Paper:

A Memory Matching Algorithm Based on Box IoU Features for Pedestrian Flow Counting

Junlin Zhang*[1], Kaoru Hirota[1], Yaping Dai[1], and Zhiyang Jia[1]

[1]School of Automation, Beijing Institute of Technology, Beijing (100081), China
E-mail: 3120200983@bit.edu.cn

Abstract. Automatic pedestrian flow counting on street view surveillance video is faced with challenges such as occlusion, illumination changes and scale changes. These challenges lead to misdetection which affect the accuracy of pedestrian flow counting. Aiming at these problems, a memory matching algorithm based on box Intersection over Union (IoU) features of inter frame bounding box is proposed. By memorizing the box IoU features between frames, the algorithm can ignore the false detected pedestrians and track the lost pedestrians to a certain extent, so as to improve the accuracy of counting. The experimental results on the UCSD data set show that the algorithm has high accuracy which reaches 97.2%.

Keywords: Pedestrian Flow Counting, Data Association, Neural Network, Pedestrian Detection

1. INTRODUCTION

Pedestrian flow statistics is important for cities to evaluate the needs of active modes, such as new infrastructure, and the impacts of policies and investments[1, 2]. People have been looking for efficient technologies to collect statistics on non-motorized facilities[3]. There are many technologies to collect pedestrian flow statistics such as artificial counting, infrared sensor counting, gate counting and counting based on stereo camera[4, 5]. Extra installation and hiring costs are required using these pedestrian flow counting methods. However, using existing surveillance systems with video based counting technologies can avoid such costs. In recent years, with the development of computer hardware, the feasibility of video pedestrian flow counting technology has been improved and has become a research hotspot. The video pedestrian flow counting algorithm is consist of pedestrian detection, feature extraction, pedestrian ID matching and pedestrian counting. Among them feature extraction and pedestrian ID matching are the key steps. The methods of feature extraction include manual extraction[6–8] and neural network extraction[9–11], and pedestrian ID matching algorithm includes Hungarian matching algorithm[12, 13], kernel correlation filtering[14, 15]. Among pedestrian flow counting application scenarios street view surveillance is one of the key scenarios. However the application of pedestrian flow counting in street view surveillance system often faces many challenges, such as diverse scenes, limited CPU computing power.

In the street view pedestrian flow counting task, a well performed statistical algorithm relies on an efficient pedestrian detection algorithm. Recently, the pedestrian detection algorithm based on deep learning is often used such as YOLOv3[16]. After the pedestrians are detected, researchers manually extract the Histogram of Oriented Gradient (HOG) features of pedestrians[9, 17], or extract the pedestrian trajectory features by Kalman filter[13]. However, including extracting HOG features, the algorithm after pedestrian detection has additional performance overhead, which takes up valuable system resources in the processing system with insufficient computing resources. After obtaining the effective features, researchers use data association algorithms like Hungarian algorithm[13], kernel function filter[15] to track pedestrians and realize pedestrian ID matching. However, due to the challenges of illumination changes, occlusions and scale changes, the bounding box of the pedestrian detection algorithm is flicker. Using Hungarian algorithm to associate the same pedestrian with flicking box results in repeat count.

In order to reduce additional performance overhead, Box IoU feature is proposed. To improve the accuracy of street view pedestrian flow counting, A memory matching algorithm based on box IoU features is proposed. Street view surveillance video is taken as input of the algorithm. YOLOv3 is taken as the detection part. Extracting the box IoU features by calculating the IoU of the inter frame bounding box is taken as the feature extraction part, and memory matching algorithm is taken as pedestrian ID matching part. To evaluate the accuracy of proposed algorithm ,the manual counting results on UCSD Pedestrian Datasetis regarded as ground truth, and an ablation experiment is set by comparing our memory matching algorithm with Hungarian algorithm under the same conditions. The experimental results show that our algorithm has high accuracy in the counting task, and the counting accuracy reaches 97.2%.

The rest of this paper is organized as follows. In Section 2, related work is briefly introduced. In Section 3, the proposed algorithm is introduced in detail. Then experiments on UCSD Pedestrian Dataset are carried out in Section 4, and the conclusions are shown in Section 5.

2. RELATED WORK

Pedestrian flow counting algorithms can be divided into four parts, pedestrian detection, feature extraction, pedestrian ID matching and pedestrian counting.

Pedestrian detection algorithms take image as input and return the bounding box of pedestrians in the input image. The accuracy of pedestrian detection algorithms based on convolutional neural network (CNN) have reached an unprecedented height[18]. Detection algorithms based on CNN can be divided into two-stage algorithms and one-stage algorithms. YOLOv3, as a typical one-stage algorithms of CNN based detection algorithms which achieves good accuracy and real-time performance.

