An Efficient Crossing-Line Crowd Counting Algorithm with Two-Stage Detection

Zhenqiu Xiao[1, *], Bin Yang[2] and Desy Tjahjadi[3]

Abstract: Crowd counting is a challenging task in crowded scenes due to heavy occlusions, appearance variations and perspective distortions. Current crowd counting methods typically operate on an image patch level with overlaps, then sum over the patches to get the final count. In this paper we describe a real-time pedestrian counting framework based on a two-stage human detection algorithm. Existing works with overhead cameras is mainly based on visual tracking, and their robustness is rather limited. On the other hand, some works, which focus on improving the performances, are too complicated to be realistic. By adopting a line sampling process, a temporal slice image can be obtained for pedestrian counting without the need for visual tracking. Only ten low level features are extracted from the input image to establish a feature vector. As a result, our algorithm is more efficient and accurate than existing methods. Pedestrians in the temporal slice image are then located by the two-stage detection algorithm, which is largely based on support vector machine and affinity propagation clustering. Moreover, a novel algorithm is proposed to determine the moving directions of pedestrians by comparing the centers of them in two temporal slice images. Extensive experiments reveal that our system achieves satisfaction performances in terms of both robustness and efficiency.

Keywords: Pedestrian counting, crowd counting, temporal slice image, affinity propagation clustering.

1 Introduction

Pedestrian counting is a basic component in computer vision. Counting the number of pedestrians in a region of interest (ROI) is useful for security monitoring and traffic control. Counting the number of pedestrians crossing a line of interest (LOI) also helps to optimize the public services such as providing the sufficient products in a shop or services in public places. However, counting pedestrians in crowded situations is a challenging task.

In this paper, we introduce a novel approach to count the pedestrians across a line of interest (LOI) via a two-stage detection algorithm, which is more robust and efficient than existing methods. Here we segment the pedestrians in the second-stage detection using the affinity propagation (AP) clustering [Frey and Dueck (2007)]. We compare the performances of

[1] School of Computer, Jiaying University, Meizhou, 514015, China.

[2] School of Design, Jiangnan University, Wuxi, 214122, China.

[3] School of Computer Science, Assumption University, Bangkok, 351078, Thailand.

[*] Corresponding Author: Zhenqiu Xiao. Email: Yewind2017@163.com.

different clustering algorithms and of two-stage and one-stage segmentation algorithms. The results reveal that the proposed method is robust and highly accurate. The contributions of our paper are summarized below:

1. A two-stage detection algorithm is developed to count the pedestrians crossed the LOI.
2. An automatic method for estimating pedestrian moving direction is proposed.
3. Compared to other relative methods, the proposed algorithm is more efficiently in pedestrians detecting.

The rest of the paper is organized as follows. In Section 2 we describe some existing algorithms in the literature to set the background for the research. Details of our two-stage detection algorithm is described in Section 3. The results of our algorithm are compared with other baselines in Section 4. A conclusion is finally drawn in Section 5.

2 Related work

Many algorithms have been proposed to count pedestrians and experiments have demonstrated that counting pedestrians in complex scenes is a difficult task. Applications generally involve counting pedestrians in a region of interest (ROI counting) and pedestrians passing through a line of interest in a fixed time period (LOI counting).

Existing methods for ROI counting can be generally divided into two categories: object detection and regression. Viola et al. [Viola, Jones and Snow (2003)] detect motion patterns by utilizing the AdaBoost learning algorithm. However, when occlusion occurs, the motion pattern detection method tends to suffer, and because of these some part-based pedestrian detectors are proposed. Wu et al. [Wu and Nevatia (2005)] count pedestrians in crowded scenes by detecting different parts of the human body, including head and shoulder, torso and legs, etc. in scenes with occlusion. To make the part-based pedestrian detector more robust, Xing et al. [Xing, Ai, Liu et al. (2011)] developed an approach combining head-shoulder detector and moving flow. Dong et al. [Dong, Parameswaran, Ramesh et al. (2007)] match the shape of foreground segment to a database with pedestrian silhouettes. This performs well when the number of pedestrians in the segment is less than [Wu and Nevatia (2005)]. The above methods try to directly detect every single pedestrian in one or several frames precisely. However, when in crowded situations, detecting a single person becomes difficult.

The regression methods can deal with scenes with significant occlusion successfully without locating the precise position of each pedestrian. Texture features should be extracted to count the crowds. To analyze the crowd in small regions, the Local Binary Pattern (LBP) method is utilized to count pedestrians in separated areas [Wang, Liu, Qian et al. (2012)]. In order to estimate the exact number of pedestrians in each of the crowd segments, foreground edge orientations Kong et al. [Kong, Gray and Tao (2006)] are extracted as features. A Gaussian process regression with radial basis function (RBF) kernel is proposed to capture both linear and locally non-linear features [Chan, Liang and Vasconcelos (2008)].

