Research Article

Pedestrian Detection and Tracking for Counting Applications in Metro Station

Chen Yan-yan, Chen Ning, Zhou Yu-yang, Wu Ke-han, and Zhang Wei-wei

Beijing University of Technology, Chaoyang District Beijing, China

Correspondence should be addressed to Chen Ning; chenningcut@126.com

Received 9 November 2013; Revised 12 January 2014; Accepted 19 January 2014; Published 27 February 2014

Academic Editor: Wuhong Wang

Copyright © 2014 Chen Yan-yan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A pedestrian counting method based on Haar-like detection and template-matching algorithm is presented. The aim of the method is to count pedestrians that are in a metro station automatically using video surveillance camera. The most challenging problem is to count pedestrians accurately in the case of not changing the position of the surveillance camera, because the view that surveillance camera uses in a metro station is always short-shot and nondirect downward view. In this view, traditional methods find it difficult to count pedestrians accurately. Hence, we propose this novel method. In addition, in order to improve counting accuracy more, we present a method to set the parameter value with a threshold-curve instead of a fixed threshold. The results of experiments show the high accuracy of our method.

1. Introduction

Metro as a mode of passenger transport with the characters of speed and comfort has been favored by the public in many cities. More and more urban residents like to choose metro as their transportation tool. Because of the large number of passengers that take metro, many cities' metro stations have often crowded especially in some rush hours. If the crowded situation cannot be evacuated promptly, it will impact the metro station's operation and management. What is more, a stampede will very likely occur if the crowded situation lasts for long or becomes severe. Therefore, to avoid the congestion and prevent accidents, metro station staff needs to get passenger flow information in real time so as to evacuate passengers promptly. Pedestrian counting information is the most important among these passenger flow information.

However, at present, the main way to get the pedestrian counting information in a metro station is to estimate the passenger flow volume from surveillance video by monitoring personnel. This way is simple and easy to implement, but it has some disadvantages that are difficult to overcome. Firstly, because of getting the information in manual collection, the pedestrian counting information that metro station staff can get is not quantitative data but qualitative data. So the information is inaccurate and easily affected by subjective factors

of monitoring personnel. Secondly, monitoring personnel is difficult to provide the metro station staff with full and correct information when monitoring points are added.

Getting pedestrian counting information in automatic collection has unparalleled advantages compared with getting the information in a manual way. These advantages are as follows. First, the automatic way can improve the precision of pedestrian counting data and turn the qualitative data into the quantitative data. Therefore, this way can provide accurate and real-time data that can help metro station staff to analyse the situation of passenger flow and handle anomalous event. Second, this way can decrease workload of the monitoring personnel and reduce influence of subjective factors.

The way to count pedestrians in a metro station automatically based on computer vision has a huge cost advantage compared to the other ways, because it can use the existing surveillance video equipment which is common in a metro station. However, the computer vision way is not mature due to the factors of complicated occlusions and cluttered scenes. For that reason, this paper studies how to count pedestrians that are in a metro station accurately using related algorithm of image processing and pattern recognition. This paper's purpose is to present a method to count pedestrians in a metro station accurately and in real time without changing the position of the existing surveillance camera.

The method presented in this paper consists of three parts. First part is to detect pedestrian using Haar-feature and MHI algorithm and then to track pedestrian using template-matching algorithm in the second part. Finally the method of threshold-curve is proposed to improve the counting accuracy.

2. Related Work

Recently, many image-processing based methods of counting people were proposed. As using in different scenes, these methods are quite different.

Mittal et al. have proposed a traffic management system using both audio and video data [1]. Hou and Pang have developed an effective method for estimating the number of people in a low-resolution image with complicated scenes in real time [2]. Chan and Vasconcelos have presented an approach to the problem of estimating the size of inhomogeneous crowds which are composed of pedestrians that travel in different directions [3]. Xiong et al. have proposed a potential energy-based model to estimate the number of people in public scenes as the fundamental research to detect the abnormal crowd behavior [4]. Sacchi et al. have exploited the image-processing tools for moving-object detection and classification in the context of an actual application involving the remote monitoring of a tourist site [5]. Schofield et al. have designed a system to distinguish between parts of the background scene and nonbackground objects (people) [6]. Hashimoto et al. have developed a people-counting system with human information sensors [7]. Amin et al. have presented a system for counting people in a scene using a combination of low cost, low-resolution visual and infrared cameras [8]. Huang and Chow have described a people-counting system using hybrid RBF neural network [9]. Schofield et al. have described a method for counting the number of people in any predefined scene using RAM-based neural network classifiers [10]. Kopaczewski et al. have presented an algorithm for people counting in crowded scenes based on the idea of virtual gate which uses optical flow method [11]. Vicente et al. have shown the algorithm implementation for a field-programmable gate array- (FPGA-) based design for people counting using a low-level head-detection method [12]. Conte et al. have presented a novel method by establishing a mapping between some scene feature and the number of people to provide an estimate of people count [13].

