Using RGB videos is more popular in people counting literature because of easy availability of color video cameras. Zeng and Ma [13] detected head-shoulder patterns in RGB videos by combining multilevel HOG features [29] with multilevel LBP features [30]. They used PCA [31] to reduce the dimensionality of the multilevel HOG-LBP feature set, and finally tracked the head-shoulder patterns to count people. Antić *et al.* [32] proposed a people segmentation, tracking, and counting system by using an overhead mounted camera. Garcia *et al.* [33] also developed an RGB system for counting people in supervised areas. Their method is based on finding heads of people by a circular pattern detector and tracking them using Kalman filter [34]. Their approach also performs the final counting using the tracked trajectories. Chen *et al.* [15] used a vertical RGB video-camera to count a crowd of moving people by segmenting the crowd based on the frame difference method [35]. Their approach extracts features to describe the individual patterns, and tracks the individuals for counting. Kurilkin *et al.* [36] compared different people detectors in their study.

The methods described above are likely to suffer from critical failures when the scenes become complicated due to shadows, light changes, compound objects, occlusion, and the presence of significant background texture. To alleviate these problems, researchers exploit stereo cameras which can provide the third dimension information. For instance, Terada *et al.* [37] proposed one of the first approaches for stereo camera based people counting in RGB video regime. They detected people using max points, tracked them with template matching and finally used the two measurements from the stereo vision for counting. In a related approach, Kristoffersen *et al.* [38] used two thermal cameras to reconstruct 3D points and proposed an algorithm for pedestrian counting based on clustering and tracking of the 3D point clouds. However, in their approach, the cost of depth computation remains high, which makes it difficult to use the approach in real-time with low computational power devices.

*2) Depth Video Methods:* With the popularity of RGB-D cameras; such as Kinect V1/V2 [6] and Prime-Sense [7], depth videos are also becoming popular in people counting applications. Zhang *et al.* [39] proposed to use a so-called 'water filling method' to detect people and counting them by the virtual line in a depth image. Barandiaran *et al.* [9], and later Pizzo *et al.* [14], [40] proposed approaches that are based on detection without tracking. These approaches detect changes in scene states across a virtual line, where the scene is divided by multiple stripes. The state of the scene changes when people pass by, thereby enabling people counting. Rauter [41] introduced the Simplified Local Ternary Patterns (SLTP) that are used to describe a human head. They trained an SVM using SLTP and tracked human heads with the nearest neighbor association methods. Vera *et al.* [42] proposed a network of cameras to count people. They devised a head detection method based on morphology geodesic reconstruction [43] and performed tracking using the Hungarian algorithm [44]. Their approach combines tracks generated by multiple cameras and the final count is based on the length of the combined track. Li *et al.* [16] proposed an embedded framework for real-time top-view people counting. They used the Kinect camera and the Jetson TK1 board [45] to detect human heads using the water filling technique [39]. Their approach also uses the nearest neighbor association method for tracking. Enrico Bondi *et al.* [46] introduced a framework for real-time people counting which follows the sequence of

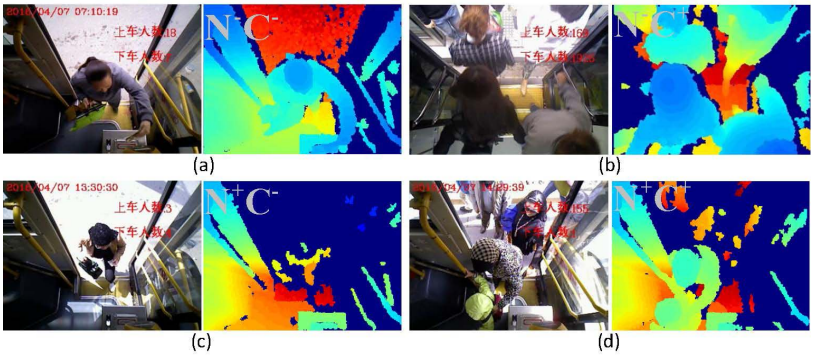


Fig. 2. Representative RGB and depth images of different scenarios in PCDS. (a) Normal one-person entry. (b) Multiple people using the same door for entering and disembarking. (c) Noisy sensor data. (d) Multiple people queuing with partial occlusion. PCDS contains multiple videos for each case shown.

background removal, head detection and tracking the projected heads. Whereas promising, their framework drastically performs in complicated scenarios where other head-like objects also appear in the scenes.

*3) Hybrid Methods:* Combining the advantages of RGB and depth data streams are well documented in related problems, e.g. action recognition [27]. Therefore, few methods have also used the hybrid approach in people counting. For instance, Gao *et al.* [17] detected head candidates in depth videos by water filling method and refined these candidates by training an SVM classifier using HOG features of the frame in RGB videos. Their approach eventually generates a set of trajectories by the nearest distance between the head candidates and the previous tracks. Liu *et al.* [47] also used RGB-D camera for detecting people. Their approach projects people into a virtual plane and trains an SVM classifier using features that are used for detecting the upper body of humans. Zhang *et al.* [48] proposed head detection by blob detection in depth frames and projected the blobs into the 3D space. Their approach filters the candidate blobs for heads by physical constraints and employs the histogram of multi-order depth gradient (HMDG) features and joint histogram of color and height (JHCH) features to train an SVM classifier. The trained SVM is used to classify the candidate blobs as heads. However, their approach remains sensitive to occlusion.

*4) Deep Learning-Based Methods:* Recently, Deep Learning [49]–[51] has demonstrated great success in object detection and classification tasks [52], which has also motivated researchers to employ it for the problem of people counting. For instance, Liu *et al.* [18] proposed a people counting system based on Convolutional Neural Network (CNN) [53] and Spatio-Temporal Context (STC) model [54]. The CNN model is used to detect people whereas the STC model is used to track heads of moving people. Similarly, Wei *et al.* [55] proposed a framework based on supervised learning. They extracted spatio-temporal multi-features by joining super-pixel based multi-appearance features and multi-motion features, and then fused the multi-features with the features extracted from the VGG-16 model [56].

### D. RGB-D Datasets

One of our major contributions is in introducing the first large-scale RGB-D dataset for the problem of people counting in outdoor settings. Indeed, a few RGB-D public datasets already exist for the related problems of e.g. people detection [57], people tracking [58], and estimating size of inhomogeneous crowd [4]. However, to the best of our knowledge, currently no large-scale RGB-D dataset exists for counting people at entrances/exits in outdoor settings. The unique setup adopted in this work makes the proposed data useful for developing techniques that find application in intelligent transport systems, surveillance, security and logistics etc.

## III. People Counting Dataset (PCDS)

In this Section, we present the **P**eople **C**ounting **D**ata**S**et (PCDS) introduced for the problem of people counting in real-world conditions. The dataset is publicly available for download at https://github.com/shijieS/people-counting-dataset.git. The provided URL also contains further explanation of the proposed dataset. Below, we focus on the most relevant details.

### A. Settings and Data Taxonomy

The data consists of videos of bus-door scenes recorded using Kinect V1 camera [6]. The camera is mounted on the ceiling of (front/back) doors of different buses, and captures people entering or exiting through the doors. Fig. 2 shows four representative scenes from the dataset. Due to the real-world scenarios, complexity of the data is apparent from the figure. Note that our dataset and method (Sec. IV) also account for passengers entering and leaving through the same door simultaneously, as shown in Fig. 2b. In comparison to the existing related datasets [14], [39], videos in PCDS are recorded by the camera with a pitch angle that is not necessarily vertical to the ground plane.