A number of papers in the literature for LOI counting relies on visual tracking [Antic, Letic, Culibrk et al. (2009); Velipasalar, Tian and Hampapur (2006)]. To avoid occlusions, most tracking methods are based on overhead cameras. Zhao et al. [Zhao and Nevatia (2003)] use human shape models to interpret the foreground in a Bayesian framework. However, it is difficult for the system to achieve real-time performance when numerous pedestrians are tracked simultaneously. Therefore, a two-level tracking structure is proposed, which uses a fast blob tracking method and the mean-shift tracking algorithm to deal with merges and splits of the blobs [Velipasalar, Tian and Hampapur (2006)]. In crowded situations, tracking a single person becomes difficult and it is necessary to segment different pedestrians from the foreground blobs. Antic et al. [Antic, Letic, Culibrk et al. (2009)] develop a system that uses k-means clustering to enable the segmentation of single pedestrian in the scene.

However, visual tracking is computationally expensive and tracking multiple targets accurately and robustly is still a challenging problem. What is more, it takes tremendous amount of time to extract the features of pedestrians from every frame in the video and process the extracted features. As a result, researchers have begun to utilize information from spatial and temporal domains. Albiol et al. [Albiol, Mora and Naranjo (2000)] first store the foreground of pedestrians passing through a gate of a train in a spatial-temporal domain and segment the pedestrians with morphological tools. Morphological operations are applied to deal with images that are lack of detailed features, so a Kanade-Lucas-Tomasi (KLT) feature tracker is utilized to gather feature trajectories, which are then clustered to estimate the number of pedestrians [Rabaud and Belongie (2006)]. As the KLT tracker is not efficient, Cong et al. [Cong, Gong, Zhu et al. (2009)] measure the flow velocity on the detection line and estimate the number of pedestrians by quadratic regression. Ma et al. [Ma and Chan (2013)] improve the work of Cong et al by introducing an integer programming method to count people crossing the line based on ROI counting. Ma and Chan [Ma and Chan (2016)] propose an integer programming method for estimating the instantaneous count of pedestrians. The number of people is estimated in a set of overlapping sliding windows on the temporal slice image. Recently, many machine learning-based works had been proposed.

Recently, many machine learning-based works had been proposed. Zhang et al. [Zhang, Zhou, Chen et al. (2016)] use multi-column convolutional neural network (MCNN) architecture to map the image to its crowd density map. The features learned by each column CNN are adaptive to variations in people/head size due to perspective effect or image resolution by utilizing filters with receptive fields of different sizes [Cui, Mcintosh and Sun (2018)]. A deep CNN-based method is proposed by Zhao et al. [Zhao, Li, Zhao et al. (2016)] which can directly estimates the crowd counts with pairs of video frames as inputs and is trained with pixel-level supervision maps. A two-phase training scheme is adopted, which decomposes the original counting problem into two easier sub-problems, estimating crowd density map and estimating crowd velocity map. Although, their methods achieved significant performances, the efficiency was unpleased, and can be improved. To accomplish this, we proposed a two-stage detection algorithm, which is based on SVM and affinity propagation clustering.

3 Proposed algorithm

In this section, we will demonstrate the architecture of the system. We first describe the procedures to obtain the temporal slice image, followed by feature extraction for classification. The two-stage pedestrian detection algorithm is then presented in detail. Finally, we discuss how to determine the moving direction of the pedestrians.

The flow chart of our approach is shown in Fig. 1. First, a cycle period *T* is set manually and Gaussian mixture model (GMM) is applied to subtract the background. Second, the temporal slice image *ST* on the *LOI* is obtained. Temporal slice image is an arrangement of the lines collected frame by frame. Each line contains information about moving pedestrians and its location. It represents the time when the line is sampled. Subsequently, 10 features are extracted from each foreground region in *ST*, and forms a feature vector *F(i)i=1…N*, where *N* is the account of foreground regions. According to these feature vectors, the foreground regions are classified into two categories, crowd and individual. The centers of the regions labelled as individuals can be located immediately. The pedestrians in the crowd class are detected by clustering. Finally, the moving directions of each pedestrian are determined.

