An Automated Vehicle Counting System for Traffic Surveillance

Kunfeng Wang, Zhenjiang Li, Qingming Yao, Wuling Huang, and Fei-Yue Wang, *Fellow, IEEE*

Abstract— **The objective of this paper is to present a detailed description of using DSP board and image processing techniques to construct an automated vehicle counting system. Such a system has many potential applications, such as traffic signal control and district traffic abduction. We use TI TMS320DM642 DSP as the computational unit to avoid heavy investment in industrial control computer while obtaining improved computational power and optimized system structure. The overall software is comprised of two parts: embedded DSP software and host PC software. The embedded DSP software acquires the video image from stationary cameras, detects and counts moving vehicles, and transmits the processing results and real-time images after compression to PC software through network. The host PC software works as a graphic user interface through which the end user can configure the DSP board parameters and access the video processing results. The vehicle detection and counting algorithm is carefully devised to keep robust and efficient in traffic scenes for longtime span and with changeful illumination. Experimental results show that the proposed system performs well in actual traffic scenes, and the processing speed and accuracy of the system can meet the requirement of practical applications.**

I. Introduction

INTELLIGENT Transportation System (ITS) has become an active research area where artificial intelligence techniques are extensively applied to enhance safety and effectiveness on the road. Among a large number of related issues, current technology involving magnetic loop detectors buried in the road provides limited traffic information and is expensive to install and maintain. It is widely recognized that vision-based systems are flexible and versatile in traffic monitoring applications if they can be made sufficiently reliable and robust [1]-[4].

A video processing system becomes even more applicable and affordable if it works on an embedded platform [5]. In approaching the design of a video-based vehicle counting system, we want it to work in real-time on a standard embedded platform and for longtime span in outdoor environments. This aim implies that very reliable methods must be utilized to endow the system with flexibility and generality, while at the same time the hardware structure and software algorithm must meet the efficiency requirements.

Manuscript received June 11, 2007. This work was supported in part by the MOST Grant 2006CB705500, NNSFC Grants 60334020, 60621001, CAS Grant 2F05N01, 2006 CAS Presidential Special Scholarship Grant, and SDAS Research Grant 2005006.

Kunfeng Wang, Zhenjiang Li, Qingming Yao, and Wuling Huang are with the Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China (e-mail: kunfengwang@gmail.com; zhenjiang.li@mail.ia.ac.cn; qingmingyao@gmail.com; wuling.huang@ia.ac.cn).

Fei-Yue Wang is with the Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing 10080, China, and also with the Department of Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85719, USA (e-mail: feiyue@sie.arizona.edu).

This paper presents an automated vehicle counting system based on TI TMS320DM642 DSP. The video input, image processing, video output and networking protocol are all realized on the DSP board. The host PC software is used to configure the DSP board parameters and access the video processing results. To detect moving vehicles, the end user need first set a few virtual loops as detection zones in the video image. The vehicle counting algorithm is based on the virtual loops and the counting results could be used in at least two manners. The first manner is that Intersection Signal Controller (ISC) could adaptively adjust the intersection signal lights according to vehicle density of each driving direction to the intersection. Another manner is that Traffic Management Centre (TMC) could collect vehicle density of a large number of roadways and release traffic abduction information to all drivers.

The remainder of this paper is organized as follows. Section II describes the hardware platform and software structure of the vehicle counting system; Section III presents the image processing algorithm implemented on the embedded platform; Experimental results are discussed in Section IV, followed by a conclusion in Section V.

II. Hardware Platform and Software Structure

For the development of an efficient image processing system, reliable and durable hardware platform is the first consideration. Aiming at the application of real-time traffic data collection, the system must perform well in computation power. Therefore, we paid special attention to finding some image processors which have powerful processing capability (including software and hardware two aspects) to complete complex image processing tasks. Two of adoptable techniques may be parallel processing and pipelining. Based on these considerations, we chose DSP as the computational unit to avoid heavy investment in industrial control computer while computer power is improved and the system structure is better optimized than on traditional PC platform.