We divide the videos in the dataset based on the bus route numbers. The dataset is recorded for three different bus routes, namely No. 25, No. 301 and No. 106 in the cities of Xi'An, XiNing and YinChuan, respectively in China. The data samples cover all the bus stops in the complete circuit route of the buses. For No. 25, the videos have been collected on 6 different days. For No. 301 and No. 106, the number of days are 5 and 4 respectively. For each day, we collected data for the front door as well as the back door. Thus, in total, there are 30 different scenes in our dataset. We can

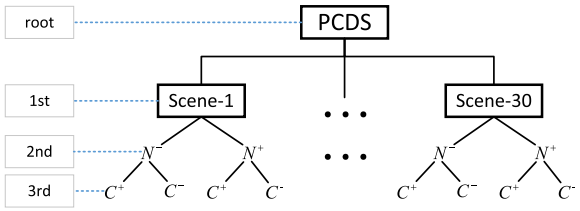


Fig. 3. Structure of the **p**eople **c**ounting **d**ata**S**et (PCDS). The root directory contains 30 scenes, with two subdirectories each. Each subdirectory either contains noisy scenes (denoted by $N^+$) or clean/normal scenes (denoted by $N^-$). The 3rd-level of directories contains crowded (i.e., $C^+$) or un-crowded (i.e., $C^-$) scenes.

TABLE I
THE NUMBER OF PEOPLE IN PCDS

| | $N^-C^-$ | $N^-C^+$ | $N^+C^-$ | $N^+C^+$ |
|---|---|---|---|---|
| Entering | 2,704 | 5,427 | 616 | 937 |
| Exiting | 2,760 | 6,647 | 668 | 1,149 |
| Total | 5,464 | 12,074 | 1,284 | 2,086 |

further sub-categorize the videos of these scenes based on their noise level and crowd in the scene. In the above mentioned URL for the dataset, we organize the dataset according to these measures. In Fig. 3, we provide the folder structure of the proposed dataset. We denote the noisy/clean scenes with $N^+/N^-$, and crowded/un-crowded scenes by $C^+/C^-$. As an example, a noisy-crowded scene is denoted by $N^+C^+$ according to the adopted notation.

The rationale of dividing the dataset into noisy and clean videos is that Kinect V1 camera is sensitive to illumination conditions. For strong illumination, there is often noise in the videos, as can also be observed in Fig. 2. The videos in our dataset are mainly recorded in either direct sunlight or diffused sunlight, resulting in a natural division of corresponding levels of noise. Similarly, the division of videos according to congestion in the scenes is also natural. During rush hours, multiple people are generally passing through the bus doors. On the other hand, sequential entry with clear separation between people is observed during normal conditions.

In light of the division provided in Fig. 3, one can expect the following from the four possible sub-categories for each scene in the proposed dataset:

- $N^+C^+$: Videos are captured in strong sunlight during rush hours, with multiple people attempting to enter/exit the bus at once.
- $N^+C^-$: Videos recorded in sharp sunlight during normal hours where people are entering/exiting bus doors in a more sequential manner.
- $N^-C^+$: The recording is performed with mild sunlight but in crowded situations.
- $N^-C^-$: The recording is done in mild/diffused sunlight with sequential entry/exit of people through the doors.

Tab. I summarizes the number of people entering and exiting the bus doors for each sub-category. We manually computed these numbers using the following strategy. Each video was watched by at least two viewers who independently counted the number of people entering and disembarking. In the case of mismatch in the counts, the procedure was repeated until all the viewers agreed to the same number.

TABLE II
VIDEO ATTRIBUTES

| Type | fps | Resolution | Channels | Count |
|---|---|---|---|---|
| RGB video | 25 | $320 \times 240$ | 3 | 4,689 |
| Depth video | 25 | $320 \times 240$ | 1 | 4,689 |

### B. Video Information

In Tab. II, we summarize the basic attributes of the videos in our dataset. We note that, these video attributes along the camera parameter details are also provided in each folder of the dataset. Moreover, the total number of people passing through the doors is also provided as the ground truth. We also provide RGB videos along the depth videos that can be used for verification purposes. However, we emphasize that the depth modality is more useful for the problem of people counting in the real-world conditions because of its robustness to e.g, illumination conditions and shadows.

## IV. PROPOSED APPROACH

The schematics of the proposed approach for people counting is illustrated in Fig. 1. Our method performs counting by analyzing the depth video frames retrieved from the RGB-D camera. The major steps involved in our approach are; 1) removing the scene background, 2) re-projecting point cloud onto the ground plane, 3) generating candidate head proposals in the projected images, 4) refining those proposals, and 5) tracking the trajectories of human heads for counting. We provide details of each of these steps below.

### A. Background Removal

There are multiple techniques for background subtraction from RGB videos [13], [59], [60]. However, depth videos are inherently different from RGB videos and such methods are not readily applicable to them. Few methods for background removal from depth videos also exist [61], [62]. However, those techniques are generally computationally expensive, which makes them unsuitable for our real-time application. Moreover, such methods were also found to be unsuitable for handling the noise in PCDS resulting from the real-world conditions. Therefore, we develop our own method for efficient background subtraction from depth videos for people counting scenarios, such that the results also remain robust to noise in the real-world data.

In our settings, a depth frame $I_D \in \mathbb{R}^{H \times W}$ is a matrix, with its each element representing the distance of a point in the real-world to the camera sensor. For a camera mounted on top of the area to be monitored (as in PCDS), one can expect that the farthest points in the scene would generally belong to the background. Based on this simple intuition, we develop a 'farthest background model' $B_I \in \mathbb{R}^{H \times W}$ of dynamic scenes that enables automatic estimation of the background on-the-fly. A major advantage of such an approach is that it can be readily used for any scene without the need of calibration for the background.

We compute $B_I$ as a map of the largest distances appearing in the sequences of depth frames, while accounting for the

possible noise accumulation. To ensure that effective $B_I$ is available for each video frame, we take the help of two intermediate models $B_c$ and $B_{2c}$, where '$c$' stands for cache. We initialize $B_I$ and $B_c$ with $I_D$ at the start of the video stream ($B_{2c}$ is initialized later, see below). For an input frame sequence, we update $B_c$ at every frame as follows:

$$B_c^t = max\{I_D^t, B_c^{t-1}\}, \tag{1}$$

where the superscript '$t$' denotes the current frame and $t-1$ indicates the previous frame. The $max\{.\}$ operation is performed element-wise. After every $n_c$ frames, we update $B_I$ by assigning it the values of $B_c$.

It is easy to see that under the above mentioned strategy, any large distance values in $B_c$ resulting from noise at any stage can eventually get stored in $B_I$. To cater for this problem, we separately initialize $B_{2c}$ with $I_D$ just after $B_I$ is updated (i.e. after $n_c$ frames), and keep updating it with every frame as follows:

$$B_{2c}^t = max\{I_D^t, B_{2c}^{t-1}\}. \tag{2}$$

We update $B_c$ as well as $B_I$ with $B_{2c}$ after each $n_{2c}$, whereas we impose that $n_{2c} - n_c \neq 0$ to ensure that the update of $B_I$ under $B_c$ and $B_{2c}$ is asynchronous. This strategy entails that a maximum value once entered in $B_I$ as a result of noise can be replaced by the correct smaller value in the later frames. For computational purpose, we also constrain $n_{2c} > 2n_c$. As a result of the asynchronous updates with intermediate models, effective $B_I$ remains available for each frame. We use this farthest background model to extract the foreground $I_F$ at each frame as follows:

$$I_F^t(u,v) = \begin{cases} 0, & |B_I^t(u,v) - I_D^t(u,v)| < \delta_{dis} \\ I_D^t(u,v), & \text{otherwise} \end{cases} \tag{3}$$

where $B_I^t(u,v)$ is the pixel value at $(u,v)$ position of $B_I^t$, $I_D^t(u,v)$ is the pixel value at $(u,v)$ position of $I_D^t$, and $\delta_{dis}$ denotes the threshold parameter for our approach.

### B. Reprojection

Generally, cameras used for counting people are installed with non-zero pitch angle, e.g. see 'Camera Calibration' in Fig. 1. The camera perspective often causes occlusion and overlap in the depth maps of people, which adds to the complexity of counting problem. The objective of "reprojection" stage is to remove the perspective distortions so that individuals become well separated in the reprojected depth frames. For that purpose, we first construct a 3D point cloud from a depth frame of the camera and then re-project it normally onto the ground plane to obtain a normalized depth image. We present details of the reprojection procedure below.

