2014 IEEE 17th International Conference on
Intelligent Transportation Systems (ITSC)
October 8-11, 2014. Qingdao, China

Research on Passenger Flow Counting based on Embedded System

Bin Hu, Gang Xiong, Yiyue Li, Zizhang Chen, Weisi Zhou, Xin Wang, Qiang Wang

Abstract—**Passenger flow counting is an important issue for management and decision in many public areas such as shopping centers, airports, gyms and galleries. This paper presents some algorithms of Head-counting and Head-tracking for embedded systems in vertically-installed cameras. The system is able to count the number of passengers who enter or leave the detection area. The results are reliable, accurate and stable.**

I. INTRODUCTION

With the development of software and hardware, Intelligent Video Surveillance (IVS) system is broadly used in many different areas. Value-added services on traditional IVS system become hotspots. In many IVS systems, we need some Functions such as Passenger flow counting, Passenger density estimation, Passenger activity recognition and Passenger motion pattern estimation. Passenger flow counting and Passenger flow statistics both fall under the category of intelligent video analysis, providing supports for various businesses.

The passenger flow counting system has many advantages. It is fully automatic and can achieve accurate results in all kind of circumstances by using advanced algorithm. However, its robustness and stability need to depend on the hardware performance. The head-like ellipse cannot represent human head perfectly. Also the color feature detection and the edge feature detection used are complex and slow especially when the background looks similar to the target. This algorithm uses pre-trained Adaboost classifier [1] together with motion detector to achieve passenger flow counting.

II. FOREGROUND DETECTION

Foreground detection algorithm is able to extract moving objects in the detection area, so that less calculation for the following head detection is needed. Common foreground detection methods are Optical flow [2], Frame subtraction and Background subtraction. This paper suggests using Frame subtraction to extract the foreground to determine the detection area in order to reduce calculations.

Bin Hu is with the Qingdao Academy of Intelligent Industries, Institute of Automation, Chinese Academy of Sciences. (The corresponding author is Bin Hu: binhu@ia.ac.cn)

Gang Xiong, Yiyue Li, Zizhang Chen, Weisi Zhou are with the Cloud Computing Center, Chinese Academy of Sciences Songshan Lake, Dongguan, 523808, China

Xin Wang, Qiang Wang are with the Shenzhen Jiaxinjie Technology Co., LTD., Shenzhen, 518102, China

The image size is adjusted to 640*360 to increase the process efficiency. Then, to remove the 'Double residual image' effect, three frames with a time interval of N-frames are differentiated twice to obtain two gradient images. These two gradient images are then binarized by a threshold and compared to obtain the final foreground.

Foreground image $D_k(x,y)$ can be expressed as:

$$D_k(x,y)=\begin{cases}1 & \text{if } |I_k(x,y)-I_{k-n}(x,y)|>Th \text{ and } |I_k(x,y)-I_{k-2n}(x,y)|>Th \\ 0 & otherwise\end{cases} \quad (1)$$

where $I_k(x,y)$, $I_{k-n}(x,y)$, $I_{k-2n}(x,y)$ are the gray-scale images of k^{th}, $k\text{-}n^{th}$, $k\text{-}2n^{th}$ frames respectively. *Th* is the binary threshold.

Median filter and morphological filter are then applied to remove noises and disturbances on the foreground image. As shown in Fig.1, extracting the area of the moving target, which is the passenger in the image, can reduce the head detection calculations.

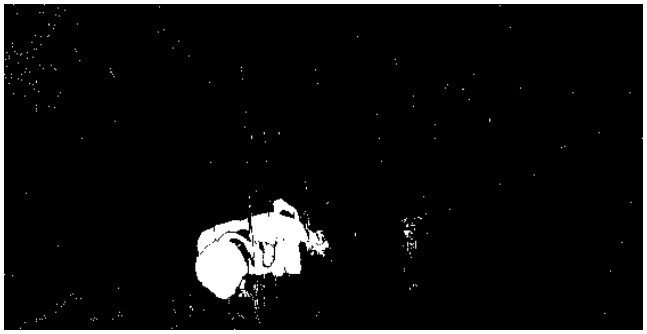


Figure 1: Foreground detection

III. HEAD DETECTION

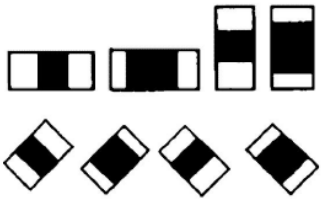
Once there is a foreground area appears in the image, a pre-trained sliding window detector is needed to detect the human head in the foreground area.