Most of the previous works can be classified into two classes by camera view. One captures the video clips by long-shot and nondirect downward view camera [1–5] and the other by short-shot and direct downward view camera [6–13] (see Figure 1). The image-processing method of the first class is the first to detect and track the foreground group by body shape feature and then segment the group into individuals for counting. And the image-processing method in the second class is the first to detect foreground group and segment it into individuals by head-shoulder shape feature and then track the individuals for counting. The second class method always has lower computational complexity than the first class

because it has no need to process occlusions. However, the view of the second class method is smaller than the first class method. Both of the two class methods can make good performances in counting people accurately by their suitable views, but they are not suitable for counting pedestrians in a metro station as its camera is always set in short-shot and nondirect downward view (see Figure 2). In this view, it is difficult to segment group into individuals because of the severe body shape deformation. To solve this problem, this paper aims to propose a novel method using Haar-like feature to detect pedestrians' head and using template-matching approach to track them for counting. The reason why we select this method is that head of the pedestrian is seldom occluded in this view. Although this method is easy to make error detected, the approach of threshold-curve can compensate it.

3. Detecting and Tracking Pedestrian

3.1. Haar-Like Based Pedestrian Detection. In this paper, pedestrian is recognized and detected using Haar-like featured detector and AdaBoost classifier. This detection process has two steps—training and recognition. The flowchart of this process is shown in Figure 3.

The first step is to train samples. First of this step, we use the mouse to cut head samples from video clips by Pedestrian Detection Software (see Figure 4). In this paper, a total of 4324 samples are collected which are composed of 1000 pedestrian head positive samples and 3324 negative samples. Each positive sample is resized to 24×24 pixels and the negative samples are collected from the Internet including images of mountain, river, cartoon, and animal. After preparing this sample set, a classifier is trained to judge whether an object is a head or not. At first glance, samples' raw pixel value is an ideal feature to train a classifier, but the challenge of calculation complexity makes it difficult to take into use. Thus, the Haar-like feature instead of raw pixel value feature has been chosen. Each Haar-like feature is composed of two or three “black” and “white” rectangles joining together—these rectangles can be up-right or rotated by 45 degrees (see Figure 5). The Haar-like features value is calculated as a weighted sum of two components: the pixel gray level values sum over the black rectangle and the sum over the whole feature area (all black and white areas) [14]. An image of 24×24 contains millions of Haar-like features, so an effective feature selecting method is necessary. AdaBoost has been proven to be an effective method in selecting classifiers from millions of weak classifiers. So the AdaBoost algorithm has been selected as the learning algorithm to train the cascade classifier. After training, a pedestrian head classifier has been got.

The second step is to recognize pedestrian by pedestrian head classifier. In order to improve the speed of recognition, the region of interest (ROI) is set in the video clips. In ROI, the head classifier is adopted to recognize pedestrian in each frame (see Figure 6). The range between upper-redline and lower-redline is ROI. Two pedestrian heads are detected and highlighted by green rectangles in ROI.

FIGURE 1: (a) Long-shot and nondirect downward view. (b) Short-shot and direct downward view.

FIGURE 2: Short-shot and nondirect downward view.

FIGURE 4: Pedestrian Detection Software.

FIGURE 3: Flowchart of Haar-like based pedestrian detection.

FIGURE 5: Haar-like features prototypes.

FIGURE 6: Recognizing pedestrian head in frame image.

3.2. MHI Based Orientation Detection. To represent how motion pedestrian works a motion history image (MHI) is formed. In an MHI, H_t pixel intensity is a function of the temporal history of motion at that point. For the results

FIGURE 7: (a) Original ROI-image. (b) MHI ROI-image.

