A Simple Approach for Real Time Speed Estimation of On Road Vehicles Using Video Sequences

A. Mosayebi¹, H. Khosravi²,*

¹,² Electrical Engineering Department, Shahrood University, Shahrood, Iran

ARTICLE INFO

Article History:
Received 15 March 2021
Received in revised form
18 April 2021
Accepted 20 June 2021
Available online 24 July 2021

Keywords:
Speed estimation,
videos, ROIs,
lane borders,
centroids

ABSTRACT

This paper provides an efficient and simple approach towards real-time speed estimation of on road vehicles for surveillance applications. In the presented method, videos are supposed to be captured with a stationary camera mounted on a two-lane road and there is no need for the camera to be calibrated. The algorithm has two main phases, in the first phase there is an interactive procedure in which lane borders and real world distances are defined just once at the beginning. Then, based on the already received information, two rectangular ROIs are defined for each lane. In the second phase, approximate binary mask of the foreground is created differencing the two consecutive frames. Eventually, calculating centroids and the norm values of the binary mask in the ROIs, algorithm can compute the time that it takes each vehicle to pass between the two aforementioned lines and thus, average speed can be computed. In short, although the algorithm of this paper is simple, it is real-time and efficient, and its implementation doesn’t require any specific hardware. The average error of speed estimation is ±3 km/h and the detection accuracy is 83 %.}

1. INTRODUCTION

Video based approaches have recently become popular in many intelligent transport systems, such as vehicle detection, vehicle make and model recognition, and speed estimation. They are becoming substitutions for the traditional sensors since they appear to be more efficient and flexible than the sensors. Vision based methods are cheaper to use too. Vehicle speed estimation is an important part of a transport surveillance system, providing it with many capabilities as traffic control, preventing accidents and so forth.

Many different approaches have been proposed already, some of which used the side view videos of the vehicles while others used the front view ones. There are methods that consider occlusion and other challenging conditions like lightning changes or shadows, while others ignore such circumstances. IN the following, a few of these works are described briefly. In [1], a pixel-wise weighting list is used for dynamic foreground extraction. A shadow removal step is also applied using colors and edges. Forming a spatial-temporal profile, traffic flow and speed are both

* Corresponding Author: mosyebiahmad@yahoo.com
Electrical Engineering Department, Shahrood University

http://dx.doi.org/10.47176/TMI.2021.29
© 2022 by the authors. Licensee TMI, Tehran, Iran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).
estimated. However, the method may fail when there are dark shadows or when there’s occlusion due to improper placement of camera. In another approach, an automatic camera calibration is performed by finding three vanishing points, using vehicle movement and vehicle edges. A 3D bonding box is constructed for each vehicle and then the speed is measured [2]. In [3], Kalman filter is used to track vehicles and estimate the speed, and principal component analysis is applied for classification of the tracked vehicles. There is also an algorithm in which camera calibration is used to avoid the perspective distortion, then centroids are tracked, and through this, speed can be estimated. In this approach occlusion is not considered as it can rarely happen in the used sequences [4]. In [5], headlights of the vehicles are paired and tracked to estimate the speed in low light conditions. Pinhole model and Euclidean distance are used in the speed estimation process.

The proposed system in this paper is designed to estimate average speed of moving vehicles, using image sequences which are captured by a stationary camera mounted on a two lane road. Images are oblique and they are captured by a camera with a wide-angle lens. The presented approach simply aims to estimate the speed using a fast and efficient algorithm.

There are a few steps in order to measure the speed as shown in the flowchart of Figure 1. The captured video is the input of the estimation system, in phase 1, through an interactive procedure, some necessary parameters are determined by the user as a one-time job at the beginning of the process. After that, the speed estimation process starts by first converting each incoming RGB frame to grayscale, then the binary mask of the moving vehicles can be created differencing every two consecutive frames and thresholding. Then centroids of the binary foreground in the already determined ROIs are obtained. In the end, computing the passing time between the two virtual lines and having the real world distance between them, the average speed can be measured.