First, we convert the foreground image $I_F$ into 3D points in the camera coordinates. For every pixel in $I_F$, we recover its 3D point as follows:

$$\begin{cases} X_C = \dfrac{u - c_x}{f_x} \cdot I_F(u,v) \\ Y_C = \dfrac{v - c_y}{f_y} \cdot I_F(u,v) \\ Z_C = I_F(u,v), \end{cases} \tag{4}$$

where $(f_x, f_y)$ denote the camera focal length, $(c_x, c_y)$ is the camera principal point, $I_F(u,v)$ is the pixel value at the position $(u,v)$ in $I_F$, and $X_C, Y_C, Z_C$ are the recovered 3D point coordinates.

For projecting points onto the ground, we must first convert 3D points in the camera coordinates to the world coordinates. Let us denote the world coordinate frame by $\{X_W, Y_W, Z_W, O_W\}$. We fix this frame directly below the camera coordinate reference frame, as shown in Fig. 1. To perform the transformation between the coordinate frames, we compute the homogeneous transformation matrix ($\mathbf{T} \in \mathbb{R}^{4\times4}$) based on the extrinsic parameters of the camera. To that end, we first identify $N$ points in a depth frame acquired by the camera and physically measure the corresponding points in the world coordinates. The following optimization problem is then solved using the least squares approach [63]:

$$< \mathbf{T} >= \min_{\mathbf{T}} ||\mathbf{P}_W - \mathbf{T} \ \ \mathbf{P}_C||_F^2, \tag{5}$$

where $\mathbf{P}_W \in \mathbb{R}^{4\times N}$ contains $N$ points arranged as its columns in the world coordinates, and $\mathbf{P}_C \in \mathbb{R}^{4\times N}$ contains the corresponding points in the camera coordinates. The last row of these matrices consist of 1s. For a unique solution, we constrain $N > 4$ in our measurements.

Note that, estimation of $\mathbf{T}$ is an off-line process in our approach and it is performed only once for calibration. Using the matrix $\mathbf{T}$ we eventually transform all points in $I_F$ to a 3D point cloud in the word coordinates. We then project this point cloud normally onto the ground plane. Intuitively, multiple points in the 3D point cloud can be mapped to the same point on 2D ground plane. In our approach, we only store the 2D mappings of the highest points in the 3D point cloud. Concretely, for the points $(X_W^{(i)}, Y_W^{(i)}, Z_W^{(i)}), \forall i$ in the 3D point cloud, we compute a 2D ground plane projection $I_H(x,y)$ as follows:

$$\begin{cases} \mathcal{Z}^{(x,y)} = \{Z_W^{(i)} | X_W^{(i)} = x \wedge Y_W^{(i)} = y, \forall i\} \\ I_H(x,y) = \max(\mathcal{Z}^{(x,y)}) \end{cases} \tag{6}$$

where $(x,y)$ indexes points in the 2D plane. Henceforth, we refer to $I_H$ as the "height image" because each point/pixel in this image represents the highest point in the corresponding 3D point cloud. The effects of reprojection can be understood as acquiring the depth image from a camera mounted directly above a person as opposed to a tilted camera. In the illustrations to follow, e.g. Fig. 4; the top-view of the height images results from the performed reprojection.

### C. Candidate Head Proposals

Although the reprojected height images separate individuals well, the loss of information due to occlusions in the original depth frames can not be recovered from these images. Due to their height, human heads suffer the least from the occlusions caused by the camera perspective. Therefore, instead of tracking the complete human body to count people, we focus on reliable localization of human heads in the height images and eventually use head trajectories for people counting.

To locate human heads in the reprojected images, we exploit our prior knowledge about a human body. We make use of

TABLE III
THRESHOLDS OF EACH PART OF HUMAN MODEL

| | | Head | | Shoulder | | Lower body | |
|---|---|---|---|---|---|---|---|
| | | min | max | min | max | min | max |
| $L$ | pixels | 10 | 25 | 25 | 60 | / | / |
| $W$ | pixels | 10 | 25 | 10 | 20 | / | / |
| $H$ | cm | 15 | 30 | 10 | 30 | 60 | 170 |

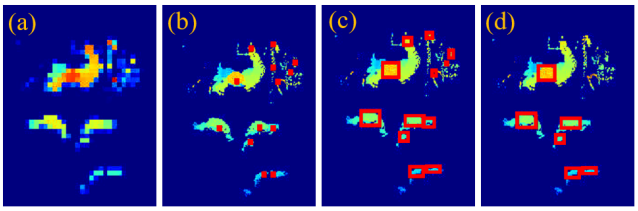


Fig. 4. Intermediate results for generating head proposals. a) Down sampled height image. b) Local maximum point in original height image. c) Expanding local maximum points. d) Filtering the local maximum areas. The images are cropped for better visualization.

a 3D human model, by dividing the body into three major parts, namely; *Head*, *Shoulders*, and *Lower body*. We place a cuboid on each of these parts that represents the volume where the body part is most likely to be located in the 3D space. For instance, we expect the *head* to be located in the cuboid of dimensions $W_h, H_h, L_h$. The exact dimensions of the cuboid would vary from person to person. Therefore, we empirically place minimum and maximum thresholds on these dimensions in our approach. The used thresholds are summarized in Tab. III. Our intuition is that we can locate the corresponding body parts of individuals in the height image using the human model. Therefore, the thresholds in the table cover reasonably large ranges to account for the variability in human sizes in height images.

With the help of underlying human model, we generate candidate proposals about the human heads possibly present in the height image by sequentially performing the following steps. 1) Down sampling the height image, 2) computing the local maxima in the down sampled image, 3) expanding the local maximum points, and 4) filtering the expanded areas. Below, we describe each of these steps in detail.

*1) Down Sampling:* Recall that our aim is to develop a real-time method that can deal with the real-world noise. To reduce computations and mitigate the adverse effects of noise in this step, we first down sample the height image $I_H$ by averaging its $w_b \times w_b$ dimensional disjoint patches. As a result, we get an image $I_B$ with its pixel at $(x, y)$ location computed as follows:

$$I_B(x, y) = \frac{\sum_{u=w_b x}^{w_b x + w_b} \sum_{v=w_b y}^{w_b y + w_b} I_H(u, v)}{w_b^2}, \tag{7}$$

where, $(u, v)$ denotes a pixel location in $I_H$.

*2) Local Maximum Point Computation:* Intuitively, the pixels corresponding to human heads are more likely to have the largest values in height images. This property is also well preserved in the down sampled image $I_B$, as can be seen in Fig. 4a. Thus, to locate the areas that can potentially belong to human heads in $I_B$, we adopt a simple strategy of identifying a set $\mathcal{C}_b$ of the pixels in $I_B$ that contain the maximum values in their 8-connected pixels. These pixels are then used to identify the local maximum points in the original height image. Note that, the $i^{\text{th}}$ element of $\mathcal{C}_b$, i.e. $\mathcal{C}_b^i$ is computed as the mean of a set of pixels in the height image. We represent the set of the desired maximum pixels in $I_H$ as $\mathcal{C}_H$, and compute the $j^{\text{th}}$ element of that set, i.e. $\mathcal{C}_H^j$ as follows:

$$\mathcal{C}_H^j = max\{\text{pixels in } I_H \text{ corresponding to } \mathcal{C}_b^i\}. \tag{8}$$

As a result of this operation, we are able to efficiently identify the local maximum points in our height image. Fig. 4b illustrates the computed points from the corresponding down sampled image in Fig. 4a.

*3) Expanding Local Maximum Points:* A local maximum point in $I_H$ may or may not belong to a human head. Therefore, we must analyze the local vicinity of the maximum point and compare it with our human model to ascertain that the point is indeed located on a human head. We adapt the seed fill method [64] to expand the local maximum points into rectangles such that the object bounded by each rectangle can be compared with the human model. The procedure for expanding the local maximum point is given as Algorithm 1.