FIGURE 8: (a) Criterion 1. (b) Criterion 2.

presented here a simple replacement and decay operator is used:

$$H_\tau(x, y, t) = \begin{cases} \tau & \text{if } D(x, y, t) = 1 \\ \max(0, H_\tau(x, y, t-1) - 1) & \text{otherwise.} \end{cases} \quad (1)$$

The result is a scalar-valued image compared with which moving pixels are brighter [15]. Examples of MHI are presented in Figure 7(b). Green circle represents the orientation of this pedestrian.

3.3. Template-Matching Based Pedestrian Tracking. Object association is of great importance to the application of multiple targets tracking. In order to get the associations between a single recognized rectangle and a specific template, the method of template-matching is adopted. In other words, we use template-matching algorithm to judge whether a single recognized rectangle matches to a specific template [16]. If the association between one recognized rectangle and one template satisfies the following two criteria, this recognized rectangle is considered as matching to this template.

(1) The center of recognized rectangle is located in the searching area of the template (see Figure 8(a)). The formula is listed as follows:

$$\begin{aligned} & \text{Template}(x) - \text{Length}_{\text{Search}} \\ & < \text{Detect}(x) < \text{Template}(x) + \text{Length}_{\text{Search}}, \\ & \text{Template}(y) - \text{Length}_{\text{Search}} \\ & < \text{Detect}(y) < \text{Template}(y) + \text{Length}_{\text{Search}}. \end{aligned} \quad (2)$$

In formula (2), Template (x) stands for x position of template in frame image, and Detect (x) stands for x position of detection rectangle in frame image, and so on for Template (y) and Detect (y). Search represents the searching threshold of the template.

(2) The orientation of detection rectangle keeps the pace with the deviation angle of the template (see Figure 7(b)). The formula is presented as follows:

$$\begin{aligned} & \text{Angle}(D_{\text{cur}}(x, y) \rightarrow T_{\text{last}}(x, y)) \\ & > \text{Angle}(T_{\text{last}}(x, y) \rightarrow T_{\text{first}}(x, y)) - \text{AT}, \\ & \text{Angle}(D_{\text{cur}}(x, y) \rightarrow T_{\text{last}}(x, y)) \\ & < \text{Angle}(T_{\text{last}}(x, y) \rightarrow T_{\text{first}}(x, y)) + \text{AT}. \end{aligned} \quad (3)$$

In formula (3), D refers to the detection rectangle, T refers to the template, cur stands for the current frame, first stands for the first frame of the template, last stands for the last frame of the template, AT stands for the threshold of the angle, and x refers to x position and so does y .

If one detection rectangle matches with one template, this template is updated as in Table 1.

In order to make the template's position updated with a fast speed while the template's size with a slow speed, the specific values ($\alpha_1 = 0.95$, $\alpha_2 = 0.35$) are set in this paper.

If one detection rectangle cannot match each template in template list, a new template is created and added to template list. This new template is initialized as in Table 2.

While a template is unable to match with a new detection rectangle for a moment (5 frames in this paper), it is feasible to delete this template from the template list so as to judge whether or not this template is a pedestrian. In this paper,

FIGURE 9: Data collection software.

TABLE 1: Template update.

Character	Explanation	Update criterion	Update method
x	x position of center	Matching	$x = x \times (1 - \alpha_1) + \alpha_1 \times x_{\text{detection}}$
y	y position of center	Matching	$y = y \times (1 - \alpha_1) + \alpha_1 \times y_{\text{detection}}$
dx	MHI accumulated value of x direction	Matching	$dx+ = dx_{\text{detection}}$
dy	MHI accumulated value of y direction	Matching	$dy+ = dy_{\text{detection}}$
Width	Template width	Matching	Width = width $\times$ (1 - α_2) + $\alpha_2 \times$ width _{detection}
Height	Template height	Matching	Height = height $\times$ (1 - α_2) + $\alpha_2 \times$ height _{detection}
Score	The number of matching with detection rectangle	Matching	Score++
Scoreperfrm	The number of matching in current frame	Matching, the end of frame	Scoreperfrm++ matching 0 the end of frame
Scoremax	The max of scoreperfrm in template	The end of each frame	Max(scoreperfrm, scoremax)
Frame	The frame of last matching	Matching	Frame = frame _{detection}

the template is considered as a pedestrian if it meets the following conditions:

$$\begin{aligned}
 & \text{MIN_SIZE} < \text{width} \times \text{height} < \text{MAX_SIZE}, \\
 & \text{score} > \text{GOOD_SCORE}, \\
 & \frac{\text{score}}{(\text{frame} - \text{startfrm} + 1)} > \text{GOOD_SCORE_RATIO}, \\
 & \left| \frac{dy}{(\text{frame} - \text{startfrm} + 1)} \right| > \text{GOOD_DIR_AVER}, \\
 & \text{score max} > \text{GOOD_SCORE_MAX}, \\
 & |y - \text{starty}| > \text{Y_AXIS_MIN}.
 \end{aligned} \tag{4}$$

In formula (4), the parameters of MIN.SIZE and MAX.SIZE are set for deleting some templates the size of which is too big or too small. The parameters of GOOD.SCORE, GOOD.SCORE.RATIO, and GOOD.SCORE.MAX are set because templates created by image noise are more difficult to match than the other templates. And the parameters of GOOD.DIR.AVER and Y.AXIS.MIN are set because the

FIGURE 10: The scatter gram of the frequency of the template samples.

positions of templates produced by the noise of images change a little. The method of how to set these parameters' thresholds will be elaborated in the next section.

4. Threshold-Curve Set

These parameters, which decide whether a template is regarded as a pedestrian template or not, have great influence on the accuracy of counting. Hence, in order to guarantee

FIGURE 11: (a) The scatter gram of the frequency of the template samples in x -axis. (b) Detail with enlarged scale of (a).

FIGURE 12: A new scatter gram of the frequency of the template samples in x -axis.

TABLE 2: Template initialization.

Character	Initialization
x	$x_{\text{detection}}$
y	$y_{\text{detection}}$
dx	$dx_{\text{detection}}$
dy	$dy_{\text{detection}}$
Width	Width _{detection}
Height	Height _{detection}
Score	1
Scoreperfrm	1
Scoremax	1
Frame	Frame _{detection}
Startfrm	Frame _{detection}
startx	$x_{\text{detection}}$
starty	$y_{\text{detection}}$
Label	A unique number

counting accuracy, a novel method to set these parameters' thresholds is presented. The novelty of this method is that these parameters' thresholds are set not by fixed values but by curve values. In other words, whether the template is regarded as a pedestrian or not is decided by different thresholds' values when it is in different positions of the

TABLE 3: All fields in a record.

Character	Explanation
Width, height, dx , dy , score, x , y , startx, starty, label, scoremax	These are shown in Section 2
Size	Width $\times$ height
Frames	Frame – startfrm + 1
Score ratio	Score/frames
dx scorediraver	$ dx /\text{frames}$
dy scorediraver	$ dy /\text{frames}$
x axis	$ x - \text{startx} $
y axis	$ y - \text{starty} $
x axisaver	$x\text{-axis}/\text{frames}$
y axisaver	$y\text{-axis}/\text{frames}$

frame image. This novel method's implementation includes two steps. The first step is data collection of the samples, and next step is to make an analysis of the data and calculate the threshold-curve.

4.1. Data Collection. In order to find out the different characteristics between the pedestrian template and other noise templates, a piece of software to collect pedestrian templates is designed. The piece of software is shown in Figure 9.

In this software, the display of the rectangular template and identification code are shown in the video clip in real time (see Figure 9). This software will pop up a new dialogue box named “dataForm” automatically while a template cannot match with a new detection rectangle for 5 frames. There are two buttons in this dialogue box—“Yes” and “No.” The relevant characteristics of this template will be added to the pedestrian template database if the “Yes” button is pressed. In this database, each field stands for a specific feature of the template and each record represents a pedestrian template. All fields are listed in Table 3.

In this paper, 908 pedestrian templates to make a more comprehensive analysis have been collected.

FIGURE 13: (a) MIN_SIZE threshold figure. (b) Detail with enlarged scale of (a).

4.2. Data Analysis. From what is presented above, we judge a template by whether a pedestrian uses the parameters of MIN_SIZE, MAX_SIZE, GOOD_SCORE, GOOD_SCORE_RATIO, GOOD_DIR_AVER, GOOD_SCORE_MAX, and Y_AXIS_MIN or not.

