Vision based Vehicle Tracking using a high angle camera

Raúl Ignacio Ramos García
gramos@clemson.edu

Dule Shu
dshu@clemson.edu

Abstract

A vehicle tracking and grouping algorithm is presented in this work using Lukas-Kanade Tracking technique. Every frame of the sequence is downsampled to improve efficiency. A section of the highway is marked to establish a detection zone where the detection and tracking of the features take place. For every frame a background subtraction is done and only the foreground pixels inside the detection zone are kept; features will be specified only for these pixels, grouped and tracked until the vehicle leaves the detection zone, we also store the amount of vehicles that had entered the zone. Feature points computed in the downsampled frames are later displayed in the original size images of the sequence.

1 Introduction

Vehicle tracking has become an increasingly demand for a reliable, high-efficiency modern transportation system. Among all the vehicle tracking technologies, vision-based tracking has become an attractive option for its easy installation and operation, relatively low cost, as well as the powerful functions (e.g. the estimation of the speed and size of a vehicle, statistics on type and number of vehicles, etc.) As a result, more and more research has been conducted in this field.

In this paper we have suggested a method of tracking vehicles in a highway as well as counting the number of them. A Lucas-Kanade [1, 2] based algorithm is introduced to track each vehicle using feature points in a detection zone in a video. Each individual vehicle is grouped and tracked with feature points of different color.

The paper is organized as follows: Section 2 introduces some research work done previously as the reference for this research. Section 3 describes the method used for vehicle tracking. The experimental results of implementing the method using program is provided in Section 4. Conclusion is given in Section 5.

2 Previous Work

Extensive research regarding vision based vehicle tracking has been done over the past years. Several techniques have been developed as a result of these studies. One is *Blob Tracking* [2]. In this approach a background model without moving objects is generated for the scene. During the sequence each frame would be compared with the background model by doing the absolute difference between them and consequently obtain a foreground blob representing the vehicles. Another method is the *Active Contour Tracking* [2,4], this method tracks the contour of the foreground blob. The *3-D Model Based Tracking* [2,5] constructs a three-dimensional model using an aerial view of the scene to eliminate all occlusions. *Feature-Tracking* [1,3] is another method that uses feature points to track the objects; this method had brought good result where partial occlusion exists. This method also requires a groping algorithm to track multiple features. Other methods can be found in [2].

3 Method

In this work we accomplished a vehicle tracking in a highway with a camera located at a high point of the ground. A mixture of blob and feature tracking methods were used in this research. The procedure is explained as follows:

1) Background Modeling: To model the background a single frame of scene (figure 1) was taken containing no objects of interest [6]. Then this image is smooth to avoid noise caused by the image lighting effects. The

background and images in the video sequence are also downsample. A background subtraction is done by computing the absolute difference of graylevel values between the background image and the current image [3]. In this way non zero pixel are the foreground pixels belonging to objects (vehicles) on the highway, as shown in figure 2.

Figure 1. Background

Figure 2. Background subtraction

The next step is to obtain a better shape of the blob by doing dilations and erosions to figure 2. To avoid unimportant objects that appear after the subtraction, a section of the image is specified to be the zone where the vehicles ought to be tracked. In this zone (i.e. detection zone) foreground pixels are kept and the rest are discarded as shown in figure 3 and 4.

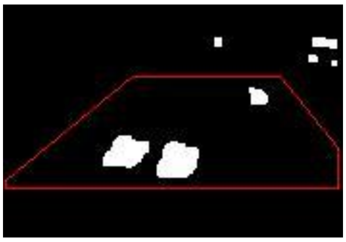


Figure 3. Better blob shape and detection zone.

Figure 4. Pixels inside the detection zone.

