Flow Mosaicking: Real-time Pedestrian Counting without Scene-specific Learning

Yang Cong[1,2], Haifeng Gong[2,3], Song-Chun Zhu[2,3], Yandong Tang[1]
[1]State Key Laboratory of Robotics, Shenyang Institute of Automation, CAS
[2]Lotus Hill Institute, Ezhou, China
[3]Department of Statistics,UCLA
congyang@sia.cn, {hfgong,sczhu}@stat.ucla.edu, ytang@sia.cn

Abstract

In this paper, we present a novel algorithm based on flow velocity field estimation to count the number of pedestrians across a detection line or inside a specified region. We regard pedestrians across the line as fluid flow, and design a novel model to estimate the flow velocity field. By integrating over time, the dynamic mosaics are constructed to count the number of pixels and edges passed through the line. Consequentially, the number of pedestrians can be estimated by quadratic regression, with the number of weighted pixels and edges as input. The regressors are learned off line from several camera tilt angles, and have taken the calibration information into account.We use tilt-angle-specific learning to ensure direct deployment and avoid overfitting while the commonly used scene-specific learning scheme needs on-site annotation and always trends to overfitting. Experiments on a variety of videos verified that the proposed method can give accurate estimation under different camera setup in real-time.

1. Introduction

The crowd counting problem can be classified into two tasks: crowd counting across a detection line in certain time duration (line of interest counting, or LOI counting), and estimating the total number of pedestrians in some region at each time (region of interest counting, or ROI counting). In the published literature, there are two classes of methods for ROI counting 1) feature and pixel regression, 2) pedestrian detection. Feature or pixel **regression** methods extract the feature vectors in the region of interests and use the machine learning algorithm to regress the number of pedestrians from number of features and pixels in foreground blobs or segmented motion segments. The features include edges[7], wavelet coefficients[14], or combination of a large bank of features[16, 4]. The regression method may be linear regression[7], neural network[6], Gaussian process regression[4] or discrete classifier[16]. The number of features carried by one pedestrian is heavily affected by camera perspective. So they always need retraining using large amount of annotated data from the specific scene, which makes it inconvenient to deploy in practical applications. **Pedestrian detection** methods count pedestrians by multi-target detection. The detection is completed by background differencing[8], motion and appearance joint segmentation[20], silhouette or shape matching[22], or standard object recognition method. Though there are great progresses in object detection in recent years, robust detection of pedestrian under crowd environment is still a chal-

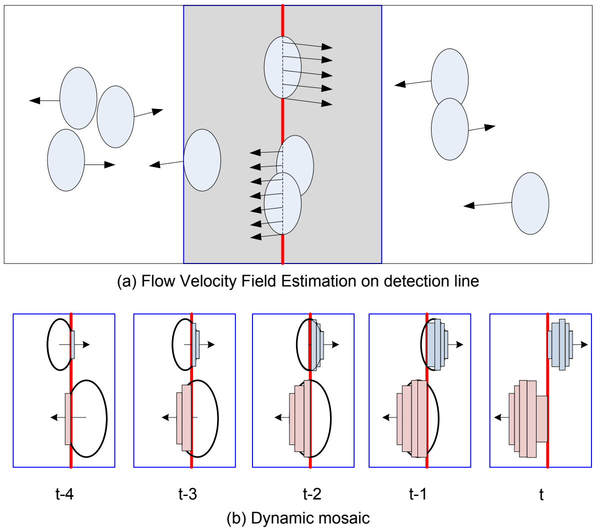


Figure 1. LOI counting by flow velocity estimation: (a) when pedestrians are crossing the detection line (red line), we can regard the moving crowd as fluid flow, and compute the flow velocity field on the detection line; (b) at each frame, using the velocity as thickness, we can crop the slices passed the line and accumulate them into patches. Finally, the number of pedestrians can be estimated in each patch .

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

lenging problem. The performance of pedestrian detection method will decrease rapidly when the crowd density and occlusion degree increase. Feature regression and pedestrian detection are only applied to ROI detection. For LOI counting, most literature adopt **feature tracking**. Features across frames are tracked into trajectories, and the trajectories are clustered into object tracks. Examples include [18], [2], [3], [13], [1],[5]. The tracking based method are hardly robust under crowd environment, and its time consumption is often huge for real-time system.

In this paper, we present a novel LOI and ROI counting algorithm based on flow velocity field estimation. We regard pedestrians crossing the line as fluid flow and compute the flow velocity on the line. By integration over time, we can estimate the number of pixels and edges passed through the line (see Figure 1). At last, we estimate the number of pedestrians by quadratic regression, with the number of pixels and edges as input. The regressors are trained off line in different camera tilt angles, and the calibration information has been taken into account.

The contribution of this paper lies in three aspects:

1. We proposed a robust algorithm for flow velocity field estimation on detection line, which takes the slow and smooth prior[21] along line into account and can be used in other applications.

2. Without scene-specific learning,we adopted an offline learning method, tilt-angle-specific learning, to facilitate application and avoid overfitting.

3. Our method unified and solved both of these two types crowd counting problem, LOI and ROI, at the same time. Moreover, this method can run real-time for LOI and high frame rate for ROI; and lots of experiments have validated the effectiveness of our method.

The remainder of the paper is organized as follows. We define our flow velocity estimation model in section 2. In section 3, the flow chart of our counting algorithm is proposed. The experiment results and conclusion are presented in section 4 and 5 respectively.