Along the height image $I_H$ and the set of local maximum points $\mathcal{C}_H$, the algorithm requires the maximum allowable height and width of a head (from Tab. III) as the input. It also

**Algorithm 1** *ExpandingLocalMaximumPoints*

**Input:**
$I_H$: height image. $\mathcal{C}_H$: local maximum point set.
$\delta_h$: expanding threshold. $W_{max}$: maximum head width.
$L_{max}$: maximum head length.
**Output:**
$\mathcal{E}_H$: the set of expanded rectangles.
**Initialize:**
$H \leftarrow rows(I_H)$; % height of the image.
$W \leftarrow columns(I_H)$; % width of the image.
**for each pixel** $(x_0, y_0) \in \mathcal{C}_H$ **do**
  $l \leftarrow W; r \leftarrow 0; t \leftarrow H; b \leftarrow 0;$
  $\mathcal{C}_0 \leftarrow \{(x_0, y_0)\}; \mathcal{C}_s \leftarrow \emptyset;$
  **for each pixel** $(u, v) \in \mathcal{C}_0$ **do**
    $\mathcal{N}_P$ is 8-connected pixels of $(u, v)$
    **for each pixel** $(x, y) \in \mathcal{N}_P$ **do**
      **if** $(x, y) \notin \mathcal{C}_s$ **and** $|I_H(x_0, y_0) - I_H(x, y)| \leq \delta_h$ **and** $r - l \leq W_{max}$ **and** $b - t \leq L_{max}$ **then**
        $l \leftarrow min(x, l); r \leftarrow max(x, r);$
        $t \leftarrow min(y, t); b \leftarrow max(y, b);$
        $\mathcal{C}_s \leftarrow \mathcal{C}_s \cup \{(x, y)\}$
        $\mathcal{C}_0 \leftarrow \mathcal{C}_0 \cup \{(x, y)\}$
      **end if**
    **end for**
    $\mathcal{C}_0 \leftarrow \mathcal{C}_0 - \{(u, v)\}$
  **end for**
  $\mathcal{E}_H \leftarrow \mathcal{E}_H \cup \{(l, r, t, b)\}$
**end for**
**return** $\mathcal{E}_H$

uses an expanding threshold $\delta_h$ as an input parameter, that restricts the expanded rectangles to contain object pixels with similar values. The algorithm eventually results in a set $\mathcal{E}_H$ that contains the expanded rectangles as its elements. The main iteration of Algorithm 1 runs over each element of $\mathcal{C}_H$, that are called *seeds* in the context of seed fill method [64]. For each local maximum pixel, $l, r, t$ and $b$ respectively denote the left, right, top and bottom of the expanded rectangle. On line 2 of the algorithm, these values are initialized with $W, 0, H, 0$. On line 3, $\mathcal{C}_0$ records all the pixels that need to be considered in the inner loop, whereas $\mathcal{C}_S$ stores all the pixels which have been processed. $\mathcal{C}_S$ is initialized as $\emptyset$. The first inner 'for loop' (lines 4-15 in the algorithm) performs the actual expansion process. It first identifies the 8-connected neighborhood of a considered pixel (line 5) and then iterates over each of the neighboring pixels (line 6-13) to evaluate the condition given on line 7 of the algorithm. If the condition is satisfied, the $l, r, t, b$ values are updated and this pixel is added into $\mathcal{C}_s$ and $\mathcal{C}_0$. The algorithm gradually expands a seed in $\mathcal{C}_H$ to a rectangle that is upper-bounded by the maximum dimensions $W_{max}$ and $L_{max}$ while ensuring that the pixel values in the expanded rectangles remain close to the seed value so that the rectangle only bounds a single object.

*4) Filtering Local Maximum Areas:* Despite capitalizing on the physical attributes of human body parts, we can still expect that few rectangles in $\boldsymbol{\mathcal{E}}_H$ may not actually belong to human heads (see e.g. Fig. 4c). Therefore, we further filter the computed rectangles using the human model. In the filtration process, we also consider incomplete human heads resulting from occlusions. To filter, we discard all the rectangles in $\boldsymbol{\mathcal{E}}_H$ that do not satisfy the following condition:

$$\begin{cases} \dfrac{L_{min}}{2} \leq L \leq L_{max} \\ \dfrac{W_{min}}{2} \leq W \leq W_{max}, \end{cases} \tag{9}$$

where, $L$ and $W$ represent the length and width of a rectangle and the subscripts $min, max$ denote the minimum and maximum lengths allowed in Tab. III for 'Head'. Notice that we reduced the minimum allowed values in Eq. (9) by half. This is done to account for occlusions that can often cause the size of a head in our height image $I_H$ to reduce significantly.

### D. Head Proposals Refinement

For the cameras installed on top of pathways, human heads in video frames rarely overlap in real scenes, as can be observed in Fig. 5a-b. However, the set $\boldsymbol{\mathcal{E}}_H$ may contain few overlapping rectangles (see Fig. 5c), therefore we can further refine this set by discarding the overlapping rectangles. To do that, we consider all the groups of overlapped rectangles in $\boldsymbol{\mathcal{E}}_H$, and for each group, we store only the rectangle with the highest seed value and discard the remaining rectangles. We denote the refined set of rectangles by $\boldsymbol{\mathcal{F}}_H$. Fig. 5d illustrates the result of this refinement.

The rectangles contained in the set $\boldsymbol{\mathcal{F}}_H$ are highly likely to correspond to human heads in $I_H$, however it is still possible that some of those rectangles may actually belong to other objects in the scene. Differentiating between a head

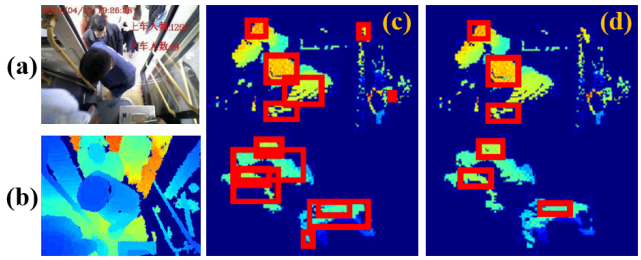


Fig. 5. Result of removing overlapped rectangles: a) color image, b) depth video frame, c) expanded local maximum areas, and d) result of removing overlapped rectangles. The region of interest in height images is cropped and expanded for better visualization.

and a non-head rectangle in $\boldsymbol{\mathcal{F}}_H$ is a non-trivial task because occlusions and other factors, e.g. presence of high round-shaped objects like bag-backs, can result in patterns in $I_H$ that are very similar to human heads. We hypothesize that despite their close similarity with the human heads, the non-head objects can be automatically identified by analyzing their relevant features. Hence, we design a compound discriminative feature that accounts for different relevant attributes of objects to classify them as 'heads' and 'non-heads'. We momentarily defer the discussion on the proposed feature to the text to follow. We extract the proposed features for the elements of $\boldsymbol{\mathcal{F}}_H$ and train an SVM classifier over those features to further discard the rectangles that bound non-head objects.

For the SVM training, we manually label each extracted feature for a rectangle as 'head' or 'non-head'. This off-line training is carried out only once in our approach on the training data. For the test frames, we similarly extract the features of head proposals and classify them as 'heads' or 'non-heads' using the trained SVM. The 'non-heads' are discarded in further processing. Our proposed compound feature vector is a concatenation of two major types of features that we call *Basic Geometric Features* (BGF) and the *Nearest Rectangle Difference Feature* (NRDF). The BGF itself is a combination of four different features explained below:

- *Shape Feature* $(H_r, W_r, R, P)$, where $H_r$ is the height of a rectangle, $W_r$ is the width of the rectangle, $R$ is the ratio of $W_r$ to $H_r$ and, $P = H_r \times W_r$.
- *Symmetry Feature* $(S_H, S_V)$, where $S_H$ captures the horizontal symmetry and $S_V$ represents the vertical symmetry. We define $S_H$ and $S_V$ as follows:

$$\begin{cases} S_H = \dfrac{2 \sum\limits_{y=t}^{t+H_r} \sum\limits_{x=l}^{l+\frac{W_r}{2}} |I_H(x,y) - I_H(2l + W_r - x, y)|}{H_r W_r} \\ S_V = \dfrac{2 \sum\limits_{x=l}^{l+W_r} \sum\limits_{y=t}^{t+\frac{H_r}{2}} |I_H(x,y) - I_H(x, 2t + H_r - y)|}{H_r W_r} \end{cases} \tag{10}$$

where, $(l, t)$ denotes the top-left corner point of the rectangle. Inclusion of this feature in our compound feature is motivated by the natural symmetry of human heads.