A. System Overview

The system is made up of three parts: video camera, embedded DSP board, and host PC. The camera is fixed on polls or other tall structures to overlook the traffic scene. The DSP board acquires the video from the camera, captures images from the video, detects and counts moving vehicles from sequential images, and then sends the counting results and real-time images after compression to host PC through network. The host PC works as a graphic user interface through which the end user can configure the DSP board parameters, access traffic video processing results, and save information into database. Fig. 1 depicts the general architecture of the automated vehicle counting system.

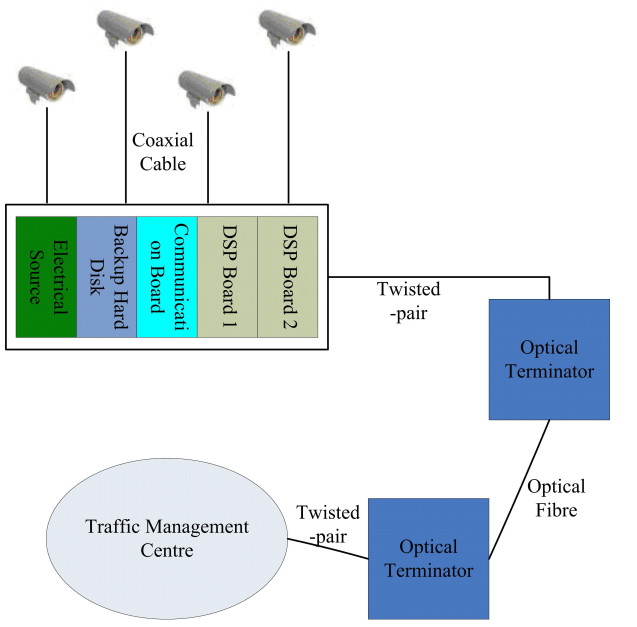


Fig. 1. The general architecture of the automated vehicle counting system.

There are two DSP boards defaulted in the system. Each DSP board handles two video inputs which correspond to two traffic scenes. The traffic scene may be intersection or road section. Here assume a four-highway-intersection is under consideration. If there are more or less highways, just add on or cut off suitable DSP boards. This doesn't make any trouble because the communication board could extend data transmission easily. The communication board transmits the image processing results to the TMC through Ethernet port.

B. Hardware Platform

Fig. 2 shows the reduced structure of DSP board which is the main computational device in the vehicle counting system. The core of the DSP board is a high-performance fixed-point digital signal processor, i.e., TI TMS320DM642 DSP. The DM642 DSP is based on the second-generation high-performance advanced VelociTI^{TM} VLIW architecture developed by Texas Instrument Corporation (TI), U.S., and is an excellent choice for digital signal processing tasks. With performance of up to 4800 million instructions per second (MIPS) at a clock rate of 600 MHz, the DM642 system is able to offer cost-effective solutions to real-time signal processing challenges.

The key parts of the DSP board include one DM642 DSP, one FPGA, two video decoders, one video encoder, two SDRAMs, and one Flash memory. The decoder chip (SAA7115HL) converts the analog video input (PAL format) to YUV422 digital image, and the encoder chip (SAA7105HL) converts the YUV422 digital image after processing to the analog video output (PAL format) for onsite display. The FPGA chip performs on screen display (OSD) function as a coordinated processor. The DSP board connects with the communication board through a local area network, and the communication board communicates with the TMC through Ethernet port.

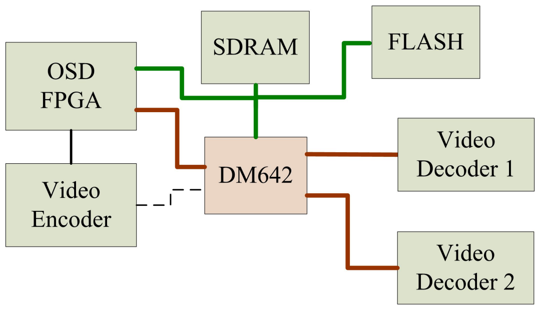


Fig. 2. Hardware structure of the DSP board.