2) Feature Detection: For every frame features are detected only for the foreground pixels inside the detection zone using Lucas-Kanade-Tomasi feature detection algorithm [1]. The gradient in the x and y directions of the image (*Ix* and *Iy* respectively) are computed to determine good features using the fallowing expressions:

$$Ix = I * Gy * \frac{\partial}{\partial x}(Gx) \qquad Iy = I * Gx * \frac{\partial}{\partial y}(Gy)$$

Where *I* refers to the image, *Gx* and *Gy* are a Gaussian kernels convolve in the x and y directions of the image *I*, *∂/∂x(Gx)* and *∂/∂y(Gy)* is the derivative of the Gaussian kernel convolve with the image in x and y directions respectively. The next equation shows how to obtain the derivative of Gaussian in the x direction, where σ is the variance:

$$\begin{aligned} \frac{dG(x)}{dx} &= \frac{d}{dx}\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-x^2}{2\sigma^2}\right)\right] \\ &= -\frac{x}{\sigma^2\sqrt{2\pi\sigma^2}}\exp\left(\frac{-x^2}{2\sigma^2}\right) \\ &= -\frac{x}{\sigma^2}G(x) \end{aligned}$$

Good feature points are determined to compute the eigenvalues of the gradient covariance matrix *Z* of every pixel. The elements of the matrix are obtained from the gradient images:

$$Z = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Eigenvalues (λ) of matrix *Z* can be calculated by solving the traditional eigenvalue and eigenvector problem of the form:

$$Ze_i = \lambda_i e_i \qquad (i=1,2)$$

where e_i is the eigenvectors and λ_i is the eigenvalue. So λ_1 and λ_2 have to be large enough than a threshold to be consider as a good feature, since large eigenvalues means that more information can be obtain from the data of matrix *Z* and thus the feature can track. In this way only the minimum of the resulting eigenvalues is kept for the thresholding:

$$\min(\lambda_1, \lambda_2) > \lambda_{Threshold}$$

Figure 5 shows the features obtained from a given image of the sequence.

Figure 5. Feature points in downsampled image.

3) <u>Grouping</u>: In this step we present our own approach for feature grouping. As new vehicles enter the detection zone, labels are assigned to each blob in the background subtraction image (figure 6). These labels are assigned to each feature point as well. These numbers (value of labels) represent the group of the feature point. Through the trajectory of a vehicle feature points area not only tracked but also the value of the previous assigned label are tracked as well. In this way the same label value is passed from frame to frame until the vehicle leaves the detection zone.

Figure 6. Grouping vehicles.

Label values are allowed to be an integer from 1 to 5 in the detection zone, so when a new vehicle enters the detection zone one of these label values is assigned to it and at the same time the number of vehicles that had entered is updated.

4) <u>Feature Tracking</u>: The detected feature points are tracked using Lucas-Kanade-Tomasi feature tracking [1] between two consecutive frames. A window of size N is used to gather more information of the texture around the feature point because the value of a single pixel can change due to noise. In this way the gradient matrix Z is compute as:

$$Z = \sum_{i=1}^{N} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

The displacement **d** of a feature is computed to minimize the residue error (ε):

$$\mathbf{d} = [dx\ dy]^T$$

where dx and dy are the displacements in x and y directions. The displacement **d** is calculated by iteratively solving the following equation for Δ**d** (where Δ**d** = d):

$$Z\mathbf{d} = \mathbf{e}$$

where

$$\mathbf{e} = \sum_{i=1}^{N} (I_i - J_i)[I_{xi} \quad I_{yi}]^T$$

I and J are the two consecutive images. This is done to minimize the nonlinear error. Each time a new Δ**d** is calculated the value for **d** is updated by:

$$\mathbf{d}(i+1) = \mathbf{d}(i) + \Delta\mathbf{d}$$

until the condition norm(Δ**d**) < ε is satisfied. Then the coordinate of a feature point is updated by adding **d** to the formal coordinate **x**(t):

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{d}$$

where **x**(t+1) is the updated coordinate of the feature point for the next frame.

