A Video-based Traffic Violation Detection System

Xiaoling Wang\textsuperscript{1,2}, Li-Min Meng\textsuperscript{1}, Biaobiao Zhang\textsuperscript{2}, K.-L. Du\textsuperscript{2}

1. College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China
2. Enjoyor Labs, Enjoyor Inc., Hangzhou 310030, China
648774168@qq.com

Abstract—Traffic violation detection systems are effective tools to help traffic administration to monitor the traffic condition. It can detect traffic violations, such as running red lights, speeding, and vehicle retrogress in real time. In this paper, we propose an improved background-updating algorithm by using wavelet transform on dynamic background, and then track moving vehicles by feature-based tracking method. A complete traffic violation detection system is realized in C++ with OpenCV.

Keywords—traffic violation detection, vehicle tracking, red running.

I. INTRODUCTION

With the growth of the number of vehicles, the number of traffic accidents is rapidly rising. Therefore, it is important to capture traffic violations to ensure traffic safety and reduce traffic accidents. Traffic violations are the most important cause of the accidents. The traffic violation detection system can effectively monitor the traffic, and capture violations such that law enforcement can be applied. This helps to reduce illegal driving so as to ensure a smooth traffic flow.

II. MOVING DETECTION

It is very difficult to identify moving vehicles, then to track and classify them in real time within a complex environment. There are a variety of approaches to vehicle detection in video streams, including background difference, inter-frame difference, inter-frame corresponding, and edge detection methods [1-4].

A. Background Difference Method

The background difference method is commonly used in video processing. A background image without vehicles is firstly obtained. This image is then subtracted from the current input image, and the difference image is obtained. One can determine whether vehicles exist in input image by binarization of the difference image. This method is computationally fast. However, it needs to update the background image in real time when the environment changes.

B. Inter-frame Difference Method

The inter-frame difference method subtracts the current frame from the previous frame, and then we can find the changing area with setting the threshold value. When the target is in motion, there will be residual image in moving direction which the motive vehicles can be detected by.

C. Edge Detection Method

The edge detection method applies edge detection on the input image, and then applies image denoising on the resulting image. The denoised image is matched with a template image. If they match each other, there is a vehicle. This method has few impacts on the environment. It outperforms the background difference method. However it has high computational complexity for vehicle model matching. Thus, this method is not suitable for real-time processing.

D. Optical Flow Method

The optical flow method [6] is an effective way to detect moving targets. It detects moving objects by the change in the time domain of pixel intensity of an image sequence, and the relationship between the structure of objects and movement. This method is, however, computationally intensive and is susceptible to noise.

E. Block Matching Method

The block matching algorithm (BMA) [7] is a video detection method based on motion vectors. It splits an image into $M \times N$ macro blocks. We can get the motion vectors by searching for optimal matching of a macro block of the current frame in the next frame. A moving vehicle is composed of many macro blocks which perform the same movement. This method is often used in motion estimation.

III. DYNAMICAL TRACKING

This paper discusses a dynamical background update method based on the wavelet transform, which is combined with background difference and feature-based tracking to detect moving vehicles. There are four steps as follows.

- Perform background difference with the filtered image, so as to detect moving objects quickly and effectively. However it is always sensitive to the changing background, so it is necessary to update the background in time. In this paper, a dynamical background update algorithm is proposed. The algorithm utilizes a weighting factor to adjust the speed of background update

$$I_d(x, y) = \alpha I_s(x, y) + (1 - \alpha) I_d(x, y), \quad (1)$$

where $I_s(x, y)$ is a pixel of the source image, $I_d(x, y)$ is a the pixel of the background image, $\alpha$ is
a weighting parameter that is used to adjust the speed of background update within the current image, and $\text{mask}(x, y)$ is a mask template that extracts from the foreground image. This method has the merit of avoiding the interference of the foreground, when updating the background.

$\alpha$ has a significant effect on the accuracy of vehicle detection. If the background updates too fast, there are extra holes in the foreground image; otherwise, the movement affects detection performance. Taking advantage of wavelet transform can solve this problem. 2-D discrete wavelet transform splits a source image into four components: the average, horizontal, vertical, and diagonal components. Filter the horizontal, vertical, and diagonal subimages after performing the wavelet transform. Then reconstruct the background with the average component. This eliminates trajectory.