C. Software Structure

The overall software is comprised of two main parts: embedded DSP software and host PC software. As depicted in Fig. 3, the embedded DSP software can be described as a multilayer structure, i.e., hardware layer, DSP/BIOS OS layer, APIs layer and application layer. DSP/BIOS is a scalable real-time OS kernel which is designed for applications that require real-time scheduling and synchronization, host-to-target communication, or real-time instrumentation. DSP/BIOS provides preemptive multi-threading, hardware abstraction, real-time analysis, and configuration tools. The APIs include Image Processing Library, Network Development Kit, and some other libraries offered by the third-party of TI. The application is developed by us to complete video processing, vehicle counting, networking communication, and some other tasks.

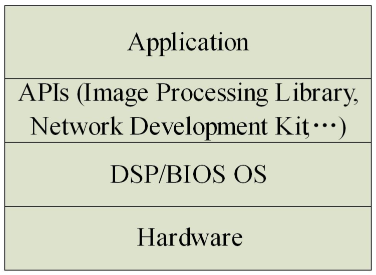


Fig. 3. Software structure of the DSP board.

The host PC software acts as a graphic user interface in the TMC. By virtue of the PC software, the user in the office can conveniently access and control every DSP board through network, for instance, the user can view real-time video images, configure image processing parameters related to the DSP board, and save image processing results into database. Some key DSP board parameters include: IP address, traffic scene description superimposed on video images, virtual loop positions, vehicle counting period, and so on.

III. Image Processing Algorithm

In this section we detail the image processing algorithm used for automated vehicle counting. Some important problems include virtual loop setting, background model maintenance, disturbance removal under daytime and nighttime, switch detection between daytime and nighttime, vehicle detection and counting, and so on.

A. Virtual Loop Setting

The proposed system employs a loop-based approach to detect and count moving vehicles in the scene. In virtue of the PC software, the user is able to view the real-time image sequence and define a set of regions of interest (ROI) in a video image. Each ROI is denoted as a virtual loop as large as a common car projected onto the corresponding location in the image (see Fig. 4). The ROIs are laid out to facilitate vehicle detection and may be linked using average operators to make the counting more accurate. For instance, the vehicle counting results of ROI 1 and 7 in Fig. 4 can be averaged to express the flow of the bottommost lane.

Loop-based vehicle detection methods mainly have two advantages. First, only ROIs in the image are processed, as to reduce the computation load; second, object tracking, occlusion handling, and some other complex processing steps are not required to count vehicles. On the other hand, it is clear that the major disadvantage of such methods is their limited monitoring ability due to reduced processing areas.

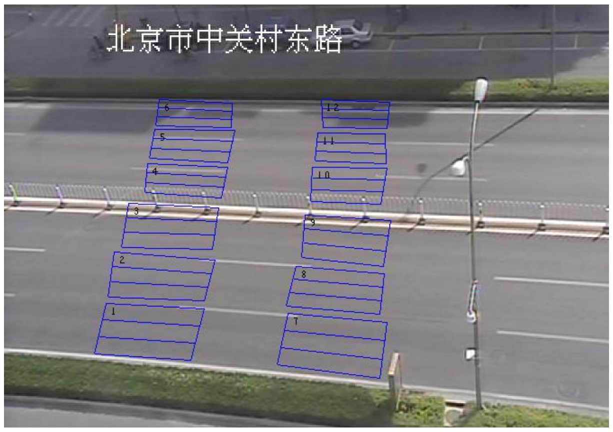


Fig. 4. Representative ROIs in the image.

B. Background Model Maintenance

Background subtraction is widely used to detect moving objects and even temporarily stopped objects under stationary camera situations. Here the focus is turned to maintaining the background model under complicated traffic scenes. An adaptive Gaussian mixture model method [6] is utilized in the system. Each pixel in the ROIs is modeled as a mixture of Gaussian distributions based on its RGB components. Each distribution is characterized by a mean vector μ, a covariance matrix Σ, and a weight ω representative of the frequency of occurrence of the distribution. The Gaussian distributions are sorted in the order of the most common distribution to the least common distribution. Pixels which match the most common distribution are classified as background while the rest are classified as foreground. In the system the learning rate α is set as 0.025, as to make the system able to learn a new background in just a few seconds.