We find out that the counting error is bigger than expected if these parameters are set by fixed value. That is because template's detecting and tracking are susceptible to the surroundings. Therefore, a novel method which is to set the parameter with different values is proposed. The value is decided by the last position of the template. In other words, we want to get not a fixed value but a curve value to be the parameter's threshold, so we call it threshold-curve.

Figure 10 is the scatter gram of the frequency of the template samples. In this figure, colorful dots stand for the last position (the position where the template is deleted) of pedestrian templates which have been collected in Section 4.1, and the different colors represent the different frequency of their occurrences (red for 3 times, green for 2 times, and so on). It is observed that these dots' y-value is quite close and x-value is disperse, so we use these dots' x-value as the independent variable of the threshold-curve. To help readers better understand this new method, we will take the example of setting the threshold of MIN_SIZE to further explain how to set these thresholds' values.

4.2.1. MIN_SIZE Threshold-Curve. As is mentioned above, MIN_SIZE threshold-curve has been gotten according to x-values of the template samples. The x-value is divided into 704 groups as this video resolution is 704 * 576 pixels. Each group includes 576 pixels from $(1, y_n)$ to $(576, y_n)$. Figure 11 is the scatter gram of the frequency of the template samples in x-axis. The figures show that it is easy to make mistakes to adopt this method of classification, which are shown in Figure 11(b). In this figure, none of the x-values of the pedestrian templates comes to 55; however, it does not mean that a pedestrian will never appear at this position $(55, y_n)$. And events with low frequency may account for this error.

It is the most direct way to expand the number of the samples so as to avoid this mistake. However this solution

TABLE 4: Pixels in a big group.

$(1, y_n)$	$(1, y_{n+1})$	$(1, y_{n+2})$	$(1, y_{n+3})$
$(2, y_n)$	$(2, y_{n+1})$	$(2, y_{n+2})$	$(2, y_{n+3})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(576, y_n)$	$(576, y_{n+1})$	$(576, y_{n+2})$	$(576, y_{n+3})$

FIGURE 14: A new MIN_SIZE threshold figure.

is not feasible because more samples would increase the intensity of manual work. The solution we adopt is to integrate four small groups into a big group, each of which includes $4 * 576$ pixels. The detail is presented in Table 4.

A new scatter gram of the frequency of the template samples in x-axis is shown after using this method (see Figure 12).

The templates in a big group form a template set, each of which has the property of size. We will get a set of the templates by this way. In order to get the threshold of the MIN_SIZE parameter, we sort this template's size set from small to large and collect the 10~(th) percentile value of this set (see Figure 13(a)). The reason why 10~(th) percentile value is chosen instead of the smallest value lies in the principle of reducing the impact of events with low frequency.

Both the size value and the frequency distribution of the pedestrian templates have impacts on the threshold of

FIGURE 15: MIN_SIZE threshold-curve figure.

TABLE 5: Set MIN_SIZE threshold.

Group number	X-coordinate	Frequency	10~(th) percentile size value
35	137–140	5	1849
36	141–144	9	1764
37	145–148	2	3364
38	149–152	6	2809
39	153–156	6	1521

the MIN_SIZE. For example, in Table 5, when x -coordinate value equals 145–148, we consider that the value of the MIN_SIZE is a little larger than its actual value because its frequency is only 2 and the sizes nearby (1849, 1764, 2809, and 1521) are smaller. To guarantee the veracity of the threshold-curve, the following is used to calculate the MIN_SIZE value of the templates:

$$f_{\text{sum}} = f_{n-2}\gamma_2 + f_{n-1}\gamma_1 + f_n + f_{n+1}\gamma_1 + f_{n+2}\gamma_2,$$

$$\text{Update } S_n = \frac{\gamma_2 f_{n-2}}{f_{\text{sum}}} s_{n-2} + \frac{\gamma_1 f_{n-1}}{f_{\text{sum}}} s_{n-1} + \frac{f_n}{f_{\text{sum}}} s$$

$$+ \frac{\gamma_1 f_{n+1}}{f_{\text{sum}}} s_{n+1} + \frac{\gamma_2 f_{n+2}}{f_{\text{sum}}} s_{n+2}. \quad (5)$$

f_n stands for the frequency value of group n ; γ_1, γ_2 stand for weight value (in this paper, $\gamma_1 = 0.6, \gamma_2 = 0.4$); s_n stands for the size value of group n ; Update S_n stands for the new size value of group n . This method to get a new MIN_SIZE threshold figure is adopted (see Figure 14).