The rest of the paper is organized as follows. Section 2 explains the interactive procedure in which the user is supposed to define lane borders and real world distance, in this section it is described how the rectangular ROIs are created based on what user has already defined. Section 3 describes how the foreground mask is made. In section 4, it is described how centroids are calculated, in section 5, speed estimation scheme is explained, section 6 discusses the experimental results, and section 7 presents the conclusions.

![Flow chart of the speed measurement steps](image-url)

**Fig. 1.** Flow chart of the speed measurement steps
2. RECEIVING USER’S PARAMETERS AND CREATING ROIS

In this section, it is explained that what parameters are received, and how they are used for subsequent steps of the method.

2.1. Defining Lane Borders

In this step of the method, user is asked to define the lane borders. User has to specify 6 points in the road scene by clicking on the mouse, red points shown in the Figure 2 are the mentioned points and the lane borders are drawn by green lines. In order to be able to calculate the speed, the algorithm needs the user to define the real world distance between two horizontal lines in the scene, one at y= 0.4H and the other at y=0.75H and H is the height of the input frames. These percentages are considered since they seem to be appropriate for videos captured with an angle and field of view similar to our videos. The two virtual lines are drawn in yellow in Figure 2.

![Figure 2. User-defined parameters](image)

Those 6 points are used to form two filled polygons as binary masks, one for the left lane and another for the right lane.

2.2. Creating ROIs

The application is supposed to be real-time, therefore, operations must be done on smaller areas of the image instead of the entire image, so two rectangular ROIs are defined for each lane, and they are defined as follows. For each lane, the two lower vertices of the upper rectangle are the intersection points of the polygon sides and the line y=0.75H, and for the lower rectangle the two lower vertices are the intersection points of the polygon sides and the line y=0.85H. Widths of the upper and lower rectangles are 0.35H and 0.1H respectively. In Figure 3 for example, right lane binary mask is shown and intersection points are depicted as red and green circles.
2.3. Defining Real World Distance

To estimate the speed in the next phase of the algorithm, the real world distance has to be known, so in this step, user is asked to define such distance between the lines $y=0.4H$ and $y=0.75H$.

3. PRE-PROCESSING

3.1. RGB To Grayscale Conversion

After capturing each RGB frame, it is converted to grayscale using Equation (1).

$$Grayscale = 0.299 \, R + 0.587 \, G + 0.114 \, B$$ (1)

3.2 Frame Differencing

There are some common ways to extract foreground mask, like mixture of Gaussians and averaging frames, but for this work, frame differencing is chosen due to its being efficient and less time-consuming. Every input frame, after being converted to grayscale, is subtracted from the previous grayscale frame, then, thresholding absolute of the resulting image, foreground mask is created.

4. CALCULATING CENTROIDS

To calculate the centroid of the binary image in each ROI, x and y coordinates have to be calculated. Primarily, first-order moments are calculated for each coordinate separately as in the Equations (2) and (3).

$$m_{10} = \sum_x \sum_y x \, image(x,y)$$ (2)  
$$m_{01} = \sum_x \sum_y y \, image(x,y)$$ (3)

Then x and y coordinates are calculated as in the Equations (4) and (5), and the Equation (6) shows the resulting centroid. The red circle in Figure 4 shows the position of the calculated centroid within the drawn rectangular ROI.

$$x = m_{10}/(\sum_x \sum_y image(x,y))$$ (4)  
$$y = m_{01}/(\sum_x \sum_y image(x,y))$$ (5)  
$$Centroid = (x, y)$$ (6)
A very simple idea has been chosen to accomplish the main goal of this work. For each lane, as the centroid in the upper ROI passes a certain border and the norm of the binary mask within the ROI exceeds a predefined threshold, it is considered that the vehicle has entered the ROI, and the algorithm starts to count the frames until the vehicle leaves the ROI. Counting the frames stops as the centroid in the lower ROI passes a certain border and the norm of the binary mask, within the lower ROI, exceeds a predefined threshold. Once the counting is finished and the number of frames is known, the speed of the vehicle can be determined using the Equation (7), in which \( n \) is the number of frames and \( d \) is the real-world distance which is already defined by the user.

\[
\text{Speed}_{km/h} = \frac{d \times \text{frame rate} \times 3.6}{n}
\]  

A pseudocode of the algorithms is shown in the Figure 5.