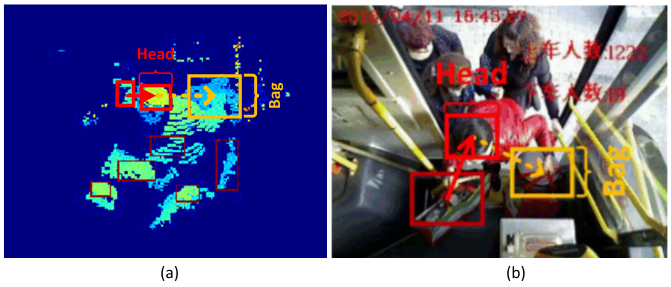


Fig. 6. Illustration of NRDF feature. a) The height image. b) The corresponding color video frame. There is a bag rectangle (yellow) in both images, whose nearest rectangle is the head (red). The yellow arrow is the NRDF feature of the bag rectangle and the red arrow is the NRDF feature of the head rectangle.

- *Zero Pixel Feature* $(N_0, R_0)$, where $N_0$ denotes the number of zero pixels appearing in the image area bounded by a rectangle, and $R_0 = \frac{N_0}{H_r W_r}$ is the rate of zero pixel appearance.
- *Expansion Ratio Feature* $\Psi \in \mathbb{R}^5$ contains the ratios of the area of a rectangle in $\mathcal{F}_H$ to five different rectangles achieved by using different expansion thresholds $\delta_h$ in Algorithm 1. By varying the values of the expansion threshold we can expect different rectangles resulting for different kinds of objects in the scene. Therefore, the expansion ratio feature provides important clues about an object being a head or not. In our original algorithm, we let $\delta_h = 15$ to arrive at the set $\mathcal{E}_H$. To compute the expansion ratio feature, we select the values of $\delta_h$ from $\{20, 25, 30, 35, 40\}$ to generate five different rectangles corresponding to each element of $\mathcal{F}_H$ and calculate $\Psi$ for each element.

We concatenate the above mentioned four geometric features into a vector in $\mathbb{R}^{13}$. Notice that, although we do consider varied areas of $I_H$ in the above mentioned features, the compound feature only accounts for the information that is local to individual rectangles. In the real-world scenarios, the relative locations of the rectangles (that we suspect to contain human heads) can provide useful information about a bounded object being a human head or not. Therefore, we further define NRDF to account for this additional information. For each rectangle in $\mathcal{F}_H$, we compute NRDF as a vector in the 3D-space that is directed towards the center of the rectangle from the center of its nearest rectangle in our current set of head proposals. This feature is further illustrated in Fig. 6. The resulting NRDF $\in \mathbb{R}^3$ is concatenated with the above mentioned feature vector to finally arrive at our compound feature vector in $\mathbb{R}^{16}$.

### *E. Tracking and Counting*

Using the compound features introduced above, we refine the head proposals in $\mathcal{F}_H$. Notice that, this set is computed for a single depth frame in our approach. To eventually count the people passing through a scene, we must also track the trajectory of individual heads (i.e. people) in a continuous video stream. For that purpose, we exploit $\mathcal{F}_H$ in maintaining a record of head trajectories in the incoming video stream. We count the number of people passing by the camera by counting the number of trajectories disappearing in our records. We use the direction of movement to determine if the person has entered or exited the bus. Concrete technical details of this procedure are provided below.

To track individuals in the scene, we maintain a set of trajectories $\mathcal{T}$ for the continuous video stream. The set is initialized as 'empty' when the stream starts. With each frame the set gets updated by adding, removing or updating its elements. An element of this set is given by $\{\mathcal{F}_H^i, P^i\}$, where '$i$' indicates the $i^{\text{th}}$ element, and $P^i$ is the probability of that element bounding a human head. This probability is available to us from the SVM classifier trained to arrive at the refined set $\mathcal{F}_H$. In the text to follow, we refer to an element of $\mathcal{T}$ as a *node* for brevity.

To update nodes with each coming frame, we first match the potential nodes of the new frame with the current nodes in $\mathcal{T}$. To that end, we compute $\eta = ||(x_o - x_n), (y_o - y_n), (s_o - s_n)||_2$, where $(x_o, y_o)$ indicates the center of the rectangle represented by a node in $\mathcal{T}$, $(x_n, y_n)$ is the center of a rectangle in the new frame, and $s_o$ and $s_n$ are the seed values for the respective rectangles. We consider two nodes to be matched if $\eta < \delta_m$, where we empirically fix the value of $\delta_m$. If a new node does not match any existing node, it is added to $\mathcal{T}$ as a new element. If an existing node in $\mathcal{T}$ is not updated for $Q$ consecutive frames, we remove that node from our set. The removed node increments our count of a person passing by the camera. When a node is removed, we determine the direction of the movement performed by the individual (i.e. 'enter' or 'exit') by analyzing the centers of the first and the last rectangle for that node. The information on the centers of rectangles (and their seeds), number of updates for each node, and the time stamp of the last update for each node are maintained in our approach by book-keeping.

Using the simple strategy explained above, we can track the trajectories of individual objects in the scene. However, tracking of 'human-heads' in the above method completely relies on the accuracy of $\mathcal{F}_H$. If a non-head object still eludes our refinement process discussed in the preceding Section, the approach may count extra individuals in the scene. To circumvent this problem we exploit the observation that human-heads generally follow similar trajectories in path-ways, which can be differentiated from the trajectories of non-head objects. Thus, we train binary SVM classifiers (one each for 'entering' and 'exiting' directions) to identify a given trajectory in $\mathcal{T}$ as 'head' or 'non-head'. We propose another feature for training the classifiers that is formed by concatenating a) the mean and variance of all nodes involved in a trajectory, b) the total number of updates for the trajectory, c) velocity of the trajectory computed as the rate of change in the center locations of the bounding rectangles, and d) the difference between the maximum and the minimum seed values for the trajectory. We use the SVMs trained over these features to refine our final count of the people entering or exiting the bus doors/path-ways. We provide an illustration of tracking and counting process in Fig. 7.

## V. Experiment

We evaluate the proposed method using our proposed dataset, PCDS, that contains a large number of pedestrians

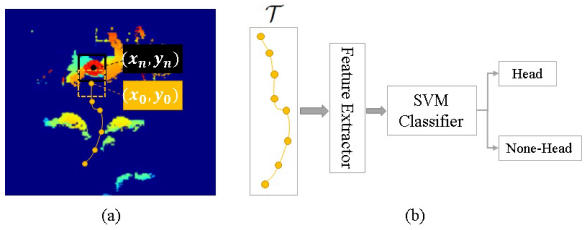


Fig. 7. Illustration of tracking and counting process. (a) Yellow rectangle represents the head detected in the previous frame, and yellow dots indicate a node of $\mathcal{T}$. Black rectangle is the head detected in the current frame. (b) The features of the node removed from $\mathcal{T}$ are extracted and input to SVM to label as head or none-head.

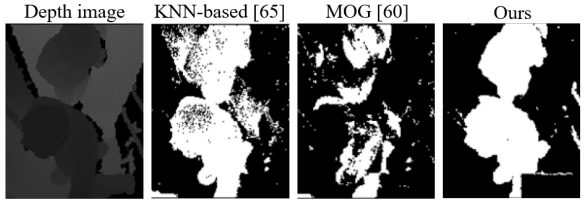


Fig. 8. Background subtraction. Images are cropped for better visualization.

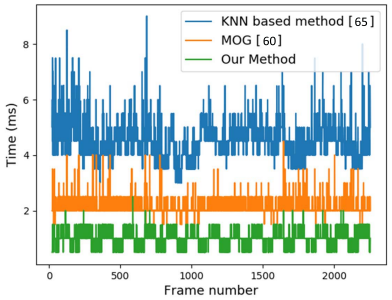


Fig. 9. Computational time for background subtraction for each frame. Timings for a sequence of 2,263 frames are shown.