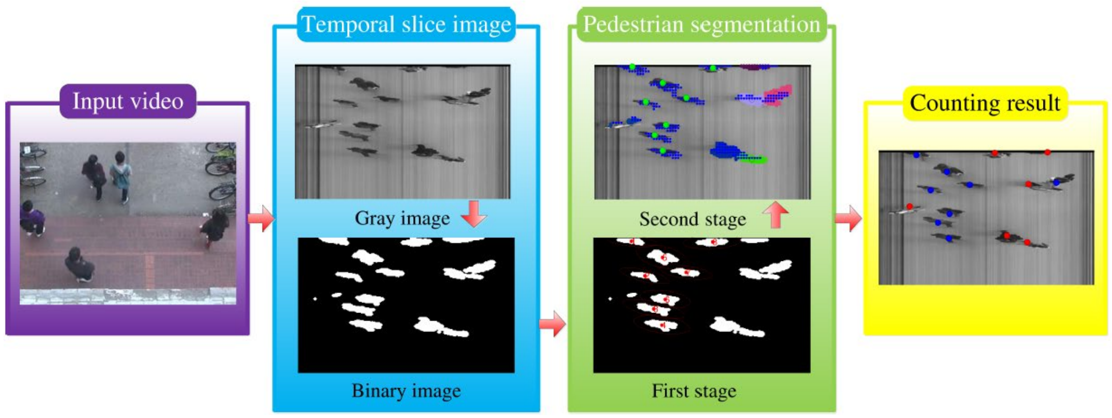


Figure 1: Flow Chart of the Approach

3.1 Background subtraction and line sampling

The proposed method creates a temporal slice image which contains the location of pedestrians and the time when the pedestrians appear. All the frames are transferred to gray scale images before processing. Gaussian mixture model is applied for detecting moving objects. For each pixel on the LOI, *I(p)* denotes the grayscale value of pixel p. The probability that the pixel presents this color at frame t is defined as:

$$P(I(p),t)=\sum_{k=1}^{K}\omega_k(t)g(I(p),\mu_k(t),\sigma_k^2(t)) \tag{1}$$

where *K* is the number of mixture models (usually it is defined as 3), $\omega_k(t)$ is the weight of the kth model at frame $t(\sum_{k=1}^{K}\omega_k(t)=1)$ is the center of the kth model at frame *t*, $\sigma_k^2(t)$ is the variance of the *k*th model at frame *t*, and *g* is a Gaussian function

$$g(I(p),\mu,\sigma^2)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{[I(p)-\mu]^2}{\sigma^2}} \tag{2}$$

Typically, if $|I(p)-\mu_k(t)|<2.5\sigma_k(t)$, we consider that the current pixel *p* at frame *t* matches the *k*th model. On the other hand, if the pixel *p* does not match any of the models, it is labelled as foreground. When the current pixel p matches the *k*th model, the *k*th model will be updated by

$$\begin{cases}\mu_k(t+1) = (1-\alpha)\mu_k(t) + \alpha I(p) \\ \sigma_k^2(t+1) = (1-\alpha)\sigma_k^2(t) + \alpha(\mu_k(t) - I(p))^2\end{cases} \quad (3)$$

Here α is a preset parameter to control the updating speed. Suppose that for all the models that pixel p matches, the $k_m{}^{th}$ model has the largest weight $\omega_{k_m}(t)$. Then this largest weight is updated by

$$\omega'_{k_m}(t+1) = (1-\beta)\omega_{k_m}(t) + \beta \quad (4)$$

Here β is also a preset parameter to control the updating speed. For other models, the weights are updated by

$$\omega'_k(t+1) = (1-\beta)\omega_k(t) \quad (5)$$

Finally, the updated weights are normalized by

$$\omega_k(t+1) = \frac{\omega'_k(t+1)}{\sum_{i=1}^{K} \omega'_k(t+1)} \quad (6)$$

When there is no model that can match the pixel p, the model that has the smallest weight will be replaced by a new model with $\mu_k(t+1) = I(p)$.

After the background subtraction on the LOI, each pixel on the LOI are labelled either white (if it is the foreground) or black (if it is the background). By applying the line sampling procedure on the LOI, the binary temporal slice image *ST* is obtained.

Two LOIs are drawn to obtain the temporal slice images (see Fig. 2(a) for example). The solid line is the main line to collect the information of foreground and the dash one, named associate line, is utilized to help the system to determine the direction of each pedestrian. The gray scale temporal slice image obtained from the main line during the first frame to the *200th* frame is shown in Fig. 2(b). The binary temporal slice images obtained from the two lines are presented in Fig. 2(c) and Fig. 2(d).

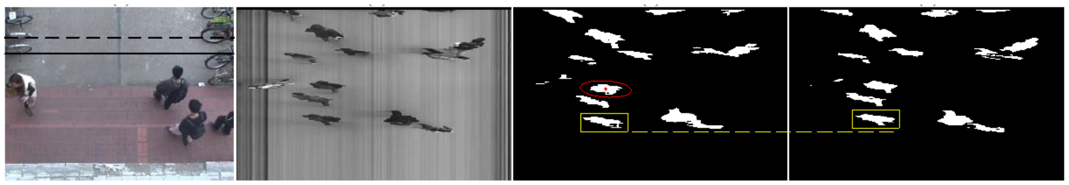


Figure 2: a) LOIs, and temporal slice images from b) gray level, c) the main line and d) the associate line. The red ellipse is an example of covering ellipse in Secyion 3.2. The yellow rectangles show an example of direction determination in Section 3.4

3.2 Feature extraction

Features are extracted from each foreground region in the binary temporal slice image. In this paper, we extract 10 low level features to form a feature vector *F(i)i=1…N*. Each pixel in foreground region has its coordinate $(\omega, t) \in ST$. Let $cov = \Phi\Lambda\Phi^{-1}$ be the covariance matrix of pixel coordinates in a foreground region, here $\Phi = [\nu_1, \nu_2]$ in which ν_1 and ν_2 are the eigenvectors and $\Lambda = diag(\lambda_1,\lambda_2)$ (here $\lambda_1 < \lambda_2$) is the matrix of eigenvalues. Ten features are then extracted from each foreground region:

1. the area size of the foreground region S;

2. the length of long axis of the covering ellipse $a = \sqrt{S \times \lambda_2/\lambda_1}$;
3. the length of minor axis of the covering ellipse $b = \sqrt{S \times \lambda_1/\lambda_2}$;
4. the length of long axis over the length of minor axis a/b;
5. the difference of the time axis Δt;
6. the difference of the line axis Δw;
7. the perimeter of the foreground region L;
8. the variance of the time axis σ_t^2;
9. the variance of the line axis σ_w^2;
10. the perimeter over the area size L/S.