2. Flow Velocity Field Estimation

When multiple pedestrians cross a detection line, it is a challenge situation for both tracking and counting, especially when the camera tilt angle is less than 75°. We formulate the moving pedestrians as a flow field. As shown in Fig 1, when pedestrians cross the detection line, we compute the flow velocity on the detection line. At each frame, using the velocity as thickness, we can crop the slices passed the line and accumulate them into patches. Therefore, the first step is to compute the flow velocity on the line robustly. It is well known that estimating the flow velocity field robustly is a difficult task. In LOI counting, the pedestrians are mainly moving across the detection line in two directions, and we just need care the motion vectors on the line, so we can constrain the directions of possible velocities near the normal direction of the line.

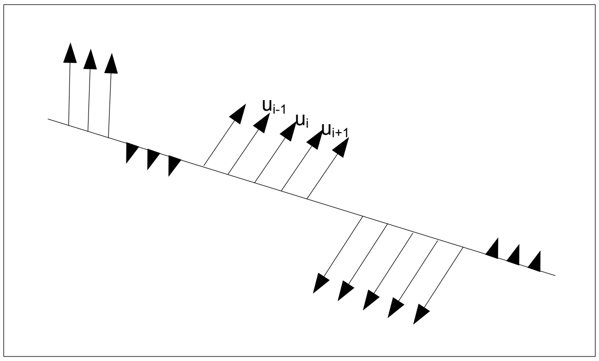


Figure 2. Flow Velocity Field Estimation: we solve 1-D flow velocity field on the detection line in two directions robustly.

In traditional optical flow, the velocity is estimated by minimizing an energy function composed of two items — iso-brightness constraint and smoothness constraint[10]. The iso-brightness constraint requires that the pixel intensity should be consistent after moving. The smoothness constraint requires that the neighbor pixels should have similar velocities. Based on the similar setting of traditional optical flow, we introduce the following improvements to make the estimation robust

1. In addition to spatial smoothness constraint, we add two additional constraints, slow motion constraint and temporal smoothness constraint.

2. In traditional optical flow, only iso-brightness between two frames are considered. We use multi-frame iso-brightness constraint to improve the robustness.

3. In traditional optical flow, all the energy terms are expressed in squares of L2 norms, which is helpful for close form solution.. We use L1 norms[12] instead, which is more robust.

In our application, users are required to specify a detection line, (see Figure.2). We assume that there are N points on the line, $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$. We solve the velocities for all point $\{\mathbf{u}_1, \cdots, \mathbf{u}_N\}$ by optical flow on the line. Let I_t be image at time t, and $I_t(\mathbf{x})$ be the pixel value at position $\mathbf{x}$. The color of particular point on an object is constant over time, so we should minimize **multi-frame iso-brightness constraint**

$$E_{\text{br}} = \sum_{i=1}^{N} \sum_{\tau=-T}^{T} |I_{t+\tau}(\mathbf{x}_i + \mathbf{u}_i\tau) - I_t(\mathbf{x}_i)| \qquad (1)$$

where $[-T, T]$ is the time span during which we assume objects crossing the line can be matched near rigidly. To

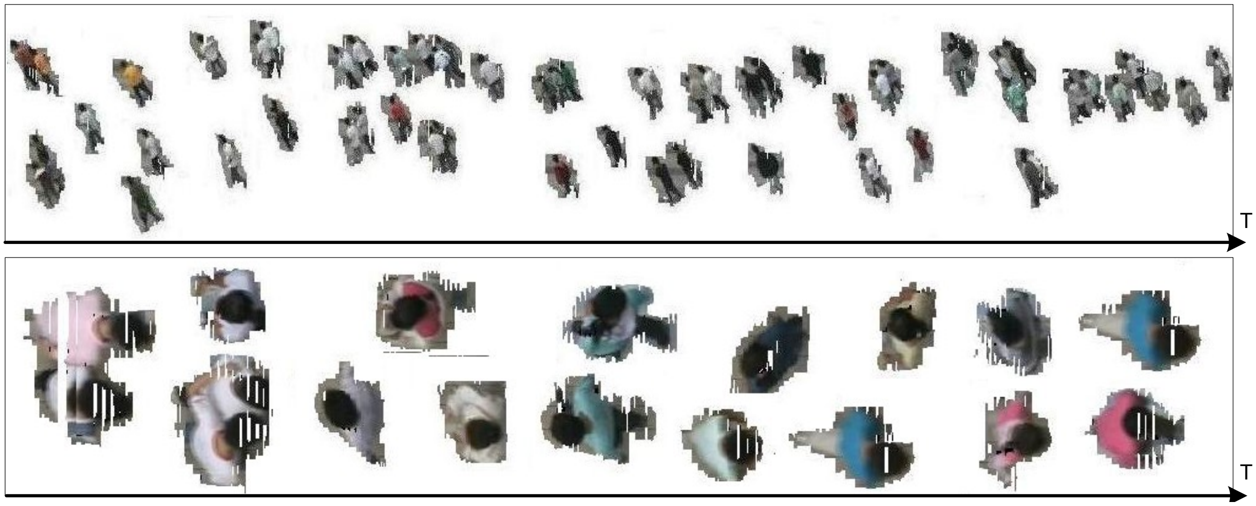


Figure 3. The mosaic results in one minute: top, camera tilt $\approx 45°$; bottom, camera tilt $\approx 90°$. From the illustration, one can see that the blobs stacked from slices are quite reasonable. It verifies that our estimations of velocities are accurate enough. For better illustration, some blank segments are removed and positions of some slightly collided blobs are adjusted.