Adaboost is the head detection classifier. It is a common classifier proposed by Freund and Schapire in 1995. It can form strong classifiers by combining several week classifiers. And the results are normally fast, stable and accurate.

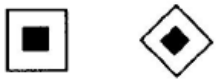
In order to detect whether the foreground area has a head, the classifiers need to build models containing several head features. Harr feature [3] is one of the common features used in object detection, suggested by Papageorgiou and improved by Viola and Jones. It uses more than three features at the same time. Haar has three main features: edge feature, linear feature, central diagonal line feature. As shown in Fig.2: the

feature models are black and white rectangles. The Feature model score of each model is the difference between the sum of the black pixels and the sum of the white pixels. Then the Harr feature score is a combination of the Feature model scores, the coordinates and the size of the rectangles. The rectangles can be anywhere at any size on the image, so a large number of rectangular detection windows are formed. In Adaboost classifier, each detection window represents a weak classifier. The weak classifiers are trained from weak to strong by iteration. In this way, strong classifiers are built.

Edge features



Linear features



Central diagonal line features

Figure 2: Image features

Finally, the Adaboost classifier is formed from several strong classifiers, which are built up by many weak classifiers. Efficient classifiers are used first to eliminate most useless windows and then complex classifiers are applied to classify the rest in detailed. As a result, most useless windows with no head in the foreground area are filtered at the beginning and the others are classified accurately. The process speed can be largely increased.

The head detection has two steps: training and detecting. In the training process, a large amount of positive and negative samples are extracted to train weak classifiers. However, due to the large size of the training samples, training process can be relatively slow. Therefore, the weak classifiers are trained before detecting process so that it increases the detection speed as shown in Fig. 3.

A large amount of positive and negative samples are needed to ensure the performance of the Adaboost classifier. These samples are extracted from images in various environments. 1300 positive samples and 600 negative samples are extracted and adjusted into a size of 30*30 pixels to train the classifiers.

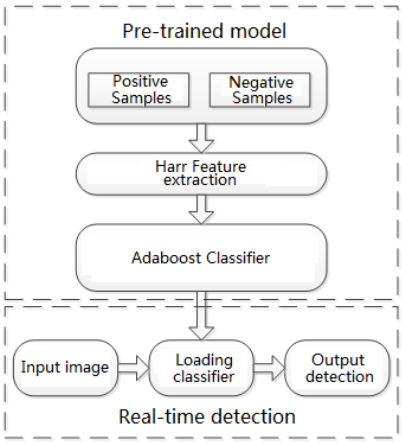


Figure 3: Pre-trained Model

Positive samples

Negative samples

Figure 4: Positive and negative samples

After the Adaboost classifier is trained, it can be used to detect the object in the detection area. Normally the classifier window slides through the detection area in different sizes. In order to optimize the speed and the accuracy, the classifier window is enlarged at a certain rate. If the enlarge rate is too small, there will be too many classifier windows. A large amount of calculation is needed and the detection speed will be slow. However, if the enlarge rate is too large, the middle-size targets will not be detected. In this particular case, the best enlarge rate is 1.2.

There are many overlapping detected windows as detection results. However, only one of them is needed. Therefore, the overlapping detected windows should be reconciled.

First, the overlapping area and the number of overlapping windows are calculated. If in certain area, the overlapping area and the number of overlapping windows satisfy the thresholds *T1* and *T2* respectively. These overlapping windows are regarded as one detected target. Otherwise, the windows are regarded as false positive results.

Then the overlapping windows are combined. The best way achieve this is to calculate the averages of the coordinates and the sizes of the windows. Finally the detected target is

recorded into a target list.