- Select a detection area in the foreground image, and perform edge detection. The contours of the vehicles are then obtained. When the contour area reaches a threshold, the contour will be identified by the original tracking rectangle. Define the area ratio by

$$r = \frac{A_{\text{rect}}}{A_{\text{roi}}}.$$  \hspace{1cm} (2)

where $A_{\text{rect}}$ is the area of the rectangle, and $A_{\text{roi}}$ is the region of interest. When $r > 1/2$, the rectangle captures the tracked object.

- Calculate the color histogram of the tracking target. The histogram of $H$ component in HSV space is obtained, and the probability distribution of every pixel in the target is denoted $w_i$.

- Calculate the center of gravity of the moving target

$$M_{00} = \sum_{i=1}^{n_1} w_i, \quad M_{10} = \sum_{i=1}^{n_1} w_i x, \quad M_{01} = \sum_{i=1}^{n_1} w_i y, \quad M(x, y) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right),$$  \hspace{1cm} (3)

where $M_{00}$ is the zeroth-order moment of $w_i$, $M_{10}$ and $M_{01}$ are the first-order moments of $w_i$, and $M(x, y)$ is the center of gravity. Repeatedly adjust the tracking center to $M(x, y)$, until the distance between the tracking center and $M(x, y)$ is less than a threshold value $T$. Then the new tracking window can be obtained as the search box of the next frame. This process is shown in Fig. 1.

![System function diagram](image1.png)

**Fig. 1** The flowchart of dynamic tracking

![Simplified flowchart](image2.png)

**Fig. 2** System function diagram

![Simplified flowchart](image3.png)

**Fig. 3** A simplified flowchart of the system

**IV. SYSTEM DESIGN**

The system consists of three modules, namely video loading, detection of violating vehicles, and violation evidence storage. The detection of violating vehicles contains image preprocessing, violation detection, and background update. The system function diagram is shown in Fig. 2.

The system was built in C++ by Visual Studio 2010 and OpenCV 2.3.1.

The function `DrawToHDC()` in OpenCV realizes video loading. We create the CvImage class.

OpenCV library already contains a variety of image processing functions, such as Gaussian filtering function `GaussianBlur()`, image space mapping function `cvtColor()`, background subtraction function `absdiff()`, threshold function `threshold()`, Canny edge detection function `canny()`, histogram calculation function `calcHist()`.
calcHist(), and many other commonly used functions. Fig. 3 is a simplified flowchart of the system.

- **OnTimer()**: Load video.
- **Processing()**: Perform image processing, filtering, and background difference, and output foreground image.
- **LineDetect()**: Detect stop lines and lanes.
- **LightDetect()**: Detect traffic lights.
- **Tracking()**: Perform edge detection, area setting and detection of moving vehicles, and output the original tracking window.
- **VehicleInfo()**: Show the violation information.
- **TrackDetect()**: Track moving vehicles, and get the centers of gravity.
- **DrawTrack()**: Draw tracks.
- **ShowDetectedImg()**: Store and show the detected violation image.
- **PlayVideo()**: Play the video.

The procedure of violation detection is given in Fig. 4. As shown in Fig. 4, the system includes image processing, vehicle tracking, violation identification and information storage. The most important step in image processing is background update. We propose a real-time adaptive background update algorithm to improve vehicle detection. We track the vehicles by the center, and decide whether they violate or not by thresholding. For example, for the center \((x_M, y_M)\), when the coordinate \(y_M\) is not in the range of threshold \(y_T\) (when \(y_M > y_T\)), and the traffic light is red, the vehicle runs the red light. Finally, this system automatically saves and displays the violating images.

Fig. 4 Violation detection flowchart
V. RESULTS

The rate of background update affects the accuracy of motion estimation. The video has a rate of 25 frames per second. The results according for different rates of background update (0.01, 0.03, 0.05 and 0.1) is shown in Fig. 5. It is seen from Fig. 5 that the method performs the best when $\alpha$ is 0.03.

![Fig. 5 Extracted foregrounds for different rates](image)

(c) $\alpha = 0.05$  (d) $\alpha = 0.1$

Fig. 5 Extracted foregrounds for different rates

VI. CONCLUSION

We have developed a system that can detect violations, track vehicles effectively, and automatically save and display the information. It realizes intelligent traffic management. We combine the wavelet transform and dynamical background update to eliminate the vehicle trajectory. The combination of the background difference method and feature-based tracking method enables a more fast and accurate detection and tracking.

REFERENCES