reduce computation, for high resolution videos, we match frames $t-2$, t and $t+2$ and in low resolution ones, we match $t-1$, t and $t+1$. Similar to traditional optical flow, we apply **spatial smoothness constraint** to make the minimization well-posed

$$E_{sp} = \sum_{i=1}^{N-1} |\mathbf{u}_i - \mathbf{u}_{i+1}|. \tag{2}$$

As the pedestrians cannot move very fast, **slow motion constraint** is imposed to make the estimated velocities more robust

$$E_{sl} = \sum_{i=1}^{N} |\mathbf{u}_i|. \tag{3}$$

In most of cases, pedestrians are moving smoothly without abrupt acceleration, so we have **temporal smoothness constraint**

$$E_{tp} = \sum_{i=1}^{N} |\mathbf{u}_i - \mathbf{u}'_i| \tag{4}$$

where $\mathbf{u}'_i$ is the velocity of the i-th point in previous frame. The slow and smooth motion prior has been verified by [21] in human motion perception.

Finally, we combine all the four energy Eq (1–4), and find the velocity field by minimizing

$$E = E_{br} + \alpha E_{sl} + \beta E_{tp} + \gamma E_{sp}, \tag{5}$$

where α, β and γ are weights for the corresponding energy items (typically, we set $\alpha = 200$, $\beta = 10$, $\gamma = 2000$).

Because the velocities lie on a line, it can be solved by dynamic programming. We avoid derivatives to ensure robustness and global minimum. The total energy (5) can be expanded in frame-wise and frame-pair-wise items,

$$E = \sum_{i=1}^{N} D_i + \sum_{i=2}^{N} D_{i-1,i} \tag{6}$$

where $D_i = \sum_{\tau=-T}^{T} |I_{t+\tau}(\mathbf{x}_i + \mathbf{u}_i\tau) - I_t(\mathbf{x}_i)| + \alpha|\mathbf{u}_i| + \beta|\mathbf{u}_i - \mathbf{u}'_i|$ and $D_{i-1,i} = \gamma|\mathbf{u}_{i-1} - \mathbf{u}_i|$. For each velocity $\mathbf{u}_i$, it has K candidate solutions, $\{\mathbf{u}_i^{(1)}, \cdots, \mathbf{u}_i^{(K)}\}$, where K is relate to the maximal velocity. In this paper, the range of velocities is $[-\mathrm{MaxVel}, +\mathrm{MaxVel}]$, where MaxVel is 20 for 90 degree and 10 for 45 and 65 degrees and the step size is one pixel. For each candidate $\mathbf{u}_i^{(k)}$, we can associate a minimal partial energy up to i, as $E_i^{(k)}$. With these notations, the detail of steps are as below:

1. For each candidate $\mathbf{u}_1^{(k)}$ for $\mathbf{u}_1$, let $E_1^{(k)} = D_1$. Let $i = 2$.
2. For each candidate velocity k for $\mathbf{u}_i$, let $E_i^{(k)} = D_i + \min_{k'}[D_{i-1,i} + E_{i-1}^{k'}]$, and record the match in previous frame k'.
3. If $i = N$, choose $E = \min_k E_N^{(k)}$, trace back for the optimal candidate at each frame and exit; else let $i \leftarrow i + 1$ repeat step 2.

3. Counting by regression

3.1. Dynamic mosaic

The velocity field on the detection line, as shown in Figure 2, is segmented by moving directions after small velocities removed. We drop the short segments and count moving in two directions separately. For each segment, we

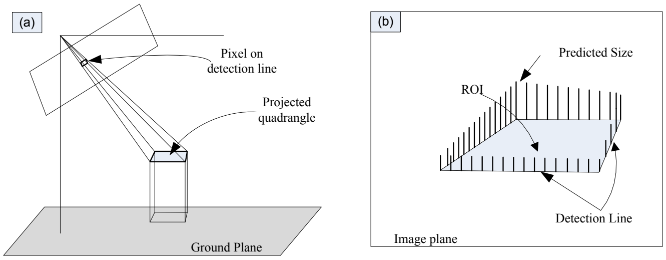


Figure 4. Perspective normalization: (a) illustrates the area of a pixel in the image plane projected to the shoulder height plane; (b) is the weight coefficient of each pixel on the boundary of the ROI. The farther the distance is, the greater the value.

use its average velocity as thickness to crop a slice. The slices at the same position are accumulated and stacked together over time to construct a blob. We call this process dynamic mosaic, as Figure 1(b). The mosaic is terminated when there are no motions or the current blob size reaches a threshold. Figure 3 shows two results on mosaic for two different camera tilt angle, from which we can conclude that our flow velocity estimation and dynamic mosaic are effective.

Before dynamic mosaic, the background pixels from the slices are removed by a background model. Three background models are tried in this paper, Gaussian Mixture model[19], LBP[9] and Incremental PCA[24], and found to give similar results.

After dynamic mosaic stops for one blob, we estimate the number of pedestrians in the blob by regression on its normalized area and number of edges. We can take into account other complex features, but the simple features still produce acceptable results.