TABLE IV
SUMMARY OF THE LABELED HEAD PROPOSALS USED

| | | train | test | total |
|---|---|---|---|---|
| entering | head | 3123 | 2058 | 5181 |
| | non-head | 4165 | 2802 | 6967 |
| exiting | head | 3409 | 2172 | 5581 |
| | non-head | 2872 | 1655 | 4527 |

entering/exiting bus doors imaged by a Kinect camera installed on top of the door. The dataset provides the opportunity to thoroughly evaluate the major components of our approach individually as well as analyze its performance for the overall task of people counting. We first analyze the efficacy of our background subtraction procedure and compare its performance with the popular MOG [60] and KNN-based methods [65]. Then, we separately analyze the performance of our method for the tasks of human head identification, human head tracking and finally, people counting as a whole.

### A. Background Removal

Background removal is a major task in many surveillance related problems. For our approach, reliable background subtraction is necessary for the success of subsequent processing of video frames. Therefore, we separately analyze the performance of our method for this task. We use the popular Gaussian mixture-based background segmentation method (MOG) [60] and the K-nearest neighbors (KNN) based method [65] to benchmark our technique. We note that other approaches for background subtraction also exist, however the selected baseline methods are chosen for their well-established effectiveness for the depth videos. We carefully optimized parameter values of the baseline methods on our dataset using cross validation. For the proposed method, we empirically chose $n_c = 150$ and $n_{2c} = 500$ in all our experiments.

Fig. 8 shows a typical mask image generated by MOG [60], KNN-based method [65], and the proposed background subtraction procedure. As can be seen, the mask images generated by both KNN and MOG methods contain significant amount of noise which can be detrimental for the subsequent processing in our approach. On the other hand, the proposed method is able to preserve the masks of individual humans very well, with negligible noise. For further qualitative analysis of background subtraction, we also provide videos comparing our method with the existing approaches on the following URL: https://youtu.be/oiuYq_Pfx6c.

Whereas our method achieves reliable background subtraction, it is also required to obtain those results efficiently for the overall task of *real-time* people counting. We show the computational time (in ms) for processing each frame of a typical frame sequence in our dataset for the proposed approach and the baseline methods in Fig 9. The time is computed on a 1.7GHz processor with 2GB RAM for the task of background subtraction. The proposed method averages around 1.0 ms/frame in comparison to 2.1 ms/frame and 4.5 ms/frame of MOG and KNN-based method respectively. High quality background subtraction with a small time required to process each frame makes our background subtraction highly desirable for the broader problem of real-time people counting.

### B. Human Head Identification

An essential component of counting people in our approach is to accurately identify human heads in the scene. We identify human heads by first generating candidate head proposals and then refining them. In our approach, the process of generating the candidate proposals is intentionally kept relaxed, and it also results in identifying multiple non-head objects in the scene (e.g. shoulders, bag-packs) to be considered as potential human heads. The refinement process (in Sec. IV-D) then discards the non-head objects to identify the human heads.

To analyze the performance of our method for human head identification, we first manually labeled 12,148 rectangle proposals in height images for people entering the buses as 'heads' and 'non-heads'. These proposals were generated automatically by the method in Sec. IV-C. We then trained and tested the SVM classifier employed in our approach using these proposals. We also performed the same routine for 10,108 candidate head proposals for the people exiting the buses. The details of the train-test distributions and the labels of proposals used in this analysis are provided in Tab. IV.

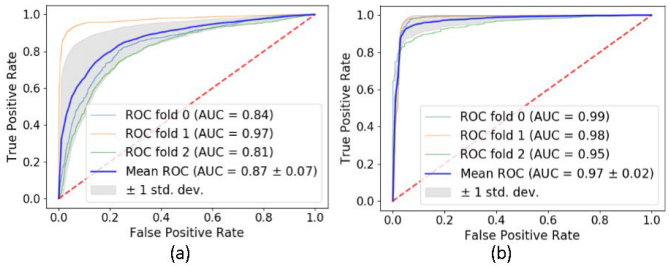


Fig. 10. The ROC curves of classifiers for head identification. a) People entering the buses: SVM parameters $\gamma = 20.2$ and $C = 16$. b) People exiting the buses: SVM parameters $\gamma = 3.4$ and $C = 520$.

TABLE V
EVALUATION OF HEAD IDENTIFICATION

| | | precision | recall | f1-score | sample |
|---|---|---|---|---|---|
| entering | head | 0.92 | 0.92 | 0.92 | 2058 |
| | non-head | 0.94 | 0.94 | 0.94 | 2802 |
| exiting | head | 0.95 | 0.92 | 0.93 | 2172 |
| | non-head | 0.97 | 0.98 | 0.98 | 1655 |

In Fig. 10, we show the ROC curves for the classifiers trained for the refinement of head proposals. The curves show results of our three-fold experiments, with corresponding AUC values.

From Fig. 10, we can argue that the employed classifiers are able to identify human heads in the proposals successfully. We note that the classification performance depicted by Fig. 10 is better for the people exiting buses than for the people entering buses. The reason behind this phenomenon is that while providing the ground truth we only labeled those proposal rectangles as 'heads' that bounded complete human heads. For the case of people entering the buses, many half-heads appeared in the frames due to queuing of people on bus doors. On scrutiny, we found that most of those heads resulted in false positive identifications in our experiment. However, this is not problematic for the overall approach because the final results rely more strongly on tracking of heads on multiple frames, and the half-heads eventually transform into complete heads in the subsequent video frames. We also provide the details of precision, recall and the f1-scores for our head identification experiment in Tab. V.

### C. Tracking

Our overall approach relies strongly on the tracking method introduced in Sec. IV-E. Similar to the head identification method, we separately analyzed the tracking procedure by evaluating the performance of the classifier employed for tracking. For that, we manually labeled 1,332 tracks in our dataset as 'head' and 'non-head' for people entering the buses. Among the labeled tracks, we used around 30% samples for testing and the remaining samples were used for training the classifier. We also followed the same routine for 1,330 tracks for people exiting the buses. The information on the test-train distribution and the labels of the tracks used in our analysis is summarized in Tab. VI. We empirically selected $\delta_m = 15$ and $Q = 8$ in our experiments.

TABLE VI
SUMMARY OF THE LABELED TRACKS USED

| | | train | test | total |
|---|---|---|---|---|
| entering | head track | 442 | 183 | 625 |
| | non-head track | 509 | 198 | 707 |
| exiting | head track | 544 | 226 | 770 |
| | non-head track | 406 | 154 | 560 |

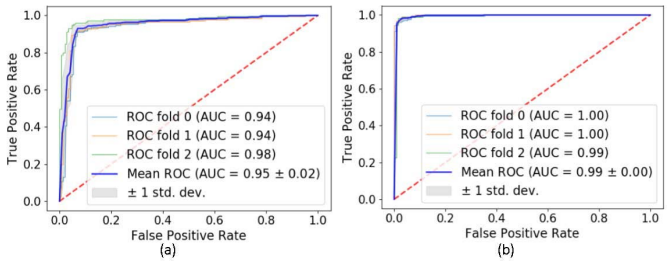


Fig. 11. The ROC curve of classifiers for tracking head trajectories. a) People entering the buses: SVM parameters $\gamma = 0.111$ and $C = 368$. b) People exiting the buses: SVM parameters $\gamma = 0.05882$ and $C = 896$.

TABLE VII
EVALUATION OF TRACKING PERFORMANCE

| | | precision | recall | f1-score | sample |
|---|---|---|---|---|---|
| entering | head track | 0.92 | 0.97 | 0.94 | 183 |
| | non-head track | 0.97 | 0.92 | 0.95 | 198 |
| exiting | head track | 0.98 | 0.97 | 0.98 | 226 |
| | non-head track | 0.96 | 0.97 | 0.97 | 154 |

In Fig. 11, we show the ROC curves of the classifiers used for head tracking in our approach. The figure also reports the AUC values for our three-fold experiments. It is easy to observe that our method is able to classify (i.e. track) the trajectories of human heads very accurately for both 'entering' and 'exiting' scenarios. Notice that no significant performance degradation is visible in Fig. 11a for the 'entering' scenario, which was the case in Fig. 10a. This is because tracking is performed over a sequence of frames and the incomplete heads (due to people queues) at the start of tracking eventually become irrelevant for the problem at hands. We also provide summary of the precision, recall and f1-scores of the tracking results in Tab. VII. The table indicates successful classification by the employed classifier.