One example of the covering ellipse is shown in Fig. 2©. After feature extraction, *N* feature vectors *F(i)i=1…N* is formed according to *N* foreground regions in the temporal slice image *ST*.

3.3 Two-Stage pedestrian detection

When the pedestrians are in a crowd, it is difficult to identify each individual from the temporal slice image *ST*. We propose a two-stage pedestrian detection algorithm. First the foreground regions from the temporal slice image are separated into two classes, that containing only one person and that containing more than one person, using a support vector machine classifier. Then a clustering algorithm is utilized to estimate the number of pedestrians in the crowded class.

In the first stage, support vector machine is applied to classify the foreground regions according to the extracted features *F(i)*. Support vector machine (SVM) is widely used in two-class classification problems. C-Support Vector Classification (C-SVC) [Chang and Lin (2011)] is chosen in our algorithm. In detail, given training vectors $xt_i \in R^n$, $i = 1, ..., l$, in two classes and a label vector $y \in R^l$ such that $y_i \in \{1, -1\}$, C-SVC solves the following optimization problem:

$$\min_{\omega,b,\varepsilon} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\varepsilon_i \quad (7)$$

subject to $y_i(\omega^T\emptyset(xt_i) + b) \geq 1 - \varepsilon_i$, $\varepsilon_i \geq 0, i = 1, \ldots, l,$

where $\emptyset(xt_i)$ maps xt_i into a higher-dimensional space and $C > 0$ is the cost. Since the dimension of the vector variable ω is possibly high, the problem is usually solved in the following dual form:

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \quad (8)$$

subject to $y^T\alpha = 0$, $0 \leq \alpha_i \leq C, i = 1, \ldots, l,$

where $e = [1, ..., 1]^T$ is a vector of ones, Q is an l by l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(xt_i, xt_j)$, and $K(xt_i, xt_j) \equiv \emptyset(xt_i)^T\emptyset(xt_y)$ is the kernel function. We choose the radial basis function (RBF) as the kernel function in our algorithm, which takes the form as:

$$K(xt_i, xt_j) = e^{-\tau\|xt_i, xt_j\|^2} \quad (9)$$

The parameter τ is related to the number of features, and will be discussed (together with cost *C*) later in the parameter setting section. After solving problem (8), using the primal-dual relationship, the optimal ω is

$$\omega = \sum_{i=1}^{l} y_i \alpha_i \phi(xt_i) \tag{10}$$

and the decision function is:

$$sgn(\omega^T \phi(xt) + b) = sgn(\sum_{i=1}^{l} y_i \alpha_i K(xt_i, xt) + b) \tag{11}$$

and parameter *b* in Eq. (11) is

$$b = -\frac{1}{2}(\max_{i:y_i=-1} \omega^T xt_i + \min_{i:y_i=1} \omega^T xt_i) \tag{12}$$

In Fig. 3(a), the foreground areas in the ellipses represent the individuals classified by the SVM classifier.

In the second stage, we apply affinity propagation (AP) clustering to detect each pedestrian. The AP clustering algorithm is an iteration of responsibility and availability. First the similarity is obtained:

$$s(i,k) = \begin{cases} -\|x_i, x_k\|^2 & if\ i \neq k \\ \frac{\sum_{i'=1}^{N_\tau} \sum_{k'=1,k'\neq i'}^{N_\tau} s(i',k')}{N_\tau(N_\tau - 1)} & if\ i = k \end{cases}, \tag{13}$$

where x_i and x_k $(1 \leq i,\ k \leq N\tau)$ are the coordinates of foreground region in the crowd class, and $N\tau$ is the total number of pixels of all the foreground regions in the crowd class. Then the responsibility $\tau(i,k)$ and availability $\alpha(i,k)$ are iterated by:

$$\tau_{t+1}(i,k) = s(i,k) - \max_{k' s.t.k' \neq k} \{\alpha_t(i,k') + s(i,k')\}\ , \tag{14}$$

$$\alpha_{t+1}(i,k) = \min\left\{0, \tau_{t+1}(k,k) + \sum_{i' s.t.i' \notin (i,k)} \max\{0, \tau_{t+1}(i',k)\}\right\} \tag{15}$$

And a damping procedure is applied to reduce oscillations:

$$\tau_{t+1}(i,k) = (1-\lambda) \cdot \tau_{t+1}(i,k) + \lambda \cdot \tau_t(i,k) \tag{16}$$

$$\alpha_{t+1}(i,k) = (1-\lambda) \cdot \alpha_{t+1}(i,k) + \lambda \cdot \alpha_t(i,k) \tag{17}$$

Here λ is a preset parameter to control the damping speed. The pixel points which fulfill $\tau(k,k) + \alpha(k,k) > 0$ are regarded as centers of clusters.