Now the discrete function of the thresholds of the MIN_SIZE has been got, but it is far from enough. To make it convenient to design of the program and boost the calculating speed, we intend to get a continuous function (threshold-curve) of the MIN_SIZE, which can be obtained by two steps.

Firstly, the discrete MIN_SIZE threshold data is adapted to a sum of the sin function ($f(x) = a_1 \sin(b_1 x + c_1) + a_2 \sin(b_2 x + c_2) + \dots + a_n \sin(b_n x + c_n)$) by fitting at least squares principle. As is known to all, the larger the n -value

becomes, the smaller the SSE-value (Sum of Squares for Error) becomes. However, it takes more time to calculate. Therefore, in order to work out a proper n -value that can make SSE-value and calculating speed in the ballpark, the value of n is increased one by one and the SSE-value of the fitting function is calculated. When SSE-value is less than $(0.07 \times \text{MIN_SIZE}_{\text{max}})^2 \times 704$ ($\text{MIN_SIZE}_{\text{max}}$ for the max value of MIN_SIZE), the value of n can be worked out.

Secondly, to make the threshold-curve more fault-tolerant, the fitting function is multiplied by a coefficient. If the lower limit value is needed, this coefficient should be 0.7; if the upper limit value is wanted, this coefficient should be 1.3. We will get the threshold-curve of the MIN_SIZE by this way.

The threshold-curve of the MIN_SIZE is shown in Figure 15 and the following formula:

$$\text{MIN_SIZE}(x) > 2056 \sin(0.004152x + 0.261) + 312.5 \sin(0.01739x + 2.857). \quad (6)$$

4.2.2. The Other Parameters' Threshold-Curve. The other parameters' (MAX_SIZE, GOOD_SCORE, GOOD_SCORE_RATIO, GOOD_DIR_AVER, GOOD_SCORE_MAX, and Y_AXIS_MIN) threshold-curve can be gotten like MIN_SIZE. These parameters' threshold-curve is worked out as in Figures 16, 17, 18, 19, 20, and 21 and the following:

$$\text{MAX_SIZE}(x) < 7947 \sin(0.004436x - 0.01695) + 1033 \sin(0.009265x + 0.3037) + 521 \sin(0.02501x + 0.2283), \quad (7)$$

$$\text{GOOD_SCORE}(x) > 8.687 \sin(0.004133x + 0.1825) + 1.727 \sin(0.01654x + 2.847) + 1.515 \sin(0.04421x - 1.328), \quad (8)$$

$$\text{GOOD_SCORE_RATIO}(x) > 0.7049 \sin(0.003123x + 0.6154) + 0.09464 \sin(0.01939x - 4.152), \quad (9)$$

$$\text{GOOD_DIR_AVER}(x) > 0.1152 \sin(0.006032x - 0.4753) + 0.07742 \sin(0.01011x + 1.219) + 0.02763 \sin(0.02273x - 0.04605) + 0.01126 \sin(0.05589x - 2.976) + 0.03896 \sin(0.08273x - 1.865) + 0.03893 \sin(0.08434x + 0.9109) + 0.01092 \sin(0.06136x - 1.576) + 0.02656 \sin(0.0232x + 2.896), \quad (10)$$

FIGURE 16: MAX_SIZE threshold-curve figure (see (7)).

FIGURE 18: GOOD_SCORE_RATIO threshold-curve figure (see (9)).

FIGURE 17: GOOD_SCORE threshold-curve figure (see (8)).

FIGURE 19: GOOD_DIR_AVER threshold-curve figure (see (10)).

$$\text{GOOD_SCORE_MAX}(x) > 1, \quad (11)$$

$$\begin{aligned} Y_AXIS_MIN(x) &> 56.84 \sin(0.004523x + 0.09802) \\ &+ 11.02 \sin(0.01791x + 2.443) \\ &+ 188 \sin(0.0426x - 0.538) \\ &+ 7.161 \sin(0.02761x + 2.556) \\ &+ 3.917 \sin(0.153x + 1.762) \\ &+ 3.448 \sin(0.0894x - 2.319) \\ &+ 2.787 \sin(0.1779x + 2.449) \\ &+ 181.7 \sin(0.04248x - 3.621). \end{aligned} \quad (12)$$

5. Results

To test the results of our method, we design a piece of software named Pedestrian Counting Software V1.0. This software has been developed by C# language and EmguCV library. EmguCV is the NET version of OpenCV which is an open source computer vision library developed by Intel that

TABLE 6: The fixed threshold we use.