### D. People Counting

The main objective of our approach is to perform people counting in real-time. We evaluated the people counting performance of our approach using 2,000 test videos from PCDS. We used detection rate '$\Delta$' as the metric for evaluations, which is defined as follows.

$$\Delta = \sum_{i=1}^{N_V} |n_i - \widetilde{n_i}| \Big/ \sum_{i=1}^{N_V} n_i \tag{11}$$

where $N_V$ is the total number of videos in the test data, $n_i$ is the number of people passing in the $i^{\text{th}}$ video, and $\widetilde{n_i}$ denotes the estimated number of people in the $i^{\text{th}}$ video.

TABLE VIII
PEOPLE COUNTING ACCURACY ON PCDS

| | $N^-C^-$ | $N^-C^+$ | $N^+C^-$ | $N^+C^+$ |
|---|---|---|---|---|
| entering | 85.40% | 83.25% | 77.54% | 75.32% |
| exiting | 93.04% | 92.66% | 93.71% | 91.30% |

In Tab. VIII, we report the detection rates of our method for the four different categories of videos introduced in Sec. III-A. We separately report the results for the 'entering' and 'exiting' scenarios. Based on these results, we can argue that the performance of our approach is acceptable for both scenarios. In PCDS, most of the people entering the buses use the front door. It was observed that due to significant glare from the glass of the front doors, the videos often contained large amount of noise. This generally made people counting at the front doors in the dataset more challenging. Nevertheless, the approach shows reasonable overall performance given the practical real-world conditions of the dataset.

Considering the potential low on-board computational capacity available for our method in the real-world deployment, we used a less powerful 1.7GHz Intel processor with 2GB RAM for evaluating our approach. On average, our method required 1.1ms for background removal, 15.4ms for head identification, and 5.6ms for track computation for a single frame. This amounts to 22.1ms processing time for a single frame, yielding processing of approximately 45 frames per second, which can be considered as real-time performance.

## VI. CONCLUSION

This article makes two important contributions to the problem of 'people counting' in real-world scenarios. Firstly, it presents the first large-scale benchmark public dataset for the problem. This dataset contains recorded depth videos, color videos and CSV format files with the labels containing the number of people passing through different scenes of bus doors. The videos account for a large variability in scene illumination, clutter, noise and other factors in the real-world environment, which makes the dataset particularly challenging. Secondly, the article presents a method for real-time people counting in cluttered scenes and evaluates the performance on the proposed dataset. The proposed method utilizes the depth video stream and computes a normalized height image of the scene after removing the background. The height image is essentially a projection of the scene depth directly below the camera, which helps in a clear segmentation of individual objects in the scene. This projection is used to identify heads of individuals in the scene. We utilize a 3D human model and adapt a seed fill method to reliably detect human heads. We also propose a compound feature for height images, that is utilized in our approach for head identification. Once reliably detected, individual human heads are tracked to compute their trajectory which is eventually utilized for people counting. We ascertain the effectiveness of our method by applying it to the proposed dataset. Our benchmark dataset will play a major role in advancing research in the areas of RGB-video, Depth-video and RGBD-video based people counting.

## REFERENCES


[1] B. Barabino, M. D. Francesco, and S. Mozzoni, "An offline framework for handling automatic passenger counting raw data," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2443–2456, Dec. 2014.

[2] Y. Wang, D. Zhang, L. Hu, Y. Yang, and L. H. Lee, "A data-driven and optimal bus scheduling model with time-dependent traffic and demand," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2443–2452, Sep. 2017.

[3] A. Ceder, "Bus frequency determination using passenger count data," *Transp. Res. A, General*, vol. 18, nos. 5–6, pp. 439–453, Oct. 1984.

[4] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and Bayesian regression," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2160–2177, Apr. 2012.

[5] W. Kim, B. Son, J.-H. Chung, and E. Kim, "Development of real-time optimal bus scheduling and headway control models," *Transp. Res. Rec.*, vol. 2111, no. 1, pp. 33–41, 2009.

[6] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, pp. 4–10, Apr. 2012.

[7] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," WO Patent 2008 0 240 502 A1, Dec. 12, 2010.

[8] T. Mallick, P. P. Das, and A. K. Majumdar, "Characterizations of noise in Kinect depth images: A review," *IEEE Sensors J.*, vol. 14, no. 6, pp. 1731–1740, Jun. 2014.

[9] J. Barandiaran, B. Murguia, and F. Boto, "Real-time people counting using multiple lines," in *Proc. 9th Int. Workshop Image Anal. Multimedia Interact. Services*, 2008, pp. 159–162.

[10] H. Fradi and J.-L. Dugelay, "Low level crowd analysis using frame-wise normalized feature for people counting," in *Proc. WIFS*, 2012, pp. 246–251.

[11] G. Antonini and J. P. Thiran, "Counting pedestrians in video sequences using trajectory clustering," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 8, pp. 1008–1020, Aug. 2016.

[12] I. S. Topkaya, H. Erdogan, and F. Porikli, "Counting people by clustering person detector outputs," in *Proc. AVSS*, 2014, pp. 313–318.

[13] C. Zeng and H. Ma, "Robust head-shoulder detection by PCA-based multilevel HOG-LBP detector for people counting," in *Proc. Int. Conf. Pattern Recognit.*, 2010, pp. 2069–2072.

[14] L. Del Pizzo, P. Foggia, A. Greco, G. Percannella, and M. Vento, "Counting people by RGB or depth overhead cameras," *Pattern Recognit. Lett.*, vol. 81, pp. 41–50, Oct. 2016.

[15] C. H. Chen, T. Y. Chen, D. J. Wang, and T. J. Chen, "A cost-effective people-counter for a crowd of moving people based on two-stage segmentation," *J. Inf. Hiding Multimedia*, vol. 3, no. 1, pp. 12–23, 2012.

[16] G. Li, P. Ren, X. Lyu, and H. Zhang, "Real-time top-view people counting based on a Kinect and NVIDIA jetson TK1 integrated platform," in *Proc. ICDMW*, 2017, pp. 468–473.

[17] C. Gao, J. Liu, Q. Feng, and J. Lv, "People-flow counting in complex environments by combining depth and color information," *Multimedia Tools Appl.*, vol. 75, no. 15, pp. 9315–9331, 2016.

[18] G. Liu, Z. Yin, Y. Jia, and Y. Xie, "Passenger flow estimation based on convolutional neural network in public transportation system," *Knowl.-Based Syst.*, vol. 123, pp. 102–115, May 2017.

[19] Y. P. Kocak and S. Sevgen, "Detecting and counting people using real-time directional algorithms implemented by compute unified device architecture," *Neurocomputing*, vol. 248, pp. 105–111, Jul. 2017.

[20] K. Chen and J. K. Kämäräinen, "Learning to count with back-propagated information," in *Proc. Int. Conf. Pattern Recognit.*, 2014, pp. 4672–4677.

[21] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," in *Proc. CVPR*, 2015, pp. 833–841.

[22] H. Idrees, I. Saleemi, C. Seibert, and M. Shah, "Multi-source multi-scale counting in extremely dense crowd images," in *Proc. CVPR*, 2013, pp. 2547–2554.

[23] Y. Cong, H. Gong, S.-C. Zhu, and Y. Tang, "Flow mosaicking: Real-time pedestrian counting without Scene-specific learning," in *Proc. CVPR Workshops*, 2009, pp. 1093–1100.

[24] Y. Benabbas, N. Ihaddadene, T. Yahiaoui, T. Urruty, and C. Djeraba, "Spatio-temporal optical flow analysis for people counting," in *Proc. AVSS*, 2010, pp. 212–217.

[25] R. M. Neal, "Markov chain sampling methods for Dirichlet process mixture models," *J. Comput. Graph. Statist.*, vol. 9, no. 2, pp. 249–265, Jun. 2000.