We compared 3 different kinds of clustering algorithms: mean shift clustering [Vella, Infantino and Scardino (2017)], k-means clustering and AP clustering. The results of the three clustering algorithms are presented in Fig. 3(b) to Fig. 3(d). The first row in Fig. 3 is a comparison based on the original temporal slice image while the second row is based on the crowd class classified by the SVM classifier. Comparing the clustering results between the top and bottom row, we can conclude that the first stage detection contributes to reducing the error of clustering single pedestrians. What is more, no more parameters in AP clustering algorithm is required whilst the number of clusters in k-means clustering and the bandwidth in mean shift clustering need to be estimated before clustering.

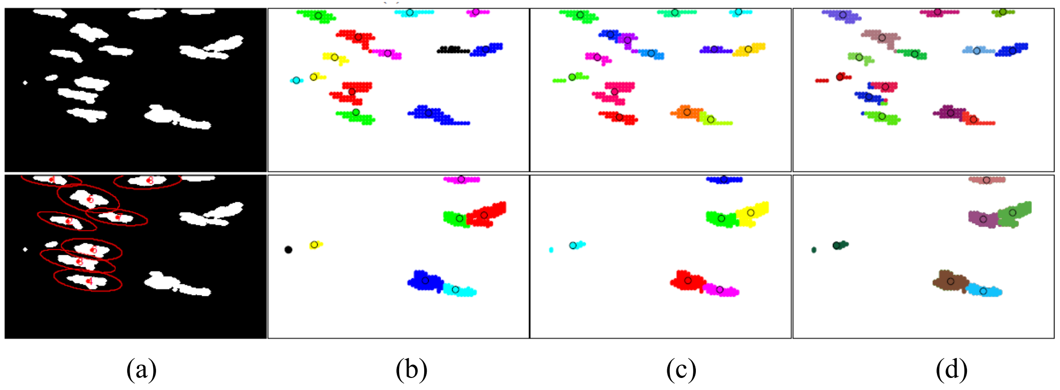


Figure 3: a) Binary temporal slice image and results of b) Mean shift clustering, c) k-means clustering and d) Affinity propagation clustering

3.4 Direction determination

Our approach uses a novel method to determine the moving directions of pedestrians by comparing the centers of them in two temporal slice images. However, the centers obtained for the two images by clustering may not match, as the number of clusters cannot be predicted. Therefore, the number of clusters is set as the result of clustering in the first temporal slice image. The centers of pedestrians in the second temporal slice image are obtained by labelling the foreground pixel to the nearest center in the first image, and new centers in the second image can be obtained according to the labelled pixels. The centers in the two images can then be compared.

When a pedestrian moves, the center of the person passes two lines at different time. By analyzing the centers from two temporal slice images, we can determine the moving directions of the pedestrian. Let $ST_c(\omega_m, t_m)$ be the central position of a pedestrian on the main line and $ST_c(\omega_\alpha, t_\alpha)$ of the associate one. Then the direction of the pedestrian can be decided by:

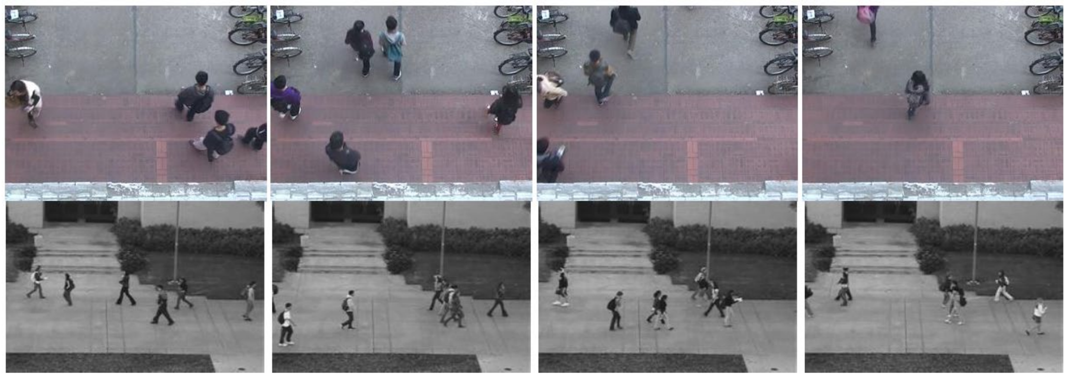
$$\text{direction} = \begin{cases} down & if \;\; t_m - t_a > 0 \\ up & if \;\; t_m - t_a < 0 \end{cases}, \tag{18}$$

An example of direction determination is demonstrated in Figs. 2(c) and 2(d). Comparing the centers of the man in the rectangles, we can find that $t_m - t_a > 0$, so the man is going downside.