Feature extraction is to describe the ID of each individual pedestrian. HOG is a commonly used descriptor in computer vision. It constitutes the features by calculating and counting the gradient direction histograms of the local regions of the image. The time complexity of calculating HOG is shown in (1).

$$T_{HOG}(\theta) = O(NHW), \quad (1)$$

where θ represents the parameters of HOG. N represents the number of detected pedestrians to be calculated. H represents the vertical size of detected area with one pedestrian. W represents the horizontal size of detected area with one pedestrian.

Besides, Kalman filtering algorithm is also used in pedestrian flow counting as a feature extraction algorithm. By setting bounding box as It extracts motion state feature of pedestrians. The time complexity of calculating Kalman filtering algorithm is shown in (2).

$$T_{KM}(\theta) = O(N^3 s^3 + N^3 o^3), \quad (2)$$

where θ represents the parameters of Kalman filtering algorithm. N represents the number of detected pedestrians to be calculated. s represents the number of state quantities. o represents the number of observation quantities.

Pedestrian ID matching follows feature extraction. It is used to match the same pedestrian in different frames. Hungarian algorithm is a widely used matching algorithm which is also known as Kuhn-Munkres algorithm. The time complexity of the original Hungarian algorithm was $O(n^4)$[19]. However, years later, researches noticed that it can be modified to achieve an $O(n^3)$ run-time[20, 21]. Hungarian algorithm is an algorithm to find the maximum matching of bipartite graph with augmented path.

The main idea of pedestrian counting algorithm is to count the pedestrians passing through a specific area. For example, Kryjak proposed a counting algorithm based on virtual lines[22] which counts the pedestrian whose center pass the virtual lines.

3. METHODOLOGY

This section presents the proposed algorithm for pedestrian flow counting. The flowchart of the algorithm when processing a single frame is shown in Fig. 1.

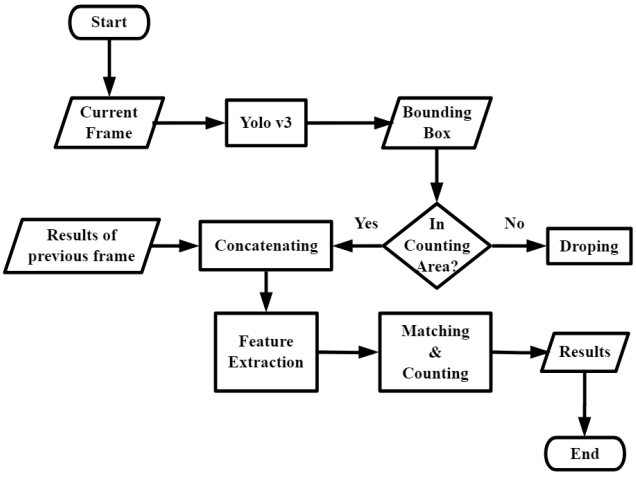


Fig. 1. Flowchart of the proposed algorithm when processing a single frame.

3.1. Algorithm Processes

Before performing the calculation, the detection area needs to be selected first. Since the pedestrians farther away from camera occupies a smaller amount in image, the output of pedestrian detection algorithm has large fluctuations with them. In order to increase the detection accuracy, the detection area is set closer to camera.

In proposed algorithm, the set of bounding boxes is recorded until the current frame of input video which is represented by Bs_n where n means the order of current frame. Similarly, the set of detection frames are recorded until the previous frame is represented by Bs_{n-1}. Both of Bs_n and Bs_{n-1} have the same structure. There are two parts in Bs_n, one is the set of bounding boxes in detection area, which is called Bsd_n, and the other is the set of counted bounding boxes, which is called Bsr_n. Each bounding box contains seven parameters: the center coordinates x and y of the box, the horizontal pixels of the box w and vertical pixels h of the box, the flag parameter $flag$, the classification parameter $class$, and the confidence parameter $conf$. Specific implementation of the proposed pedestrian flow counting algorithm is as follows:

Step 1. Using pedestrian detection algorithm to get the person bounding box in current frame. Then selecting bounding boxes which have two features. First, the center is within the detection area. Second the confidence parameter $conf$ is above the specified parameter thr_{score}. Then setting the $flag$ parameter of these bounding boxes to zero. The set of bounding boxes obtained through the processing above is recorded as Bsd_n.