C. Disturbance Removal

In traffic scenes, a foreground point detected through background subtraction may belong to one of four classes of objects: the actual moving object, the moving shadow, the illumination noise, and the non-illumination noise. The moving shadow is due to the moving object between the light source and the background scene. The shadow can be penumbra (soft shadow), umbra (dark shadow), or both [2]. The illumination noise may be caused by sudden illumination changes, the reflection of moving vehicles onto the glazed road surface, or the headlight of vehicles at nighttime. The non-illumination noise may be caused by camera jitter, swaying trees, system circuit noise, or transmission noise. We remove the moving shadow and the illumination noise in this part, and reserve the non-illumination noise removal problem until Part E.

As for moving shadow and illumination noise, it is assumed that there is negligible change in the chromaticity relative to the background [7]-[9]. In general, this assumption is reliable in slightly shadowed regions, such as illumination noise and weak shadow regions. On the other hand, we found in practice that most illumination noises, e.g., those caused by sudden illumination changes and the reflection of moving vehicles onto the road surface, do not change the background chromaticity distinctly. Therefore, we remove weak shadow and illumination noise based on photometric color invariants. Referring to [7], we simplify and redefine the $c_1c_2c_3$ color invariants as

$$c_1(x,y) = \frac{R(x,y)}{\max(G(x,y),B(x,y))} \tag{1}$$

$$c_2(x,y) = \frac{G(x,y)}{\max(R(x,y),B(x,y))} \tag{2}$$

$$c_3(x,y) = \frac{B(x,y)}{\max(R(x,y),G(x,y))} \tag{3}$$

where $R(x,y)$, $G(x,y)$, and $B(x,y)$ represent the red, green, and blue components of a pixel point in the image.

Generally, weak shadow darkens the scene surface slightly, while illumination noise does not change the background luminance much under daylight, and illumination noise (caused by the vehicle headlights) may lighten the background scene at nighttime.

Let $I(x,y)$ and $I_B(x,y)$ be the intensities at pixel point (x,y) of the input image and the background image, vector $C(x,y) = (c_1(x,y),c_2(x,y),c_3(x,y))^T$ denote the color invariants at point (x,y) of the input image, and vector $C_B(x,y) = (c_{1,B}(x,y),c_{2,B}(x,y),c_{3,B}(x,y))^T$ denote the color invariants at point (x,y) of the background image. Based on above analysis, we discard from foreground mask pixel points which satisfy the following constraints

$$\begin{cases} 0.8 < \dfrac{I(x,y)}{I_B(x,y)} < 1.2, \text{ at daytime} \\ 0.8 < \dfrac{I(x,y)}{I_B(x,y)}, \text{ at nighttime} \end{cases} \quad (4)$$

$$\frac{|C(x,y) - C_B(x,y)|}{|C(x,y) + C_B(x,y)|} < 0.05 \quad (5)$$

In umbra regions, the direct light is totally blocked, as to make color information in these regions lost. Thus, it is dangerous to assume that chromaticity features of umbra points do not change. However, umbra points have some important characteristics to label themselves: 1) Umbra points have lower color values than the corresponding background points; 2) Under daylight the blue component of umbra points increases, meanwhile at nighttime the red component of umbra points decreases. The former is because umbra regions are illuminated by the sky which is assumed to be blue at daytime, and the latter is because umbra regions are blocked by objects to see the orange street lamps which is the main light source at night; 3) Umbra regions satisfy relatively loose chromaticity invariability; 4) The appearance of umbra is independent of the object that is casting it and is dependent of the diffuse light and the surface material and direction.