3.2. Perspective normalization

The area and number of edges on each pedestrian are heavily affected by camera perspective. Some crowd counting algorithms have introduced normalization to deal with the perspective projection [11, 17]. We handle the perspective problem by projecting each pixel on the detection line to the shoulder height plane and weighting it by its area on the shoulder height plane. We use a simplified version of calibration algorithm proposed in [15] to obtain a rough calibration matrix. We treat each pixel as a square and use the calibration matrix to project vertices of the square to the shoulder plane to obtain a quadrangle. We use the area of the quadrangle as the weight of the corresponding pixel, see Figure 4. In ROI, we just need to normalize the area along its surrounding region.

3.3. Regression

Many algorithms based on pixel-feature or texture-feature need online training for application, which trends to overfitting and prevent it from broadly application, for example[4]. We present an off-line learning algorithm which do not need retraining on a new position.

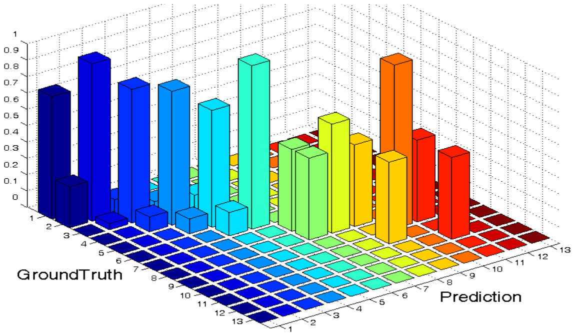


Figure 5. The regression result in 45 degree tilt-angle. The X-axis and Y-axis are the ground truth and prediction respectively, and the height of each bars(Z-axis) denotes the probability of regression result.

As the pedestrians present different appearances at different camera tilt angle. If the angle is about 45 degree, torsos and legs can be seen and the occlusion is heavy. If the angle is increased to about 65 degree, the torso and legs can be seen and the occlusion level decreases. If the angle is nearly vertical, only the head and shoulder can be seen and there are nearly no occlusion. Therefore, we classify these angles into three categories, 90 degree, 65 degree and 45 degree, and do regression separately.

For each category, we describe the dynamic blobs using two features: total number of weighted pixels A and total number of weighted edge pixels B. The edge pixels are computed by Canny edge detector. Given the calibration matrix, we can predict the height of pedestrian H (for 90 degree, H is the width of shoulder) on the detection line in pixel. Finally, we train a quadratic regressor using $(A, B, 1/H)$ as input and number of pedestrians in each blob as output. Figure 5 illustrates the regress result in the most difficult category, 45 degree. The ideal result is that all of the bars distribute in diagonal. When the pedestrian number is large, it will be a little scattered due to heavy occlusion. But we still get acceptable results.

4. Experiments

4.1. Data

There are total of 12 different videos used in our experiments, classified into 3 categories by its tilt angles and each categories includes 4 videos. As shown in Figure 6, the first row is the 90 degree categories, including 4 videos, named from 1-1 to 1-4; and the second row and third row are the 65 degree and 40 degree categories respectively. All the videos are from the LHI database[23]. These videos cover from the perspective of 45 degree to nearly 90 degree under different scenes, and all the video images are down sampled to

352×288. Moreover, all the video sequences are labeled ground truth for both of LOI counting and ROI counting. For each view angle, we use $1/3$ of all videos for training and the other $2/3$ for testing.

4.2. LOI counting

We measure the accuracy of the LOI counting by the following equation

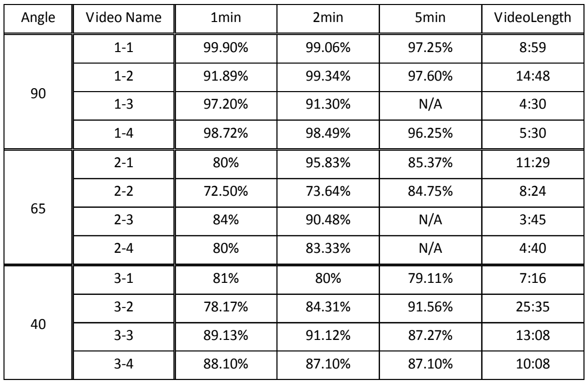
$$\text{Accuracy} = 1 - \frac{|\sum \text{\# of ground truth} - \sum \text{\# of predicted}|}{\sum \text{\# of ground truth}}. \tag{7}$$

The end users of LOI counting system always care about how many pedestrians pass the line in a certain time, such as one day, one hour or 10 minutes. So we compute the accuracy of our system in a certain time duration (see Table 1). To reduce the impact of the frames without pedestrian, we also compute the accuracy in certain number of pedestrian, see Table 2. From these two tables, one can see that for a duration of 5 minutes or 40 pedestrians, our system obtains perfect results for 90 degree camera tilt angles; and acceptable results for 65 and 40 degree angle,this is because the occlusion increases when the tilt angle decreasing. For shorter duration, the results are also reasonable. Some demonstrations are shown in Figure 6. Total numbers of ground truth and predicted numbers over time are shown in Figure 7. We can conclude that our algorithm can count the number of the pedestrians across the detection line accurately. In comparison to 90 degree condition, in which the accuracy rate are from 99% to 98.4%[1], and from 100% to 90.4% [5] in 90 degree. In the 90 degree category of table 2, our LOI result is between 97.66% and 93.33% under four different scenes, which are very competitive to [1][5]. However, our algorithm is an offline learning method, and can be used without retraining in new scenes. Moreover, our algorithm can still work even when the tilt angle decreases to nearly 45 degree. So far as I know, no one has done the LOI counting in case of 65 and 45 degree scene.