Step 2. Calculating the IoU of each box in Bs_{n-1} to all boxes in Bsd_n, and recording the results as IoU_{n-1}. Teating the pair of boxes whose IoU in IoU_{n-1} are greater than thr_{iou} as the same pedestrian's bounding box in two frames. The expression of IoU is shown in (3). thr_{iou} is the same as the parameter used in the non-maximal suppression of pedestrian detection algorithm. Treating the

boxes in Bsd_n whose IoU in IoU_{n-1} are all less than thr_{iou} as a newly entered pedestrian in the detection area, and adding one to the pedestrian flow counter for each such box.

$$IoU = \frac{TP}{FP+FN+TP}, \quad (3)$$

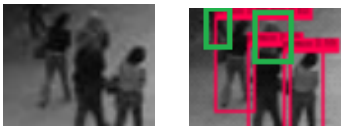
where TP represents the overlap of two bounding boxes, $TP+FP+TP$ represents the union of two bounding boxes.

Step 3. Adding one to the flag parameter of the boxes in Bs_{n-1} whose IoU in IoU_{n-1} are all less than thr_{iou}. These boxes are considered as having passed the detection area or temporarily exiting the detection area due to detection fluctuations. Recording the set of these boxes as Bsr_n, and adding one to their flag parameter. Then check the flag of each box in Bsr_n. Considering the boxes whose flag is greater than the specified parameter thr_flag as having passed the detection area. Then deleting these boxes from Bsd_n.

Step 4. Combining Bsd_n and Bsr_n into Bs_n, and then repeating from step 1 to detect the next frame.

3.2. Improvement of Proposed Algorithm

Due to challenges of occlusions, illumination changes, scale changes of surveillance video, the pedestrian detection algorithm may produce irregular results. For example, Since the occlusion rate among pedestrians is constantly changing when the crowd is moving, the blocked pedestrian is detected discontinuously, and the bounding box is deformed as shown in Fig. 2.



(a) (b)

Fig. 2. Irregular case of bounding box. (a)original picture, (b) detecting result with deformed bounding box. The deformed bounding box is represented by green rectangles.

In order to make the algorithm adapt to the irregular case, a judgment standard based on the change of the center trajectory of the bounding box on the basis of the original algorithm is added. The standard is shown in (4).

$$\Delta C > \frac{\overline{B}_W}{2}, \quad (4)$$

where ΔC represents the nearest distance from the center of boxes in $\bar{B}s_{n-1}$ to a box in Bs_n, $\bar{B}w$ represents the average width of bounding boxes in the video.

It indicates that the boxes which are recorded in Bsr_n in Step 3 need to additionally satisfy (4). At the same time, to reduce the influence of the misdetection, the bounding box appearing only one frame is dropped.

3.3. Time Complexity Analysis

Time complexity of our entire algorithm mentioned in Section 3.2 is shown in (5).

$$T_{OUR}(\theta) = O(N^2), \quad (5)$$

where θ means the parameters of our algorithm. N means the number of detected pedestrians to be calculated. In pedestrian flow counting task, time complexity of Hungarian algorithm is described by the following formula:

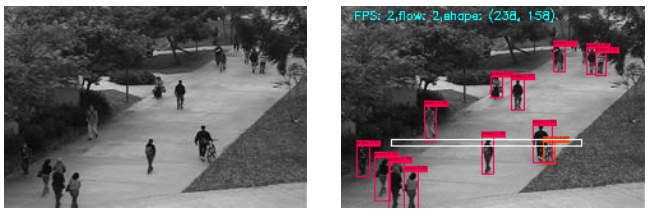
$$T_H(\theta) = O(N^3), \quad (6)$$

where θ means the parameters of Hungarian algorithm. N means the number of detected pedestrians to be calculated.

Comparing (5) with (1), (2) and (6), it can be noticed that our algorithm has lower run-time. Because N is far less than $H*W$ in most cases, n and m are greater than one,

4. EXPERIMENT

Due to lack of street view pedestrian flow counting dataset, the proposed algorithm is tested on UCSD Pedestrian Dataset [16]. The dataset contains video of pedestrians on UCSD walkways, taken from a stationary camera. There are currently two viewpoints available. The part of dataset with front-down scene is selected as experimental dataset. All videos are 8-bit grayscale, with dimensions 238 × 158 at 10 fps. Sample video frame of UCSD Pedestrian Datasets is shown in Fig. 3. Manually counting result of the pedestrian flow in the video is taken as ground truth.