4 Experiments and results

4.1 Datasets

Most of the testing videos of LOI counting in the existing literatures are not accessible for us, so it is not easy to compare the performances of different methods. Thus, we established a new pedestrian dataset, named the PUCSD dataset. We evaluate the performance of our system on two large pedestrian datasets, the PUCSD dataset and the UCSD dataset. Some representative frames from these datasets are shown in Fig. 4. The PUCSD dataset is created by setting up a camera in a building on the campus. 8 video clips were collected with a frame size of 320 × 240 and frame rate of 18 fps. The PUCSD dataset contains more than 10000 frames and the ‘vidd scene’ from the UCSD dataset contains about 34400 frames. The ground truth of the UCSD dataset is related to ROI counting, thus we marked the ground truth of the UCSD dataset for LOI counting. The line of interest (LOI) is vertically located 60 pixels from the left side.



(a) (b) (c) (d)

Figure 4: Representative frames. Top: the PUCSD dataset. Bottom: the UCSD dataset

4.2 Training and parametric settings

As postures of pedestrians may vary differently on different line locations, 24 locations of lines are simultaneously chosen at random to obtain the temporal slice images for training. The foreground regions in these training images are the pedestrians to be counted. There are two principles to set the cycle period *T*:

1. The body of a pedestrian is required to pass the line during the cycle period, so *T* cannot be too small.
2. The counting results of a cycle should be presented in time, so T cannot be set overly large. Taking the above principles into account, we set the cycle period T to 200 frames.

The locations of the main line and the associate line are related to the moving directions of the pedestrians. Firstly, both lines should be located perpendicular to the direction of the flow. Secondly, the distance between the two lines is set to half the width (if the lines are vertical) or half the height (if the lines are horizonal) of the pedestrians. Thus, the temporal slice images from the two lines are similar but have minor differences. If the distance of two lines is too large, the foreground regions from the temporal slice images will be quite different, and it will not be easy to match the corresponding centers of the pedestrians. On the other hand, if the two lines are too close, the differences of the centers cannot be recognized easily.

In Section 3.1, we introduced two parameters α and β in the updating procedure of Gaussian mixture model (GMM). Large α and β lead to fast updating speed, as well as instability. In order to keep the updating procedure robust, the parameters α and β are both set to 0.02 in this paper.

LIBSVM toolbox [Chang and Lin (2011)] is applied to train and classify the foreground regions in the first-stage detection in Section 3.3. The parameter γ (in Eq. (9)) is usually set to $1/K_f$, where K_f is the number of features ($K_f = 10$ in this paper). The parameter C (appeared in Eq. (7)) is generally set to 1. Meanwhile, in order to estimate the exact value of γ and C, cross validation is applied, instead of simply taking the empirical values. What is more, the maximum value of the features cannot be predicted, so normalization and cross validation are employed in each cycle.

In the damping procedure in AP clustering, we introduced the parameter λ. A quantized intermediate value named Net Similarity is calculated to monitor the oscillations, which is defined as:

$$\text{Net Similarity} = dpsim + expref \tag{19}$$

The sum of the similarities of the data points to their cluster centers is defined as *dpsim*, while the sum of the similarities of the cluster centers themselves (define in Eq. (14) when $i = k$) is returned as *expref*. The Net Similarity will reach a maximum value as the clustering algorithm iterates. Fig. 5 shows the Net Similarity with different values of λ, from which we can draw the conclusion that a large λ (see $\lambda \geq 0.9$ as examples) increases the number of iterations with a stable rise of Net Similarity. On the other hand, a small λ (see $\lambda = 0.6$ as an example) accelerates

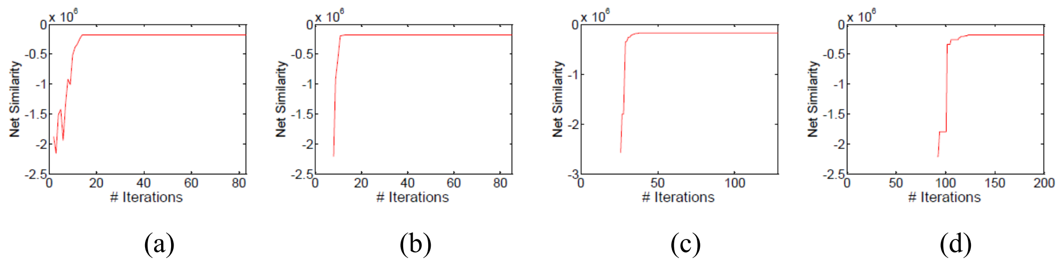


Figure 5: Parameter setting in AP clustering. a) λ = 0:6, b) λ = 0:7, c) λ = 0:9, d) λ = 0:97.

the iteration, but causes oscillations of Net Similarity. Taking both stability and efficiency into account, we set λ to 0.7 in the proposed algorithm

4.3 Experimental results

The results on the PUCSD dataset are shown in Tab.1. The error rate of the total number regardless of the moving directions is 6.573%. The error rate for the number of pedestrians walking up is 7.633%, whilst that for walking down is 8.056%.