4.3. ROI counting

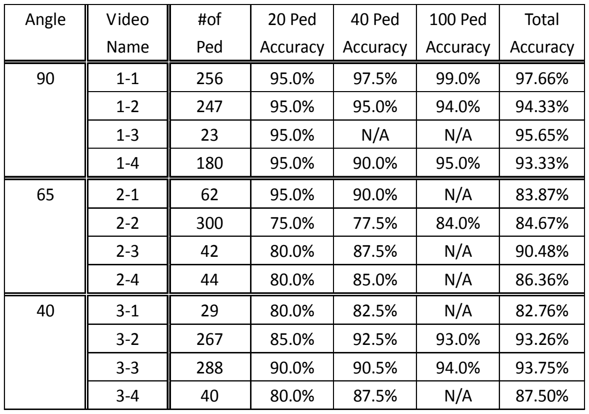
We can adapt our algorithm to perform ROI counting. If user specifies the ROI by a polygon, we can compute number of pedestrians who enter and exit the region by LOI counting. To prevent accumulated error, we first check whether there are any foreground blob in the region. If no, we reset the counter to zero. We test ROI counting on three of our videos, demonstrations are shown in Figure 8 and the curves of numbers over time are show in Figure 9. From the curves, one can see that in most times, the predicted values match the ground truth exactly.

Two measures (Mean-Squared-Error(MSE) and the Absolute Error) are used to analyze the accuracy of ROI counting quantitatively:



Angle	Video Name	1min	2min	5min	VideoLength
90	1-1	99.90%	99.06%	97.25%	8:59
	1-2	91.89%	99.34%	97.60%	14:48
	1-3	97.20%	91.30%	N/A	4:30
	1-4	98.72%	98.49%	96.25%	5:30
65	2-1	80%	95.83%	85.37%	11:29
	2-2	72.50%	73.64%	84.75%	8:24
	2-3	84%	90.48%	N/A	3:45
	2-4	80%	83.33%	N/A	4:40
40	3-1	81%	80%	79.11%	7:16
	3-2	78.17%	84.31%	91.56%	25:35
	3-3	89.13%	91.12%	87.27%	13:08
	3-4	88.10%	87.10%	87.10%	10:08

Table 1. LOI counting accuracy in time, including three categories, each row with the name in the second column corresponds to the video with the same name in Figure 6. We compare the accuracy of each video in 1min, 2min and 5min respectively. The last column is the video length used for testing.



Angle	Video Name	#of Ped	20 Ped Accuracy	40 Ped Accuracy	100 Ped Accuracy	Total Accuracy
90	1-1	256	95.0%	97.5%	99.0%	97.66%
	1-2	247	95.0%	95.0%	94.0%	94.33%
	1-3	23	95.0%	N/A	N/A	95.65%
	1-4	180	95.0%	90.0%	95.0%	93.33%
65	2-1	62	95.0%	90.0%	N/A	83.87%
	2-2	300	75.0%	77.5%	84.0%	84.67%
	2-3	42	80.0%	87.5%	N/A	90.48%
	2-4	44	80.0%	85.0%	N/A	86.36%
40	3-1	29	80.0%	82.5%	N/A	82.76%
	3-2	267	85.0%	92.5%	93.0%	93.26%
	3-3	288	90.0%	90.5%	94.0%	93.75%
	3-4	40	80.0%	87.5%	N/A	87.50%

Table 2. LOI detection accuracy in number,each row corresponds to the video with the same name in Figure 6. The third column is the total number of pedestrians in each video, and we compare the accuracy of each video in 20, 40, 100 and total number respectively.

$$\text{MSE} = \frac{\sum_{\text{all frames}} |\text{\# of ground truth} - \text{\# of predicted}|^2}{\text{total \# of frames}}, \tag{8}$$

$$\text{Absolute Error} = \frac{\sum_{\text{all frames}} |\text{\# of ground truth} - \text{\# of predicted}|}{\text{total \# of frames}}. \tag{9}$$

Table 3 shows the MSE and Absolute Error from 3 videos of different categories. In comparison to [4], in which the MSE is larger than 4.181 and the Absolute Error is larger than 1.621. we can conclude that our algorithm is much more accurate than [4]. More over, our algorithm facilitate to use in different scenes directly without anymore retraining.

Video name	Tot Num	MSE	Absolute Error
1-1	17595	1.0388	0.4088
2-1	41556	0.6308	0.3101
3-2	3539	0.139	0.1152

Table 3. ROI counting accuracy. The second column is the total number of pedestrians appearing in each test video, and the MSE and Absolute Error are adopted for comparision.

Our algorithm is implemented in C and tested on a machine with P4 2.8GHz CPU and 1G RAM. For LOI counting, it can process 25 frames per second; and for ROI counting, where more computation is needed, it can still process 17 frames per second on average. The processing time includes the time consumed by background modeling.