(a) (b)

Fig. 3. Sample frame of UCSD Pedestrian Dataset. a) original frame of UCSD Pedestrian Dataset. b) output of proposed algorithm, where the white box represents the detection area, the red box represents detected pedestrian.

The flow of pedestrian flow, which means the number of pedestrians passing a reference point per unit of time, is 2844 pedestrians per hour. As a reference, the flow of pedestrian flow in [14] is 365 pedestrians per hour and 891 pedestrians per hour. It indicates that the volume of pedestrians in UCSD Pedestrian Dataset is high enough to test a pedestrian flow counting algorithm.

To evaluate the performance of pedestrian flow counting algorithm, every 20 seconds video in dataset is taken as a clip. The part of program counts which is greater

than manual counts in a clip is regarded as false negative sample. The part of program counts which is less than manual counts in a clip is regarded as false positive sample and the rest part of program counts is regarded as true positive sample. Experimental results of our algorithm is shown in Table 1.

Table 1 Results of our algorithm tested on UCSD dataset.

Video Number	Program Counts	Manual Counts	Aggregated Error
000	13	12	8.33%
001	16	16	0.00%
002	11	13	15.38%
003	11	11	0.00%
004	15	17	11.76%
005	16	15	6.67%
006	21	21	0.00%
007	26	26	0.00%
008	16	16	0.00%
009	16	16	0.00%
010	13	13	0.00%
011	23	21	9.52%
012	16	15	6.67%
013	16	16	0.00%
014	11	13	15.38%
015	13	15	13.33%
016	22	23	4.35%
017	6	6	0.00%
018	21	21	0.00%
019	10	10	0.00%
Program Counts(total)		312	
Manual Counts(total)		316	
Accuracy		95.6%	
Precision		97.2%	
Aggregated Error		1.3%	

The accuracy is calculated by dividing the number of true positive sample by the sum of manual counts and the number of false negative sample. The precision is the ratio of true positive sample number to the manual counts. The aggregated error is the ratio of the difference between program counts and manual counts to manual counts.

The experimental results show the high performance of our algorithm in street view pedestrian flow counting task.

As a contrast, by replacing our memory matching algorithm with Hungarian algorithm, the ablation experiment is carried out. The experimental results are shown in Table 2.

The experimental results show that the counting precision drops significantly. This is due to some exceptions. First, Due to the leak detection problem of pedestrian detection algorithm, the pedestrian will be lost in some frames of video. Second, due to mutual occlusion between pedestrians, some pedestrians in the video frames cannot be fully displayed. The first exception causes the pedestrian to be lost in one frame and be detected in the next frame. Since the Hungarian algorithm is a maximum matching algorithm, the redetected pedestrian is regarded as a new pedestrian, witch results in repeated counting. The second exception will lead to rapid and irregular changes of x, y, w and h of the box. However, since the Hungarian algorithm matches with reference to the above parameters, it has the risk of wrong matching when x, y, w and h of the box changes rapidly and irregularly.

Table 2 Results of replacing memory matching algorithm with Hungarian algorithm and testing on UCSD dataset.

Video Number	Program Counts	Manual Counts	Aggregated Error
000	14	12	16.67%
001	20	16	25.00%
002	17	13	30.77%
003	19	11	72.73%
004	24	17	41.18%
005	22	15	46.67%
006	29	21	38.10%
007	39	26	50.00%
008	25	16	56.25%
009	22	16	37.50%
010	21	13	61.54%
011	39	21	85.71%
012	31	15	106.67%
013	18	16	12.50%
014	18	13	38.46%
015	25	15	66.67%
016	36	23	56.52%
017	8	6	33.33%
018	29	21	38.10%
019	13	10	30.00%
Program Counts(total)		469	
Manual Counts(total)		316	
Accuracy		67.4%	
Precision		100%	
Aggregated Error		48.4%	

Then, in order to further verify the effectiveness of IOU feature in pedestrian flow counting task,The Kalman, Hog and IOU feature is used to conduct comparative experiments with Hungarian matching algorithm and our memory matching algorithm respectively. Results are shown in Table 3. The experimental results show that the exceptions seriously affect the result of pedestrian flow counting algorithm. By memorizing the box IoU features between frames, our memory matching algorithm using Box IoU Features can effectively deal with the exceptions witch are mentioned above.

Table 3 Results of comparative experiments.