5) <u>Final Display</u>: For every feature tracked and grouped, the label given in 3 to the features is matched with a different color to distinguish different vehicles. So far figure 7 depicts the colored feature points in the downsampled image. The last step is to project these feature points onto the original size image and to display the number of the vehicles as shown in figure 8.

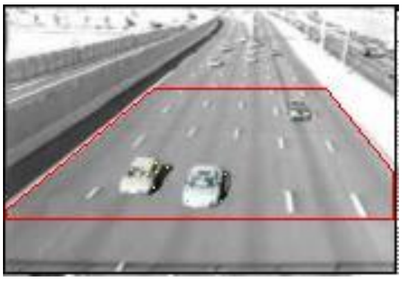


Figure 7. Feature points downsample image.

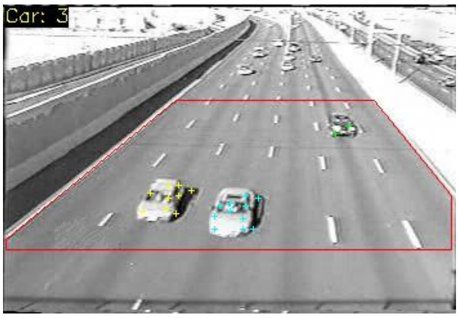


Figure 8. Feature points upsample image.

4 Experimental Results

The proposed method was tested on a video sequence of 105 frames. The video was captured by a high angle surveillance camera over a highway. No preprocessing was done to suppress shadows. In this program only 5 label values for grouping was allowed, as described in section 3 each value matches a corresponding feature color as follows: 1 – Green, 2 – Yellow, 3 – Cyan, 4 – Blue, and 5 – Magenta. It has been proved through the experiment that each individual vehicle had been successfully detected by the program using the method suggested in method section 3. The number of vehicles which have passed through the detection zone was also correctly counted. Table 1 gives a summary of the results:

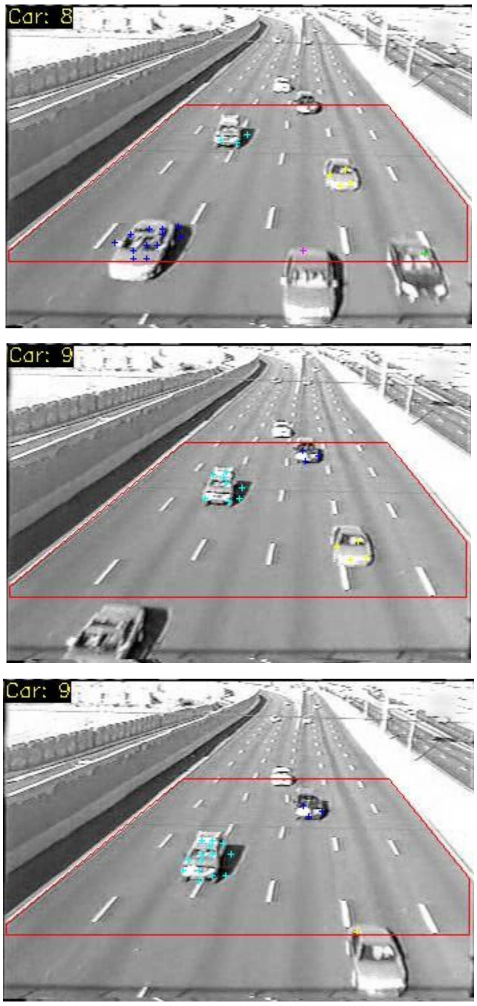
Table 1. Results

RESULTS	Total
Frames	105
Video (sec.)	12
Vehicles passing the detection zone	12
% accuracy of tracked vehicles	100
% accuracy of counted vehicles	100

Several frames of the sequence are shown here. In Figure 9, (Frame 34) detection zone is specified by red boundary. All five vehicles in the detection zone have been detected by feature points. Each vehicle is grouped by feature points of a different color. At the top left corner of the frame, a window is displayed to indicate the overall number of vehicles passing through the detection zone.