[26] G. J. Brostow and R. Cipolla, "Unsupervised Bayesian detection of independent motion in crowds," in *Proc. CVPR*, 2006, pp. 594–601.

[27] V. Rabaud and S. Belongie, "Counting crowded moving objects," in *Proc. CVPR*, 2006, pp. 705–711.

[28] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Imaging*, vol. 130, pp. 674–679, 1981.
[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR*, 2005, pp. 886–893.
[30] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[31] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.
[32] B. Antić, D. Letić, D. Ćulibrk, and V. Crnojević, "K-means based segmentation for real-time zenithal people counting," in *Proc. ICIP*, 2009, pp. 2565–2568.
[33] J. García, A. Gardel, I. Bravo, J. L. Lázaro, M. Martínez, and D. Rodríguez, "Directional people counter based on head tracking," *IEEE Trans. Ind. Electron.*, vol. 60, no. 9, pp. 3991–4000, Sep. 2013.
[34] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," *Proc. SPIE*, vol. 3068, Jul. 1997, p. 182.
[35] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *Proc. ICIG*, 2007, pp. 519–523.
[36] A. V. Kurilkin and S. V. Ivanov, "A comparison of methods to detect people flow using video processing," *Procedia Comput. Sci.*, vol. 101, pp. 125–134, Jan. 2016.
[37] K. Terada, D. Yoshida, S. Oe, and J. Yamaguchi, "A method of counting the passing people by using the stereo images," in *Proc. Int. Conf. Image Process.*, 1999, pp. 338–342.
[38] M. S. Kristoffersen, J. V. Dueholm, R. Gade, and T. B. Moeslund, "Pedestrian counting with occlusion handling using stereo thermal cameras," *Sensors*, vol. 16, no. 1, p. 62, 2016.
[39] X. Zhang, J. Yan, S. Feng, Z. Lei, D. Yi, and S. Z. Li, "Water filling: Unsupervised people counting via vertical Kinect sensor," in *Proc. AVSS*, 2012, pp. 215–220.
[40] L. Del Pizzo, P. Foggia, A. Greco, G. Percannella, and M. Vento, "A versatile and effective method for counting people on either RGB or depth overhead cameras," in *Proc. ICMEW*, 2015, pp. 1–6.
[41] M. Rauter, "Reliable human detection and tracking in top-view depth images," in *Proc. CVPR Workshops*, 2013, pp. 529–534.
[42] P. Vera, S. Monjaraz, and J. Salas, "Counting pedestrians with a zenithal arrangement of depth cameras," *Mach. Vis. Appl.*, vol. 27, no. 2, pp. 303–315, Feb. 2016.
[43] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 12, pp. 1163–1173, Dec. 1996.
[44] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, Mar. 1955.
[45] Y. Ukidave, D. Kaeli, U. Gupta, and K. Keville, "Performance of the NVIDIA Jetson TK1 in HPC," in *Proc. ICCC*, 2015, pp. 533–534.
[46] E. Bondi, L. Seidenari, A. D. Bagdanov, and A. Del Bimbo, "Real-time people counting from depth imagery of crowded environments," in *Proc. AVSS*, 2014, pp. 337–342.
[47] J. Liu, Y. Liu, Y. Cui, and Y. Chen, "Real-time human detection and tracking in complex environments using single RGBD camera," in *Proc. ICIP*, 2013, pp. 3088–3092.
[48] G. Zhang, J. Liu, L. Tian, and Y. Q. Chen, "Reliably detecting humans with RGB-D camera with physical blob detector followed by learning-based filtering," in *Proc. ICASSP*, 2016, pp. 2004–2008.
[49] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[50] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[51] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning TensorFlow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–284.
[52] J. Cao, Y. Pang, and X. Li, "Learning multilayer channel features for pedestrian detection," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3210–3220, 2017.
[53] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. CVPR*, 2012, pp. 3642–3649.
[54] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang, "Fast visual tracking via dense spatio-temporal context learning," in *Proc. ECCV*, 2014, pp. 127–141.
[55] X. Wei, J. Du, M. Liang, and L. Ye, "Boosting deep attribute learning via support vector regression for fast moving crowd counting," *Pattern Recognit. Lett.*, vol. 47, pp. 178–193, Mar. 2017.
[56] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[57] L. Spinello and K. O. Arras, "People detection in RGB-D data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 3838–3843.
[58] M. Munaro and E. Menegatti, "Fast RGB-D people tracking for service robots," *Auto. Robots*, vol. 37, no. 3, pp. 227–242, 2014.
[59] P. Kaewtrakulpong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*. Boston, MA, USA: Springer, 2001, pp. 135–144.
[60] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. CVPR*, vol. 2, 2004, pp. 28–31.
[61] C. R. del-Blanco, T. Mantecón, M. Camplani, F. Jaureguizar, L. Salgado, and N. García, "Foreground segmentation in depth imagery using depth and spatial dynamic models for video surveillance applications," *Sensors*, vol. 14, no. 2, pp. 1961–1987, 2014.
[62] M. I. Chacon-Murguia, O. A. Chavez-Montes, and J. A. Ramirez-Quintana, "Background modeling on depth video sequences using self-organizing retinotopic maps," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2016, pp. 1090–1095.
[63] A. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia, PA, USA: SIAM, 1996.
[64] S. Torbert, *Applied Computer Science*. Cham, Switzerland: Springer International Publishing AG, 2016.
[65] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, 2006.

**Shijie Sun** received the B.S. degree in software engineering from Chang'an University, where he is currently pursuing the Ph.D. degree in intelligent transportation and information system engineering. He is also pursuing the (joint) Ph.D. degree with The University of Western Australia. His research interests include machine learning, object detection, localization and tracking, and action recognition.

**Naveed Akhtar** received the Ph.D. degree in computer vision from The University of Western Australia (UWA). His research in vision and pattern recognition, regularly published in the prestigious venues of the field, including the IEEE CVPR and the IEEE TPAMI. He is currently a Research Fellow of UWA. He has also previously served on the same position at The Australian National University. His research interests include people counting and tracking, adversarial machine learning, and hyperspectral imaging.

**Huansheng Song** received the Ph.D. degree in information and communication engineering from Xian Jiaotong University, Xi'an, China, in 1996. Since 2004, he has been with the Information Engineering Institute, Chang'an University, where he became a Professor, in 2006, and he was nominated as the Dean, in 2012. He has been involved in research on intelligent transportation systems for many years. His current research interests include image processing and recognition and intelligent transportation systems.

**Chaoyang Zhang** received the B.S. and Ph.D. degrees in applied mathematics from Northwestern Polytechnical University (NWPU), Xi'an, China, in 2013 and 2007, respectively. He visited the Department of Computer Science, The University of Alberta, Edmonton, Canada, from 2011 to 2012. He has been a Teaching Fellow with Chang'an University, Xi'an, since 2014. His current research interests include linear and nonlinear feature extraction methods, pattern recognition, and remote sensing image processing.

**Jianxin Li** received the Ph.D. degree in computer science from the Swinburne University of Technology, Australia, in 2009. He has been invited to be the PC-Chair, the Proceedings Chair, and the General Chair of international conferences and workshops, including ADMA 2016, WWW 2017, and DASFAA 2018. He was also a PC Member of SIGMOD 2017, CIKM 2018, ICDM 2018, and ICDE 2019. He is a Reviewer of top journals such as TKDE. His research interests include database query processing and optimization, social network analytics, and traffic network data processing.

**Ajmal Mian** received the Ph.D. degree (Hons.) from The University of Western Australia, in 2006. He is currently a Professor of computer science with The University of Western Australia. His research interests include computer vision, machine learning, face recognition, 3D shape analysis, and hyperspectral image analysis. He has secured eight major national and international grants. He has received the Australasian Distinguished Doctoral Dissertation Award from the Computing Research and Education Association of Australasia, the UWA Outstanding Young Investigator Award in 2011, the West Australian Early Career Scientist of the Year Award in 2012, and the Vice-Chancellor's Mid-Career Research Award in 2014.