Matching Method	Feature	Aggregated Error	Accuracy	Precision
Hungarian	Kalman	35.8%	67.8%	95.3%
	HOG	3.8%	77.4%	88.9%
	IOU	48.4%	67.4%	**100.0%**
Memory	Kalman	2.5%	91.6%	96.8%
	HOG	**0.6%**	84.8%	92.1%
	IOU	1.3%	**95.6%**	97.2%

5. CONCLUSIONS

A memory matching algorithm based on box IoU features is proposed in this work. The performance of proposed algorithm is evaluated on UCSD Pedestrian Dataset by comparing algorithm counts and manual counts from video data. The walkways in UCSD Pedestrian Dataset has high pedestrian flows of 2844 pedestrians per hour with pedestrians moving in dense group. Experimental results show that the proposed algorithm achieves accuracy of 95.6%, precision of 97.2% and aggregated error of 1.3%.

The proposed algorithm can be easily deployed to street view surveillance system to realize the function of pedestrian flow counting. It does not require the installation of additional equipment other than center processing unit and surveillance to achieve high performance in the street view pedestrian flow counting task.

References:

[1] J. Zhang, Z. Shi, J. Li, "Current researches and future perspectives of crowd counting and crowd density estimation technology," Computer Engineering and Science, 40(02): 282-291, 2018.

[2] Y. Xie et al., "Efficient video fire detection exploiting motion-flicker-based dynamic features and deep static features," IEEE Access, vol. 8, pp. 81904–81917, 2020.

[3] J. Pucher, R. Buehler, "Cycling towards a more sustainable transport future," Taylor Francis, 2017.

[4] D. Beymer, "Person counting using stereo," in Proceedings Workshop on Human Motion, 2002.

[5] A. Lesani, E. Nateghinia, and L. F. Miranda-Moreno, "Development and evaluation of a real-time pedestrian counting system for high-volume conditions based on 2D LiDAR," Transp. Res. Part C Emerg. Technol., vol. 114, pp. 20–35, 2020.

[6] K .Zhang, B. Guo, Z. Liu, Y. Fan, "Improved CN algorithm based on pedestrain flow detection," Computer Engineering And Design, 41(2), 411-416, 2020.

[7] J. Zhao, Y. Li, and Z. Zhou. "Learning adaptive spatial–temporal regularized correlation filters for visual tracking," IET Image Process., vol. 15, no. 8, pp. 1773–1785, 2021.

[8] M. Danelljan, F. S. Khan, M. Felsberg, and J. Van De Weijer, "Adaptive color attributes for real-time visual tracking," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1090–1097, 2014.

[9] X. Han, Y. Wang, Y. Xie, X. Qi, "Customer Flow Statistics Method Based on Deep Learning," Computer Systems Applications, 29(4):2431, 2020.

[10] G. Ning et al., "Spatially supervised recurrent convolutional neural networks for visual object tracking," in 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017.

[11] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[12] L. Yang, G. Hu, Y. Song, G. Li, and L. Xie. "Intelligent video analysis: A Pedestrian trajectory extraction method for the whole indoor space without blind areas," Comput. Vis. Image Underst., vol. 196, pp. 1-23, 2020.

[13] M. He, H. Luo, B. Hui, and Z. Chang, "Pedestrian flow tracking and statistics of monocular camera based on convolutional neural network and Kalman filter," Appl. Sci. (Basel), vol. 9, no. 8, p. 1624, 2019.

[14] J. F. Henriques, R. Caseiro, P. Martins, J. Batista. "High-Speed Tracking with Kemelized Correlation Filters," IEEE Transactions on Pattern Analysis Machine Intelligence, 37(3):583-596, 2015.

[15] T. Yin, J. Cui, "Design of Pedestrian Flow Statistics System on Video Monitoring," Electronic Sci. &Tech. Dec, 15, 2019.

[16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv [cs.CV], 2018. "unpublished"

[17] M. Ullah, F. A. Cheikh, and A. S. Imran, "HoG based real-time multi-target tracking in Bayesian framework," in 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2016.

[18] C. Zhou and J. Yuan, "Bi-box regression for pedestrian detection and occlusion estimation," in Computer Vision – ECCV 2018, Cham: Springer International Publishing, 2018, pp. 138–154.

[19] J. Munkres, "Algorithms for the assignment and transportation problems," J. Soc. Ind. Appl. Math., vol. 5, no. 1, pp. 32–38, 1957.

[20] N. Tomizawa, "On some techniques useful for solution of transportation network problems," Networks (N. Y.), vol. 1, no. 2, pp. 173–194, 1971.

[21] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," J. ACM, vol. 19, no. 2, pp. 248–264, 1972.

[22] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "Hardware-software implementation of vehicle detection and counting using virtual detection lines," in Proceedings of the 2014 Conference on Design and Architectures for Signal and Image Processing, 2014.