Parameter	Fixed threshold
MIN.SIZE	729
MAX.SIZE	11664
GOOD_SCORE	2
GOOD_SCORE_RATIO	0.3333
GOOD_DIR_AVER	0.1945
GOOD_SCORE_MAX	1
Y_AXIS_MIN	0

includes a lot of general algorithms for image processing and computer vision.

The first experiment is to test the accuracy of counting pedestrian with fixed threshold. The value of parameters' thresholds is decided by the pedestrian template set we collect in Section 4.1. For example, the value of MIN.SIZE's threshold is the minimum value of template set. The values we use in this paper are as in Table 6.

The accuracy results of counting pedestrian with fixed threshold are as in Table 7.

FIGURE 20: GOOD_SCORE_MAX threshold-curve figure (see (11)).

FIGURE 21: Y_AXIS_MIN threshold-curve figure (see (12)).

TABLE 7: Experimental results with fixed threshold.

	Up	Down
Actual data	1749	21
Missed	411	1
Mistake	91	0
Counting data	1429	20
Accuracy	81.7%	95.2%
Actual accuracy	71.3%	95.2%

And then we test the accuracy of counting pedestrian with threshold-curve. The value of parameters' thresholds we use is from Section 4.2. The accuracy results are as in Table 8.

Compared with using fixed threshold, using threshold-curve has made a performance on counting accuracy. As seen from Tables 6, 7, and 8, the method with threshold-curve improves counting accuracy by 13.9 percent and actual accuracy by 17.7 percent compared to the method with fixed threshold; hence the method using threshold-curve is much better. In addition, the accuracy is the same whether we use fixed threshold or use threshold-curve. That is because down passenger flow is very little and it is difficult to calculate

TABLE 8: Experimental results with threshold-curve.

	Up	Down
Actual data	1749	21
Missed	135	1
Mistake	58	0
Counting data	1672	20
Accuracy	95.6%	95.2%
Actual accuracy	89.0%	95.2%

FIGURE 22: Pedestrian counting software.

threshold-curve. Therefore, we use fixed threshold instead of threshold-curve to count down passenger flow.

Although the method we present in this paper makes a good performance on counting pedestrian, counting errors like mistake and missing still exist. There are two reasons of missing counting. One is due to pedestrian wearing hats with a similar color to the surrounding. The other is pedestrian's fast speed of running. And the reason causing mistake counting is that pedestrian luggage is similar to pedestrian head.

The test video clip has been taken from Xizhimen metro station (see Figure 22). It is composed of 90000 frames, for a total amount of one hour. Its frame size is $704 * 576$ pixels and frame rate is 25 (one frame is 40 ms). The CPU we use in experiment is Intel(R) Core(TM) 2 Duo with 3.0 GHz and the software needs about 27 ms to process one frame image. Hence, this method can meet the demands of real-time processing.

6. Conclusion

This paper proposes a novel method to count pedestrians in metro station based on Haar-like detection and template-matching algorithm. This new method avoids the disadvantages of other computer vision methods in short-shot and nondirect view. Thus there are the following two points between our method and the other traditional methods.

(1) To avoid counting error due to occlusions and body-shape deformation in short-shot and nondirect view, this method uses Haar-like and AdaBoost algorithms which are mature in face recognition skill to detect pedestrian.

(2) Template-matching algorithm which includes seven parameters (MIN_SIZE, MAX_SIZE, GOOD_SCORE, GOOD_SCORE_RATIO, GOOD_DIR_AVER, GOOD_SCORE_MAX, and Y_AXIS_MIN) is presented in our method to track and count pedestrian. The value of these parameters is the key to judge a template whether a pedestrian or not; thus it has a great influence on the accuracy of counting pedestrian. Nevertheless, we found that the accuracy results cannot meet our demands if we set these parameters with the fixed value. Therefore, a novel method to set these parameters with the threshold-curve is proposed. In other words, a threshold function is built to every parameter so as to improve the accuracy of pedestrian counting.

The experiments prove that our method can make a good performance on counting application. And the accuracy of the method with threshold-curve is nearly ninety percent which is higher than that of the fixed threshold method's seventeen percent, so the method with threshold-curve is proved to be a better method.