Table 1: Result of the PUCSD dataset. The second and third columns are the actual number of pedestrians up and down

Video No.	#of Up	#of Down	Up	Down	Err Up	Err Down	Err Total
1	15	3	15	3	0	0	0
2	12	7	13	8	+1	+1	+2
3	16	11	16	12	0	+1	+1
4	28	8	29	7	+1	-1	0

5	37	11	35	11	-2	0	-2
6	16	7	17	9	+1	+2	+3
7	13	10	13	10	0	0	0
8	8	5	11	5	+3	0	+3

In Fig. 6 we compare the performances of the tracking-based counting,16 the proposed algorithm with and without the first-stage detection (abbreviated as ’cluster’ in Fig. 6) on the UCSD dataset. By comparing the performance of our system with the ground truth and the tracking-based method, we conclude that the accuracy of our system is acceptable. We can also state that the two-stage detection algorithm performs better than the algorithm just applies clustering. The errors are mainly caused by three reasons:

1. When someone stays on the line for a while, the number of clusters will increase, as well as the result of counting.
2. The result of background subtraction is affected when the color of pedestrian is similar to the ground.
3. Large moving objects such as bikes also lead to errors.

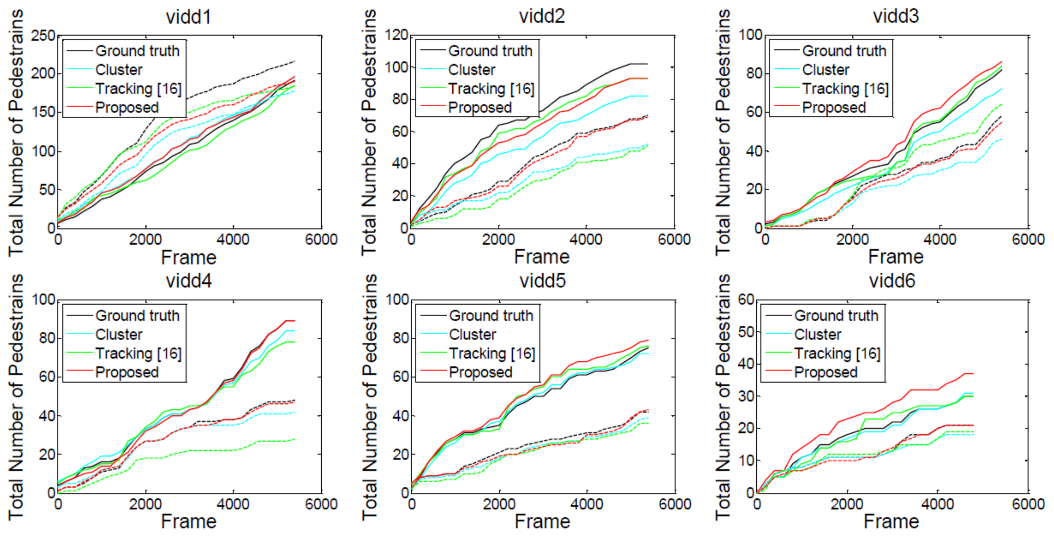


Figure 6: Results on the UCSD dataset. The solid lines represent the left direction while the dash lines represent the right direction

The algorithm is coded in MATLAB 2016b on a machine equipped with Intel i7 3.4 GHz CPU with 16 GB RAM. Running time of each step is shown in Tab. 2. The frame rate of the test video is 18 fps, so the cycle is 3.31 seconds. The counting procedure can be completed before the temporal slice image of the next cycle is obtained, thus our system can operate in real-time.

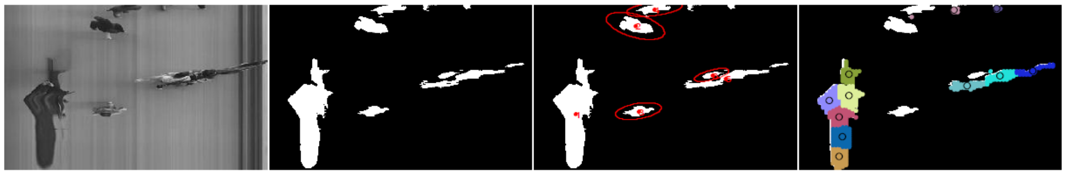
\

Table 2: Processing time of each step

Task	Feature Extraction	SVM	AP Clustering	Direction Judging	Total
Cost	0.07	1.48	1.72	0.04	3.31

4.4 Failure case

Although the proposed two-stage detection algorithm achieves satisfying results in most instances, it is expected to fail in some extreme situations. For instance, in Fig. 7, if one person stays on the LOI for a while, the foreground region in the temporal slice image will increase (see Fig. 7(a) and Fig. 7(b)). After the first-stage classification (see Fig. 7(c)) and the second-stage clustering (see Fig. 6(d)), the foreground region of the stationary person (labelled 1 in Fig. 7(c)) will be grouped into small clusters. As a result, the number of pedestrians estimated by the AP clustering algorithm will increase (see Fig. 7(d)). In this case, the visual tracking based counting methods will probably perform better than the proposed algorithm. Another cause of failure is due to background subtraction error. Specifically, if the color of the moving object is similar to the background, the result of background subtraction will not be accurate. The foreground region labelled 6 in Fig. 7(c) is an example of this. This region should be connected to that labelled 5. It is therefore necessary to select suitable background subtraction algorithms for different situations.