According to umbra characteristics, pixels that satisfy the following constraints are labeled as possible umbra points:

$$r < R \text{ and } g < G \text{ and } b < B \quad (6)$$

$$\begin{cases} \dfrac{b}{r+g+b} > \dfrac{B}{R+G+B}, \text{ at daytime} \\ \dfrac{r}{r+g+b} < \dfrac{R}{R+G+B}, \text{ at nighttime} \end{cases} \quad (7)$$

$$\frac{|C(x,y) - C_B(x,y)|}{|C(x,y) + C_B(x,y)|} < 0.1 \quad (8)$$

Expressions (6)-(8) are not directly used to detect umbra in order to minimize the *false positives (FP)*. Instead, we consider the value of an umbra pixel over time as a pixel process and build adaptive umbra mixture models for pixel values that satisfy (6)-(8). Each pixel in the image is modeled as a mixture of Gaussian distributions based on its RGB components which satisfy (6)-(8) with time. Each distribution is characterized by a mean vector μ_U, a covariance matrix Σ_U, and a weight ω_U representative of the percentage of occurrence of the distribution. The Gaussian distributions are sorted in the order of the most frequent to the least frequent distribution. The probability that a pixel with RGB color value X belongs to the umbra model can be computed as

$$\begin{aligned} P(X) &= \sum_{i=1}^{K} \omega_{U,i}\eta(X, \mu_{U,i}, \Sigma_{U,i}) \\ &= \sum_{i=1}^{K} \omega_{U,i} \frac{e^{-\frac{1}{2}(X-\mu_{U,i})^T \Sigma_{U,i}{}^{-1}(X-\mu_{U,i})}}{\sqrt{(2\pi)^3 |\Sigma_{U,i}|}} \end{aligned} \quad (9)$$

Computing (9) in a straightforward manner would be very costly. As in [6], we implement an on-line K-means approximation by checking every new foreground pixel value against the existing K Gaussian distributions. Based on the similar method as in [6], the μ_U, Σ_U and ω_U parameters for each distribution are updated at a learning rate 0.0025 which is less than α in Part B. Heuristically, the Gaussian distribution which has the most supporting evidence and the least variance is defined as the umbra model. More information about these methods can be found in [10].

D. Switch Detection between Daytime and Nighttime

As mentioned above, the system we propose in this paper exploits different processing methods at daytime and nighttime. The switch between daytime and nighttime is based on average luminance analysis and K-means clustering. We calculate the average luminance $al_{in,t}$ for every input image at time t and the average luminance al_j of every ten minutes in the past twenty-four hours to form an array of average luminance from al_1 to al_{144}.

A natural objective function can be obtained since we know there are two clusters in the problem domain. Each cluster is approximated as a normal distribution which has a mean μl_i and a standard deviation σl_i, and the jth element to be clustered is written as al_j. We know that elements are close to the center of their cluster based on the Mahalanobis distance, yielding the objective function

$$\Phi(clusters, data) = \sum_{i \in clusters} \left[\sum_{j \in ith\ cluster} \left(\frac{al_j - \mu l_i}{\sigma l_i} \right)^2 \right] \quad (10)$$

Notice that each cluster should be composed of at most two segments of connected average luminance in the sequence number axis (see Fig. 5); this can be achieved by analyzing the K-means results. The detail of the switch detection is given in Algorithm 1. If the clusters accord with mode 2 or mode 4 in Fig. 5, it is daytime now; else, it is nighttime now.

```
Algorithm 1: Switch detection between daytime and nighttime
Input - 144 elements: $al_1, al_2 \ldots al_144$ (Fig. 6(a))
Output - 2 clusters: daytime cluster and nighttime cluster
Begin
    Using $K$-means clustering to form the initial clusters which may have some noises and conflict with Fig. 5
    Noise reduction by $k$-nearest neighbor algorithm on initial clusters along the sequence number axis
    Searching for the longest segments of daytime and nighttime and computing two optimal broken points between daytime and nighttime to form the ultimate clusters (Fig. 6(b))
End
```

E. Vehicle Detection and Counting

It is not an advisable strategy to judge whether a ROI is occupied or not based only on the object pixel ratio in the ROI. This strategy has two main problems. First, the ubiety of objects is not taken into account, thus it is unable to discriminate among different vehicles occupying the same ROI. Second, the temporal continuity of objects is not taken