5. Conclusion

In this paper, by regarding the moving pedestrians crowd as fluid flow, we present a novel crowd counting algorithm, which unified both of the LOI and ROI problems together and solved it based on our flow velocity field estimation model and offline learning. Our velocity estimation model can estimate the flow velocity and get the dynamic mosaic accurately and robustly. Without scene-specific learning, an offline learning method (tilt-angle-learning) are adopted for learning and testing, and get satisfied results. If we use the training method like some feature-based method, we can get much higher accuracy rate, but it tend to arose overfitting and make no sense for application. However, our algorithm is very convenient for practical application, because it just need to calibrate without retraining under new scene. The testing results above have proved these. In conclusion, our algorithm can be applied to solve both of the LOI and ROI problem in real-time effectively and robustly.

6. Acknowledge

This work is done at Lotus Hill Institute and is supported by NSF China (60875005, 60832004, 60871078) and a 863 project 2008AA01Z126.

References

[1] A. Albiol, I. Mora, and V. Naranjo. Real time high density people counter using morphological tools. *IEEE Trans. on Intelligent Transportation Systems*, 2(4), 2001.

[2] G. Antonini and J. P. Thiran. Counting pedestrians in video sequences using trajectory clustering. *IEEE Trans. on Circuits and Systems for Video Technology*, 16(8), 2006.

[3] G. J. Brostow and R. Cipolla. Unsupervised bayesian detection of independent motion in crowds. In *CVPR*, 2006.

[4] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, 2008.

[5] T. Chen, T. Chen, and Z. Chen. An Intelligent People-Flow Counting Method for Passing Through a Gate. In *Robotics, Automation and Mechatronics, IEEE Conf. on*, 2006.

[6] S. Y. Cho, T. W. S. Chow, and C. T. Leung. A neural-based crowd estimation by hybrid global learning algorithm. *IEEE Tran. on Systems, Man, and Cybernetics*, 29(4), 1999.

[7] A. C. Davies, J. H. Yin, and S. A. Velastin. Crowd monitoring using image processing. *Electronics and Communication Engineering Journal*, 1995.

[8] L. Dong, V. Parameswaran, V. Ramesh, and I. Zoghlami. Fast crowd segmentation using shape indexing. In *ICCV*, 2007.

[9] M. Heikkilä and M. Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *PAMI*, pages 657–662, 2006.

[10] B. Horn and B. Schunck. Determining optical flow. *AI*, 17(1-3):185–203, August 1981.

[11] D. Kong, D. Gray, and H. Tao. A viewpoint invariant approach for crowd counting. In *ICPR*, 2006.

[12] A. Kumar, A. Tannenbaum, and G. Balas. Optical flow: a curve evolution approach. *IEEE Trans. on Image Processing*, 5(4):598–610, 1996.

[13] B. Leibe, K. Schindler, and L. V. Gool. Coupled detection and trajectory estimation for multi-object tracking. In *ICCV*, 2007.

[14] S. Lin, J. Chen, and H. Chao. Estimation of number of people in crowded scenes using perspective transformation. *IEEE Trans. on Systems, Man, and Cybernetics*, 2001.

[15] F. Lv, T. Zhao, and R. Nevatia. Self-calibration of a camera from video of a walking human. In *ICPR*, 2002.

[16] A. Marana, L. da Costa, R. Lotufo, and S. Velastin. On the efficacy of texture analysis for crowd monitoring. In *Proc. Computer Graphics, Image Processing and Vision*, 1998.

[17] N. Paragons and V. Ramesh. A mrf-based approach for real-time subway monitoring. In *CVPR*, 2001.

[18] V. Rabaud and S. Belongie. Counting crowded moving objects. In *CVPR*, 2006.

[19] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, 1999.

[20] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *IJCV*, 63(2), 2005.

[21] Y. Weiss and E. H. Adelson. Slow and smooth: A bayesian theory for the combination of local motion signals in human vision. Technical report, MIT, 1998.

[22] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *ICCV*, 2005.

[23] B. Yao, X. Yang, and S. Zhu. Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks. In *6th Int'l Conf on EMMCVPR*, 2007.

[24] Y. Zhao, H. Gong, L. Lin, and Y. Jia. Spatio-temporal patches for night background modeling by subspace learning. In *ICPR*, 2008.