IV. TRACKING AND COUNTING TARGETS

The target can be assumed to be in uniform linear motion in consecutive frames. Assume there is only Gaussian noise in the image. In order to track the target, Kalman filter [4][5] and Mean shift [6] are used to improve the performance of the algorithm. The coordinates of the target in each frame are matched with the one in the last frame. If they match successfully, the target coordinates are then updated. Otherwise the target coordinates are predicted and adjusted by Kalman filter.

Kalman filter is an iteration method to estimate the next state by minimizing the mean square error. It can also update the current estimation from the last estimation and the current observation, so it does not rely on any historical information. Unlike other low-pass filter or frequency domain filter, it is a time domain filter.

Kalman filter estimates the state at time K from the state at time K-1 by:

$$X_k = F_k X_{k-1} + B_k u_k + w_k \tag{2}$$

where F_k is the transformation model of X_{k-1}, B_k is a factor of input u_k and w_k is the noise. Also the observation model can be expressed as:

$$z_k = H_k X_k + v_k \tag{3}$$

where H_k is the observation model for transforming the states into observations. v_k is the zero-mean noise. Noise $\{w_0, w_1, \ldots w_k, v_0, v_1 \ldots v_k\}$ are independent to each other at any time.

From the uniform linear motion equations in 2-dimensions, the state transformation matrix and the observation transformation matrix of the Kalman filter can be expressed as:

$$X_k = \begin{bmatrix} x_k \\ y_k \\ x'_k \\ y'_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ x'_{k-1} \\ y'_{k-1} \end{bmatrix} + w_k \tag{4}$$

$$z_k = \begin{bmatrix} z_{xk} \\ z_{yk} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ x'_k \\ y'_k \end{bmatrix} + v_k \tag{5}$$

where $X_{k|k}$ and $P_{k|k}$ express the states. $X_{k|k}$ is the estimation at time k and $P_{k|k}$ is the noise covariant.

Kalman filter has two main processes. The first one is estimation, in which the next state is estimated by the current state. And the other one is update, that is, the Kalman filter obtains an estimated value and Mean shift obtains an observation value. Finally the two values are compared and optimized.

There are situations that the target overlaps during the tracking and updating process. Therefore, the overlapping rectangles need to be reconciled. First, for the overlapping rectangles, the overlapping area *Crossarea* is calculated. If $Crossarea > RectArea_{min} \times thD$, the overlapping rectangles will be combined. Where, in the formula, $RectArea_{min}$ is the area of each rectangle and *thD* is the area threshold. Otherwise the rectangles will be deleted as false positive results. This process is repeated until the target splits into two again.

The passenger flow counting can be achieved by counting the number of targets crossing a boundary. When a target crosses the boundary, the total distance *S* and the motion direction Θ are calculated. If *S* is larger than the threshold and Θ satisfy the direction condition, the target is regarded as entering the detection area and *EntranceValue* +1, otherwise, the *ExitValue* +1. Then the target is removed from the tracking list to prevent counting it twice. Some example results are shown in Fig. 5.