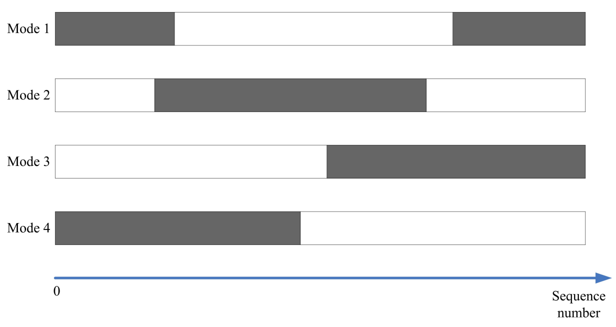
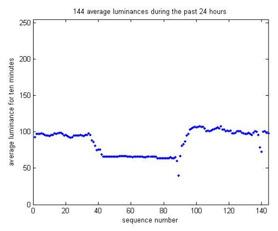
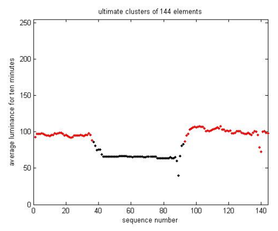


Fig. 5. Four modes of clusters: white segments represent daytime cluster, and gray segments represent nighttime cluster.



(a)



(b)

Fig. 6. Switch detection between daytime and nighttime: (a) 144 elements of average luminance during the past 24 hours; (b) ultimate clustering result based on the proposed algorithm.

into account either, thus the accuracy of vehicle counting will be heavily affected.

To overcome the disadvantages of simply thresholding, we attempt more reasonable methods. In a ROI, we utilize four characteristic lines whose end points lie equably on the four sides of the ROI (Fig. 7). Accordingly, we define five variables related to the occupancy condition: R_{ROI} denotes the occupancy rate of object pixels in the ROI; R_{a1}, R_{a2}, R_{b1}, and R_{b2} denote the occupancy rates of objects pixels on four characteristic lines. In addition, we define a five-level (0 to 4) confidence value CV to judge the vehicle existence.

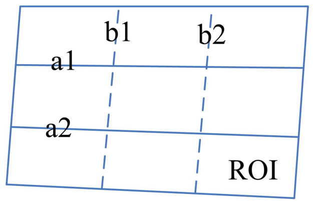


Fig. 7. Each ROI has four characteristic lines.

We compute CV based on these five occupancy rates. CV is added by one if the following constraints are satisfied,

$$\begin{aligned} & R_{ROI} > 0.3 \text{ and} \\ & (R_{a1} > 0.3 \text{ or } R_{a2} > 0.3) \text{ and} \\ & (R_{b1} > 0.3 \text{ or } R_{b2} > 0.3) \end{aligned} \qquad (11)$$

and CV is subtracted by one if the following constraints are satisfied.

$$\begin{aligned} & R_{ROI} < 0.1 \text{ or} \\ & (R_{a1} < 0.1 \text{ and } R_{a2} < 0.1) \text{ or} \\ & (R_{b1} < 0.1 \text{ and } R_{b2} < 0.1) \end{aligned} \qquad (12)$$

The maximum of CV is 4, which represents the ROI is occupied; the minimum of CV is 0, which represents the ROI is not occupied. If CV is at other confidence levels, whether or not the ROI is occupied depends on the latest 4 or 0. When the ROI is changed to occupied from unoccupied, the vehicle counting is added by one.

The proposed method performs well in applications. Lines a1, a2, b1, and b2 are jointly used to discriminate among different vehicles taking up the same ROI. Specifically, the method is able to count left-and-right vehicles or fore-and-aft vehicles. On the other hand , confidence-based vehicle detection takes the context information into account and makes the system quite effective to suppress various random noises, such as camera jitter, swaying trees, system circuit noises and transmission noises, etc.