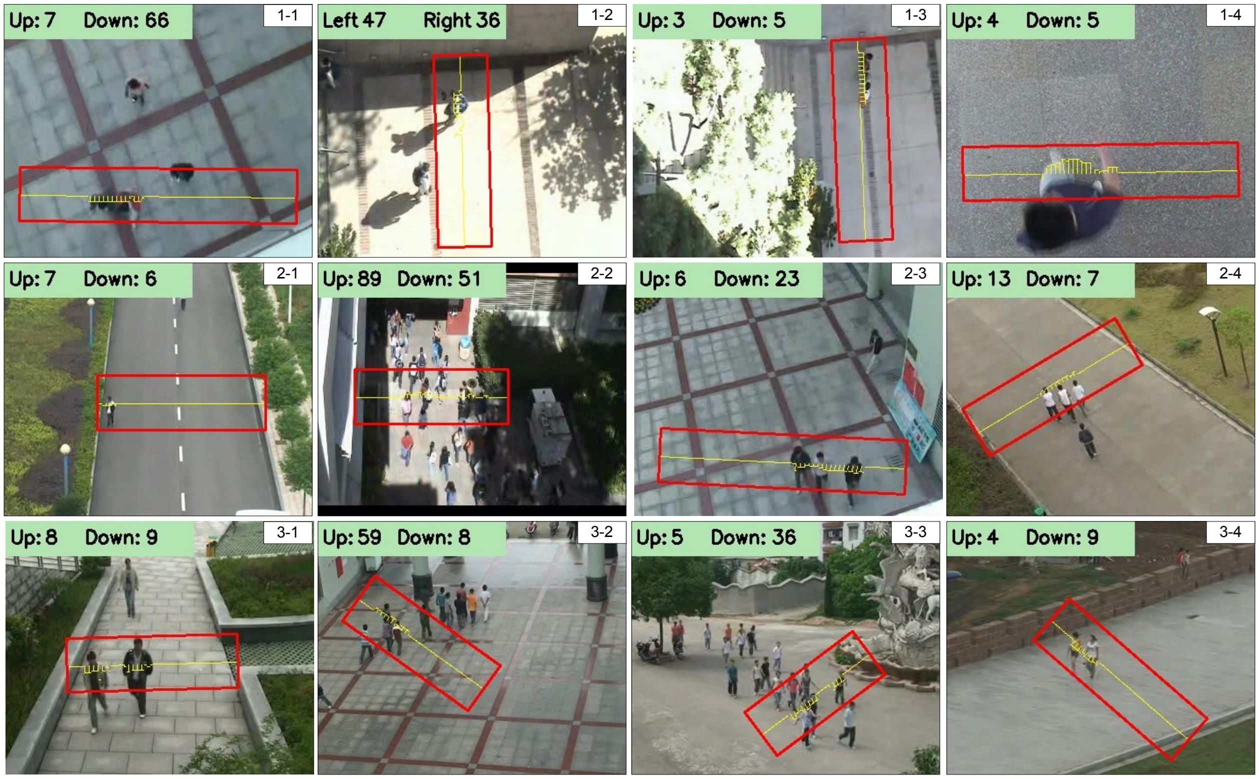


Figure 6. Demonstration of LOI results. These 12 images are classified into 3 categories(the first row is the 90 degree category, the second row and third row are the 65 degree and 40 degree category), and each category includes 4 videos under different scenes. In each image, the total number of Up and Down(or Left and Right) the detection line are shown in the top-left marked in light green background color; the yellow line is detection line, the short line segments attached to the detection line are estimated velocities, and the red rectangle is the detection region for flow velocity estimation.

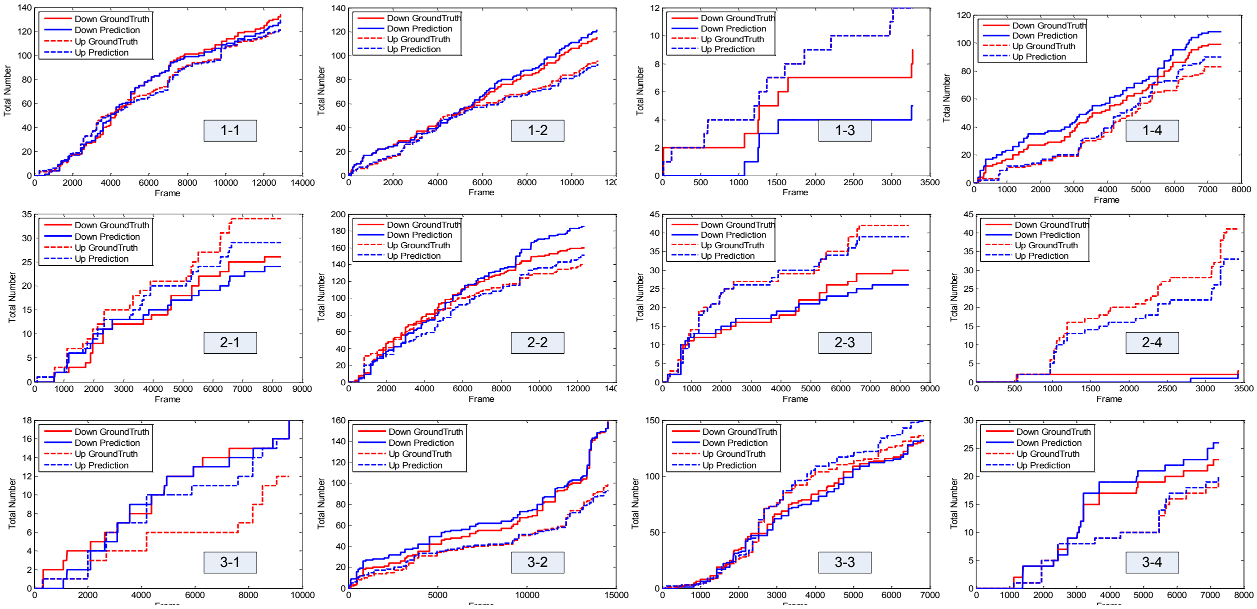


Figure 7. LOI results over time. There are total of 12 graphes, each graph corresponds to the image in the same location of Figure 6. The total number of LOI in the DOWN and UP direction are shown in the solid and dot lines respectively, and the red and blue colors distinguish the Ground Truth and the Prediction in each direction.