Figure 9. five vehicles are detected in a frame.

In Figure 10, four frames (Frame 37, 40, 44, and 49) are shown to demonstrate the process of vehicle tracking in the detection zone. Vehicles are detected by feature points, these features points are tracked throughout the detection zone, and removed as soon as the vehicles leave. Also, it can be seen in the frames that the number in the window at the left top corner increases as new vehicles are detected.



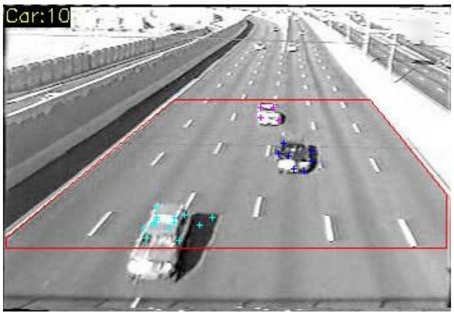


Figure 10. Process of vehicle tracking.

It should be noted that some factors in the program play a vital role in the final result of the vehicle tracking. One factor is the erosion and dilation. The sequence of dilation and erosion must be properly chosen to get a good enough background subtraction image. Too much dilation will result in the 'merging' of two foregrounds and thus make the program fail to distinguish the two vehicles in a frame. Too much erosion will lead to too small foreground so that some vehicles will disappear in the detection zone and some others will not have enough feature points to track. Another factor is to provide enough feature points for every vehicle for the program to be able to track and group accurately. In this work the suggested distance is 2 pixels in order to guarantee that no vehicles will be lost during the tracking. A third factor is the position and shape of the detection zone. For this particular image sequence, the upper bound of the detection zone must be low enough so that vehicles will not be too small or merge into each other when they are entering the zone. The lower bound of the zone also needs to be carefully chosen because it sometimes affects the counting of the vehicles. Generally, a smaller detection zone can save computation time and hence increase the speed of the program.

5 Conclusion

In this paper we proposed a method of vehicle tracking based on Lucas-Kanade tracking algorithm. There are mainly five steps involved in this method: background modeling, feature detection, grouping, Lucas-Kanade feature tracking, and display of the upsampled image with the vehicle counting. A program was written to implement the method, and a video sequence captured by a high angle surveillance camera was used to test the method. The program worked well with the given video sequence.

Given the successful vehicle detection, tracking and counting, there are still efforts that can be made to improve the experimental results. Future work is to improve the real time applicability of the program. So far the method hasn't been able to be applied to real time vehicle tracking because the program cannot run fast enough to follow the motion of the vehicles on the highway. More work also includes methods to solve the occlusion and shade problems of the vehicle as well to model the background using statistical techniques.

References

[1] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical. Report CMU-CS-91-132, Carnegie Mellon University, Apr 1991.

[2] N. K. Kanhere, S. T. Birchfield, and W. A. Sarasua. Vehicle Segmentation and Tracking in the Presence of Occlusions, TRB Annual Meeting Compendium of Papers, Transportation Research Board Annual Meeting, Washington, D.C., January 2006.

[3] S. C. Cheung and Chandrika Kamath. Robust techniques for background subtraction in urban traffic video. In Proceedings of Electronic Imaging: Visual Communications and Image Processing, 2004.

[4] N. K. Kanhere, S. J. Pundlik, and S. T. Birchfield, Vehicle Segmentation and Tracking from a Low-Angle Off-Axis Camera, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, California, June 2005.

[5] S. J. Pundlik and S. T. Birchfield, Motion Segmentation at Any Speed, Proceedings of the British Machine Vision Conference (BMVC), Edinburgh, Scotland, September 2006.

[6] Magee D. Tracking Multiple Vehicles using Foreground, Background and Motion Models. *Image and Vision Computing* , vol 22(2), pp143-155, 2004.