IV. Experiments and Discussions

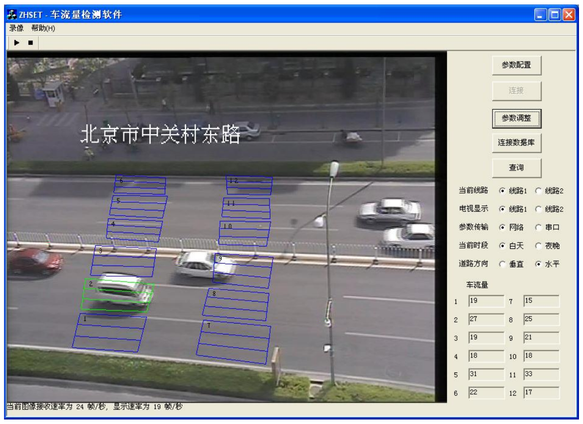
Long term tests on actual traffic scenes have been performed. The proposed system was used to count real-time traffic flow while surveying the traffic scene through the PC software. Vehicle counting at daytime and nighttime are demonstrated in Fig. 8 and the counting results for one day are shown in Fig. 9.

From Fig. 9 it can be seen that the proposed system is reliable to estimate the traffic flow for longtime span. There are roughly two valleys and two peaks in each flow chart for one day. The first valley shows the sparsest traffic flow during the whole day takes place between 4:00 am to 5:00 am. The two peaks locate respectively between 8:00 am to 9:00 am and between 5:00 pm to 6:00 pm when most people go to work and go home. The second valley is about between 11:00 am to 1:00 pm, as to indicate that most people are in the office or at home at noon. One of the major advantages of the proposed system is that it keeps robust and efficient even in busy traffic scenes thanks to the fast background rebuilding and effective vehicle detection, while the tracking based vehicle counting method may become invalid under such situations.

We also made quantitative evaluation on the counting accuracy. Let n_{real} denote the real vehicle number counted manually and n denote the vehicle number output by the system, we define the vehicle counting accuracy as

$$VAC = \left(1 - \frac{|n - n_{real}|}{n_{real}}\right) \times 100\% \qquad (13)$$

In our five twenty-minute statistics under different weather and road conditions at daytime, the VCAs are 98.7%, 99.4%, 87.6%, 95.3%, and 98.8%, respectively. Thus the average counting accuracy at daytime is estimated as 96.0%. Similarly, we estimate the average counting accuracy at nighttime as 92.2%. However, we believe that the estimation is conservative because the image compression and transmission



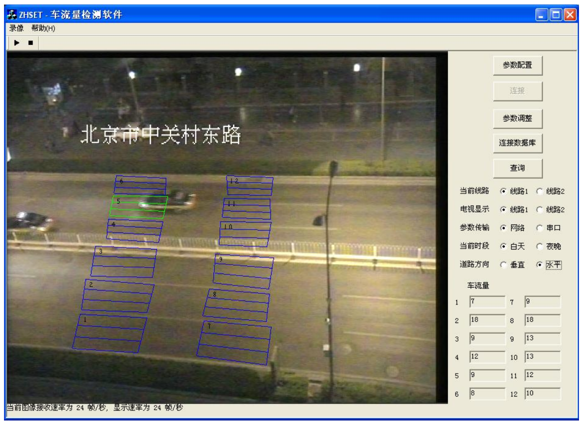
当前线路 [✓ 线路1] 线路2
电视显示 [✓ 线路1] 线路2
参数传输 [✓ 网络] 串口
当前时段 [✓ 白天] 夜晚
道路方向 垂直 [✓ 水平]

车流量

1	19	7	15
2	27	8	25
3	19	9	21
4	18	10	18
5	31	11	33
6	22	12	17

当前图像接收速率为 24 帧/秒，显示速率为 19 帧/秒

(a)



当前线路 [✓ 线路1] 线路2
电视显示 [✓ 线路1] 线路2
参数传输 [✓ 网络] 串口
当前时段 [✓ 白天] 夜晚
道路方向 垂直 [✓ 水平]

车流量

1	7	7	9
2	18	8	18
3	9	9	13
4	12	10	13
5	9	11	12
6	8	12	10

当前图像接收速率为 24 帧/秒，显示速率为 24 帧/秒

(b)

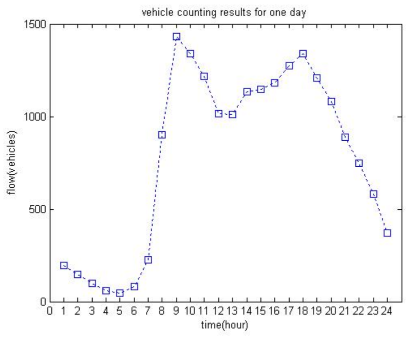
Fig. 8. Two snapshot of the host PC software showing the real-time vehicle detection and counting condition (a) at daytime and (b) at nighttime.

through network from the DSP system to the TMC slightly affects the real-time ability of the image processing algorithm, especially the confidence value *CV* in Section III(E), which is a very significant parameter in our system. In fact, when the system works, it is usually unnecessary to transmit real-time images.