Fig. 4. Centroid of the binary foreground in the ROI

5. SPEED ESTIMATION

A very simple idea has been chosen to accomplish the main goal of this work. For each lane, as the centroid in the upper ROI passes a certain border and the norm of the binary mask within the ROI exceeds a predefined threshold, it is considered that the vehicle has entered the ROI, and the algorithm starts to count the frames until the vehicle leaves the ROI. Counting the frames stops as the centroid in the lower ROI passes a certain border and the norm of the binary mask, within the lower ROI, exceeds a predefined threshold. Once the counting is finished and the number of frames is known, the speed of the vehicle can be determined using the Equation (7), in which \( n \) is the number of frames and \( d \) is the real-world distance which is already defined by the user.

\[
\text{Speed}_{km/h} = \frac{d \times \text{frame rate} \times 3.6}{n}
\]  

A pseudocode of the algorithms is shown in the Figure 5.

Fig. 5. Pseudocode of the speed estimation procedure

6. EXPERIMENTAL RESULTS

For testing the proposed algorithm, an RGB sequence is used, which is captured by a camera mounted on a two-lane road. As mentioned in the section 5, speed of a vehicle is estimated, when the passing time between the two virtual lines is known. Here, in the Figure 6, for example, a vehicle is shown in several video frames while it’s
passing the ROI in the left lane. The accuracy of the algorithm is tested by comparing the measured velocity and the real velocity of the vehicles. In table 1, as an example, estimated speeds of 6 passing vehicles are brought. Their real speeds are also presented for comparison.

As it can be seen in the Table 1, the results of the speed estimation are acceptable. The speed estimation error is because of sudden changes in the moving direction of the vehicle. The detection error occurs when there’s too much occlusion between vehicles, mostly when a sedan is moving behind a truck or bus. The algorithm is executed on a laptop with a 2.3 GHz Intel core, 8 GB ram memory and 1 TB hard disk. Processing time of the algorithm is 16 milliseconds per frame that is real-time.

![Fig. 6. Passing car in several frames](image)

**Table 1.** Estimated and real speeds of 6 vehicles

<table>
<thead>
<tr>
<th>Vehicle Number</th>
<th>Estimated Speed</th>
<th>Real Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td># 1</td>
<td>78.55 km/h</td>
<td>78.54 km/h</td>
</tr>
<tr>
<td># 2</td>
<td>86.40 km/h</td>
<td>86.40 km/h</td>
</tr>
<tr>
<td># 3</td>
<td>96.00 km/h</td>
<td>96.00 km/h</td>
</tr>
<tr>
<td># 4</td>
<td>54.00 km/h</td>
<td>61.71 km/h</td>
</tr>
<tr>
<td># 5</td>
<td>54.00 km/h</td>
<td>61.71 km/h</td>
</tr>
<tr>
<td># 6</td>
<td>Not detected</td>
<td>96.00 km/h</td>
</tr>
</tbody>
</table>

7. CONCLUSIONS

A system for speed estimation of on road vehicles is presented in this paper, in which, several steps are applied. In the proposed scheme, occlusion is not considered since it does not influence the results too much and there are also some cases of misdetection. The method has two important properties, the first one is that the implementation of the algorithm, as already mentioned, needs no special hardware and the second one is that it is real-time. These properties are the results of the efficient and simple algorithm used in this paper.

8. ACKNOWLEDGEMENT

The author would like to express his sincere thanks to Dr. Hossein Khosravi, Director of the Department of Electronics of the Shahrood University.

REFERENCES