The method presented in this paper is of practicable value to apply in a metro station, but there are also two disadvantages to this method. One is that a lot of workload is needed to collect template samples, so we expect to adopt some automatic collection methods to collect the samples instead of the manual collection method in the future. The other is that we must judge a template whether a pedestrian or not manually when we collect template samples; thus an automatic method to judge the template's property is expected to be proposed in the future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This study is supported by the Science and Technology Project of Ministry of Transport: The Study on Key Technology of Transfer Information Sensing and Co-Scheduling, no. 2012-364-220-109; and Science and Technology Planning Project of Beijing Science and Technology Commission: The Study on the Mechanism of Travel Behavior and Guiding Strategy, no. Z1211000003120100.

References

- [1] A. Mittal, A. Jain, and G. K. Agarwal, "Audio-video based people counting and security framework for traffic crossings," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 49, no. 3, pp. 377–391, 2007.
- [2] Y. Hou and G. K. H. Pang, "People counting and human detection in a challenging situation," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 41, no. 1, pp. 24–33, 2011.
- [3] A. B. Chan and N. Vasconcelos, "Counting people with low-level features and Bayesian regression," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2160–2177, 2012.
- [4] G. Xiong, J. Cheng, X. Wu, Y. Chen, Y. Ou, and Y. Xu, "An energy model approach to people counting for abnormal crowd behavior detection," *Neurocomputing*, vol. 83, pp. 121–135, 2012.
- [5] C. Sacchi, G. Gera, L. Marcenaro, and C. S. Regazzoni, "Advanced image-processing tools for counting people in tourist site-monitoring applications," *Signal Processing*, vol. 81, no. 5, pp. 1017–1040, 2001.
- [6] A. J. Schofield, T. J. Stonham, and P. A. Mehta, "Automated people counting to aid lift control," *Automation in Construction*, vol. 6, no. 5–6, pp. 437–445, 1997.
- [7] K. Hashimoto, C. Kawaguchi, S. Matsueda, K. Morinaka, and N. Yoshiike, "People-counting system using multisensing application," *Sensors and Actuators A*, vol. 66, no. 1–3, pp. 50–55, 1998.
- [8] I. J. Amin, A. J. Taylor, F. Junejo, A. Al-Habaibeh, and R. M. Parkin, "Automated people-counting by using low-resolution infrared and visual cameras," *Measurement*, vol. 41, no. 6, pp. 589–599, 2008.
- [9] D. Huang and T. W. S. Chow, "A people-counting system using a hybrid RBF neural network," *Neural Processing Letters*, vol. 18, no. 2, pp. 97–113, 2003.
- [10] A. J. Schofield, P. A. Mehta, and T. J. Stonham, "A system for counting people in video images using neural networks to identify the background scene," *Pattern Recognition*, vol. 29, no. 8, pp. 1421–1428, 1996.
- [11] K. Kopaczewski, M. Szczodrak, A. Czyzewski, and H. Krawczyk, "A method for counting people attending large public events," *Multimedia Tools and Applications*, 2013.
- [12] A. G. Vicente, I. B. Munoz, P. J. Molina, and J. L. L. Galilea, "Embedded vision modules for tracking and counting people," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 9, pp. 3004–3011, 2009.
- [13] D. Conte, P. Foggia, G. Percannella, F. Tufano, and M. Vento, "A method for counting moving people in video surveillance videos," *Eurasip Journal on Advances in Signal Processing*, vol. 2010, Article ID 231240, 10 pages, 2010.
- [14] M. Goncalo, P. Paulo, and N. Urbano, "Vision-based pedestrian detection using Haar-like features," in *Actas do Encontro Científico Guimarães*, pp. 45–50, April 2006.
- [15] A. A. Shaikh, D. K. Kumar, and J. Gubbi, "Automatic visual speech segmentation and recognition using directional motion history images and Zernike moments," *The Visual Computer*, vol. 29, no. 10, pp. 969–982, 2013.
- [16] T. Pallejà, A. Guillaumet, M. Tresanchez et al., "Implementation of a robust absolute virtual head mouse combining face detection, template matching and optical flow algorithms," *Telecommunication Systems*, vol. 52, no. 3, pp. 1479–1489, 2013.