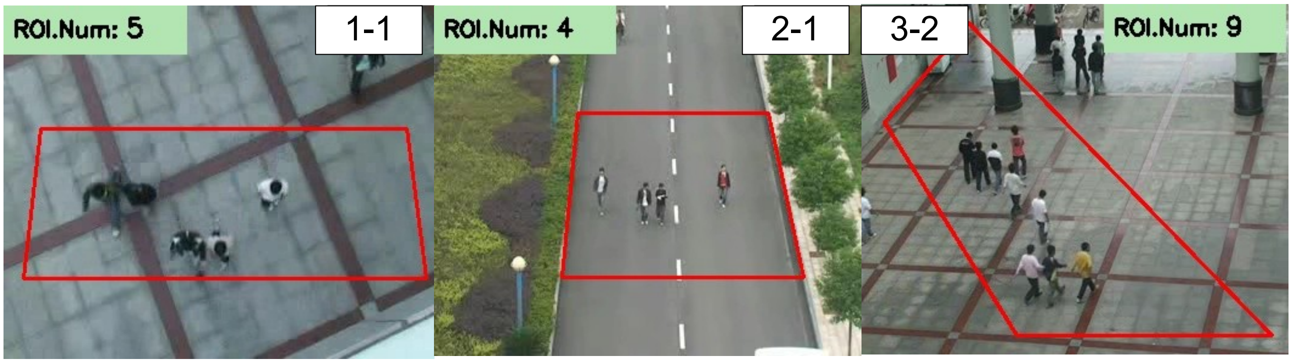


Figure 8. Demonstration of ROI results. These three video from three different categories were taken. The red quadrangles are the boundary of the ROI, the predicted result of the pedestrians number is shown in the top marked by light green color.

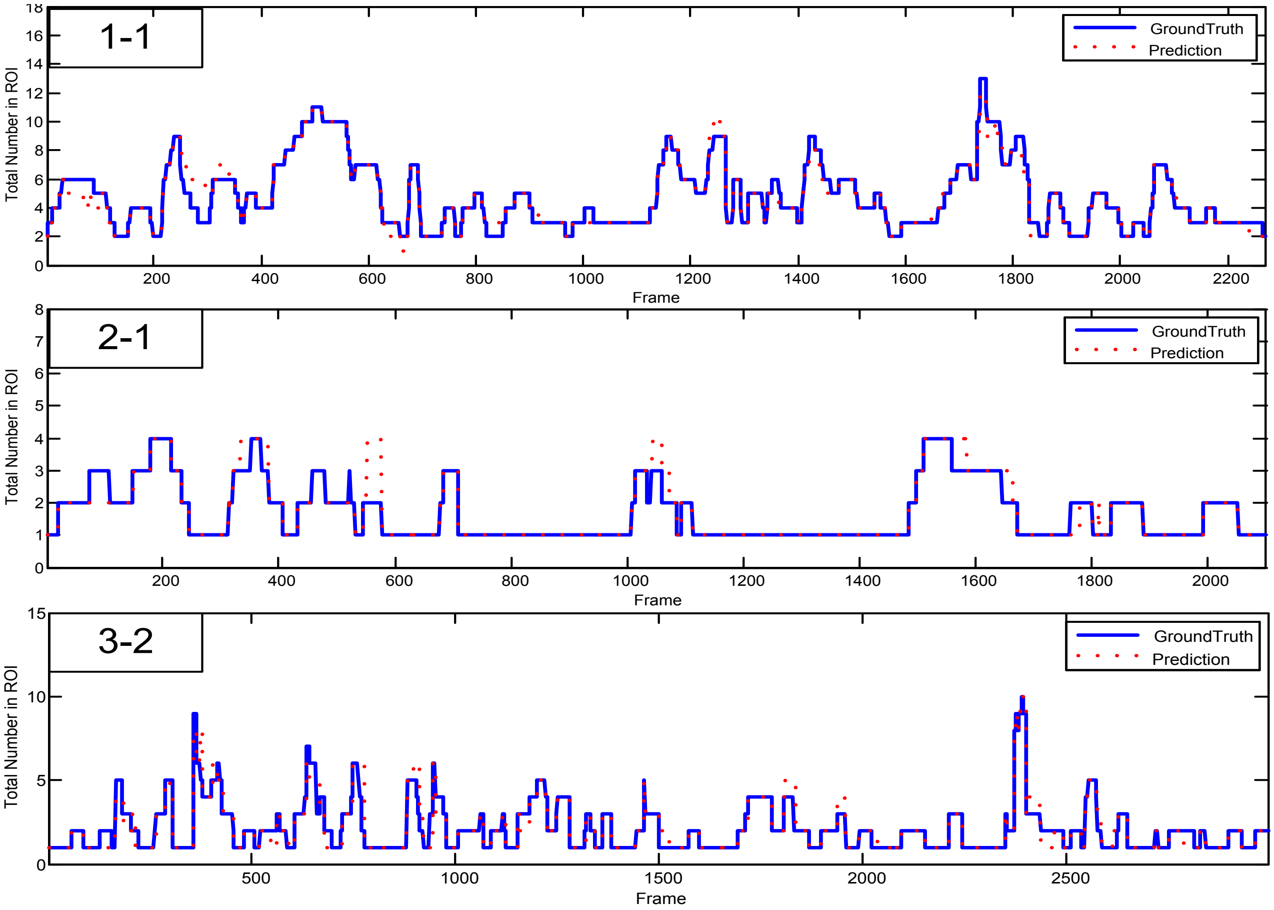


Figure 9. ROI results over time for 90 degree, 65 degree and 40 degree respectively. The Ground Truth and the Prediction are marked in blue and red solid lines.