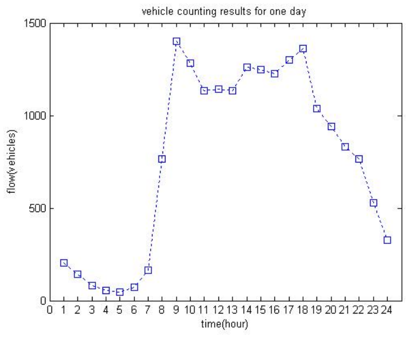
On the other hand, it must be pointed out that the proposed system fails to detect motorcycles which are too small compared with the ROI in image space to increase the confidence value *CV* through (11). Fortunately, motorcycles are quite sparse in most modern cities, such as Beijing, China, thus do not make serious influences on the generalized traffic flow.

V. Conclusion

In this paper we present a detailed description of using DSP board and image processing techniques to detect and count moving vehicles in traffic scenes. We develop the system based on embedded DSP structure to avoid heavy investment in industrial control computer while obtaining better computational power. The virtual loop-based method is used to detect and count moving vehicles. Long term tests on actual traffic scenes show that the proposed system is reliable to estimate real-time traffic flow rate. Future work



(a)



(b)

Fig. 9. Vehicle counting results for one day in the driving direction (a) from left to right and (b) from right to left of the traffic scene as in Fig. 8.

will be focused on system optimization, mainly algorithm implementation and video compression, to come up with a more powerful traffic surveillance system.

References

[1] D. Koller, J. Weber, and J. Malik, “Robust multiple car tracking with occlusion reasoning,” *in European Conference on Computer Vision*, 1994, pp. 189-196.

[2] R. Cucchiara, M. Piccardi, and P. Mello, “Image analysis and rule-based reasoning for a traffic monitoring system,” *IEEE Trans. Intell. Transport. Syst.*, vol. 1, no. 2, pp. 119-130, June 2002.

[3] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, “A real-time computer vision system for vehicle tracking and traffic surveillance,” *Transportation Research Part C*, vol. 6, pp. 271-288, 1998.

[4] H. Veeraraghavan, O. Masoud, and N.P. Papanikolopoulos, “Computer vision algorithms for intersection monitoring,” *IEEE Trans. Intell. Transport. Syst.*, vol. 4, no. 2, pp. 78-89, June 2003.

[5] P.G. Michalopoulos, “Vehicle detection video through image processing: The autoscope system,” *IEEE Trans. Vehicular Technology*, vol. 40, pp. 21-29, Feb. 1991.

[6] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” *in Proc. Computer Vision and Pattern Recognition*, June 1999, pp. 246-252.

[7] E. Salvador, A. Cavallaro, and T. Ebrahimi, “Cast shadow segmentation using invariant color features,” *Computer Vision and Image Understanding*, vol. 95, pp. 238-259, 2004.

[8] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, “Improving shadow suppression in moving object detection with HSV color information,” *in Proc. IEEE Int. Conf. Intell. Transport. Syst.*, Aug. 2001, pp. 334-339.

[9] T. Horprasert, D. Harwood, and L.S. Davis, “A statistical approach for real-time robust background subtraction and shadow detection,” *Proc. IEEE Int. Conf. Computer Vision ’99 FRAME-RATE Workshop*, 1999.

[10] K.F. Wang, Q.M. Yao, X. Qiao, S.M. Tang, and F.Y. Wang, “Moving object refining in traffic monitoring applications,” *IEEE Int. Conf. Intell. Transport. Syst.*, Sept. 2007, accepted.