(a) (b) (c) (d)

Figure 7: Failure case. a) Gray level and b) Binary temporal slice images. c) Results of SVM and d) Clustering

5 Conclusion

In this paper, a novel two-stage pedestrian detection algorithm is proposed to count pedestrians. Without tracking each pedestrian, our counting algorithm is both efficient and robust. The LOI counting is considered as classification and clustering. The two-stage pedestrian detection algorithm locates each pedestrian in the temporal slice image accurately. Experiments demonstrate the reliability of our algorithm and suggest that our LOI counting can efficiently dealt with crowded situations. However, the stationary pedestrians still cause counting errors. In future work, we will explore how to deal with this, as well as large moving non-human objects.

Acknowledgments: This research is partly supported by National Science Foundation, China (Nos. 51505191, U1536206, 61602253); the Fundamental Research Funds for the Central Universities (Nos. JUSRP11534, JUSRP51642A).

References

Albiol, A.; Mora, I.; Naranjo, V. (2000): Real-time high density people counter using morphological tools. *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 4, pp. 204-218.

Antic, B.; Letic, D.; Culibrk, D.; Crnojevic, V. (2009): K-means based segmentation for real-time zenithal people counting. *IEEE International Conference on Image Processing*, pp. 2565-2568.

Chan, A. B.; Liang, Z. S. J.; Vasconcelos, N. (2008): Privacy preserving crowd monitoring: Counting people without people models or tracking. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-7.

Chang, C. C.; Lin, C. J. (2011): *LIBSVM: A Library for Support Vector Machines*. ACM Press.

Cong, Y.; Gong, H.; Zhu, S. C.; Tang, Y. (2009): Flow mosaicking: real-time pedestrian counting without scene-specific learning. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1093-1100.

Cui, Q.; Mcintosh, S.; Sun, H. (2018): Identifying materials of photographic images and photorealistic computer generated graphics based on deep CNNs. *Computers, Materials & Continua*, vol. 55, no. 2, pp. 229-241.

Dong, L.; Parameswaran, V.; Ramesh, V.; Zoghlami, I. (2007): Fast crowd segmentation using shape indexing. *IEEE International Conference on Computer Vision*, pp. 1-8.

Frey, B. J.; Dueck, D. (2007): Clustering by passing messages between data points. *Science*, vol. 315, no. 5814, pp. 972-976.

Kong, D.; Gray, D.; Tao, H. (2006): A viewpoint invariant approach for crowd counting. *International Conference on Pattern Recognition*, pp. 1187-1190.

Ma, Z.; Chan, A. B. (2013): Crossing the line: crowd counting by integer programming with local features. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2539-2546.

Ma, Z.; Chan, A. B. (2016): *Counting People Crossing a Line Using Integer Programming and Local Features.* IEEE Press.

Rabaud, V.; Belongie, S. (2006): Counting crowded moving objects. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 705-711.

Velipasalar, S.; Tian, Y. L.; Hampapur, A. (2006): Automatic counting of interacting people by using a single uncalibrated camera. *IEEE International Conference on Multimedia and Expo*, pp. 1265-1268.

Vella, F.; Infantino, I.; Scardino, G. (2017): Person identification through entropy oriented mean shift clustering of human gaze patterns. *Multimedia Tools & Applications*, vol. 76, no. 2, pp. 1-25.

Viola, P.; Jones, M. J.; Snow, D. (2003): Detecting pedestrians using patterns of motion and appearance. *IEEE International Conference on Computer Vision*, pp. 734-741.

Wang, Z.; Liu, H.; Qian, Y.; Xu, T. (2012): Crowd density estimation based on local binary pattern co-occurrence matrix. *IEEE International Conference on Multimedia and Expo Workshops*, pp. 372-377.

Wu, B.; Nevatia, R. (2005): Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors. *IEEE International Conference on Computer Vision*, pp. 90-97.

Xing, J.; Ai, H.; Liu, L.; Lao, S. (2011): Robust crowd counting using detection flow. *IEEE International Conference on Image Processing*, pp. 2061-2064.

Zhang, Y.; Zhou, D.; Chen, S.; Gao, S.; Yi, M. (2016): Single-image crowd counting via multi-column convolutional neural network. *IEEE Conference on Computer Vision & Pattern Recognition*, pp. 589-597.

Zhao, T.; Nevatia, R. (2003): Bayesian human segmentation in crowded situations. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 459-466.

Zhao, Z.; Li, H.; Zhao, R.; Wang, X. (2016): Crossing-line crowd counting with two-phase deep neural networks. *European Conference on Computer Vision*, pp. 712-726.