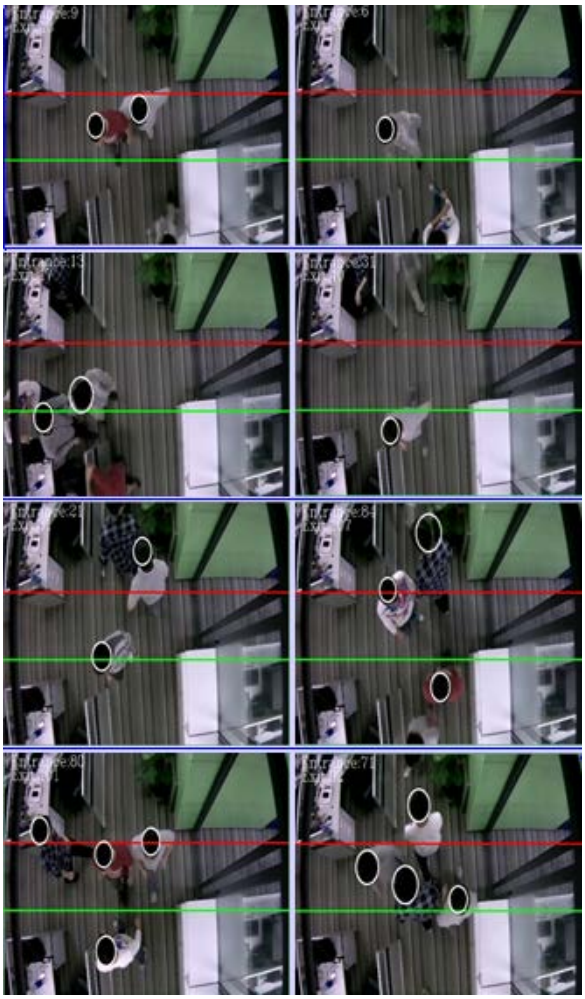


Figure 5: detection results

V. EMBEDDED SYSTEM

The algorithms in this paper are all achieved on network cameras. For the hardware, the system uses Haisi Hi3517SOC embedded platform, 3517 high resolution 1080P@60fps video input and coding system, ARM Cortex A9Processor with dominant frequency 800MHz and hardware process accelerator modulus.

For the software, image processing, feature extraction, classifier training and target tracking algorithm based on OpenCV are used. OpenCV is a multi-platform visual software package developed by Intel.

The embedded system is installed vertically above to overcome sheltering and overlapping problems. Wide-angle lens are used to enlarge the detection area. Images are captured on Linux platform and loaded through ISP optimization from DMA (DirectMemoryAccess). Then SOC hardware is used directly to change the RGB image into gray-scale image.

The passenger flow counting is achieved in the embedded system in the cameras. Therefore, no calculation is needed to analysis the videos in the servers. Also, the cameras can be installed anywhere as many as possible. Fig. 6 shows the overall structure of the passenger flow counting system.

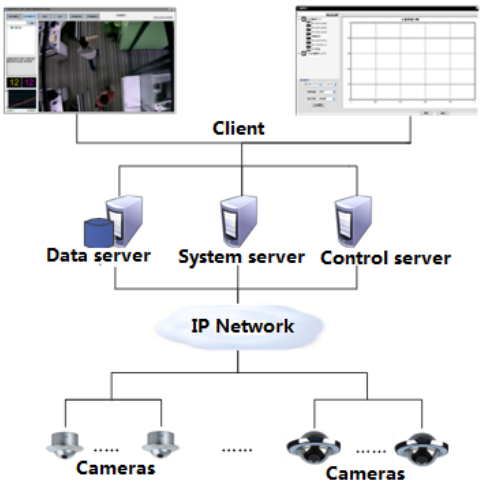


Figure 6: Passenger flow counting system

VI. CONCLUSION

The accuracy of the passenger flow counting system is directly relative to the accuracy of the target tracking algorithm. This paper uses machine learning to train well-performed classifiers, together with Kalman filter and mean shift to track the moving targets. It effectively solves the resolution, miscounting and interference problems. Also the calculation is fast and reliable in all kind of circumstances. The results show excellent instantaneous and accurate.

VII. ACKNOWLEDGMENT

This work is supported in part by Chinese MoT's S&T project (2012-364-221-108; 2012-364-221-101). Chinese MIIT's IoT development special fund project: Real time image recognition technology and application system for social security. Guangdong's S&T project (2012B091100213; 2012B090400004). Dongguan's Innovation Talents Project (Gang Xiong).

VIII. REFERENCES

[1]. Freund Y， Schapire R E. A Decision-theoretic Feneralization of Online Learning and an Application to Boosting[J]. Journal of Computer and System Sciences, 1997,55(1):119-139.

[2]. Horn B K P, Schunck B G, Determining optical flow. [C] Artificial Intelligence 17, 1981:185-203.

[3]. Lienhart R, Maydt J. An extended set of Haar-like features for rapid object detection[J]. IEEE,2002(1):900-903.

[4]. Raman K Mehra. On the identification of Variances and Adaptive Kalman filtering[J].IEEE Trans on Aut0matic Control (S0018-9286),1970,15(2): 175-184.

[5]. Wan Qin, Wang Yao-nan. Moving Objects Detecting and Tracking Based on Kalman Filter[J]. Journal of Hunan University (Natural Sciences), 2007, 34(3):36-40

[6]. COMICIUD,RAMESHV, MEER P. Real-time tracking of non-rigid objects using mean shift[C] Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2000:142-149.