Transactions on Machine Intelligence 4(1) (2021) 29–35

ISSN Online: 2821-1693

Transactions on Machine Intelligence

Journal Homepage: https://tmachineintelligence.ir/index.php/TMI

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

A. Mosayebi[1], H. Khosravi[2,*]

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

ARTICLE INFO

Article History:
Received 3 January 2021
Received in revised form
10 February 2021
Accepted 18 March 2021
Available online 19 March 2021

Keywords:
Speed Estimation,
Videos, Rois,
Lane Borders,
Centroids

ABSTRACT

This study presents a practical and efficient method for real-time speed estimation of vehicles on two-lane roads, particularly suited for surveillance applications. The proposed approach utilizes video input captured from a fixed, stationary camera without requiring prior camera calibration—making it both cost-effective and easy to deploy. The algorithm operates in two primary phases. In the initial phase, an interactive setup allows the user to manually define lane boundaries and corresponding real-world distances once at the beginning of the analysis. Based on this input, two rectangular regions of interest (ROIs) are assigned to each lane. In the second phase, moving vehicles are detected by generating an approximate binary foreground mask through frame differencing of consecutive video frames. By calculating the centroids of moving objects and measuring the norm values of the binary mask within each ROI, the time taken by each vehicle to traverse the space between the predefined lines can be determined. From this time and known distance, the average speed of each vehicle is estimated. Despite its algorithmic simplicity, the method achieves real-time performance without requiring specialized hardware. Experimental results demonstrate a mean speed estimation error of ±3 km/h and a vehicle detection accuracy of approximately 83%.

1. INTRODUCTION

Video-based methods have become increasingly prominent in intelligent transportation systems (ITS), playing a vital role in applications such as vehicle detection, classification by make and model, and speed estimation. Compared to traditional sensor-based approaches, vision-based techniques offer notable advantages in terms of flexibility, cost-efficiency, and ease of deployment. Their ability to adapt to a wide range of environmental and operational conditions has further accelerated their adoption. Among these applications, vehicle speed estimation is particularly critical, as it supports key traffic surveillance functions, including congestion analysis, traffic flow monitoring, and accident prevention.

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

http://dx.doi.org/10.47176/TMI.2021.29

A variety of techniques have been explored in the literature. Some leverage side-view imagery, while others employ front-facing camera perspectives [1–2]. While certain methods are designed to handle challenging conditions such as occlusion, illumination changes, and shadows, others operate under simplified assumptions [3]. For example, the method proposed in [4] utilizes a pixel-wise weighting strategy to extract dynamic foregrounds, complemented by a shadow elimination step based on color and edge cues [5]. Although this method enables the construction of spatiotemporal profiles for estimating both vehicle speed and traffic flow, its effectiveness diminishes in scenarios involving deep shadows or significant occlusions caused by suboptimal camera positioning.

Another notable technique, described in [6], performs automatic camera calibration by identifying three vanishing points derived from vehicle motion and edge detection. A 3D bounding box is then generated for each vehicle to facilitate accurate speed estimation [7]. In another approach, vehicle tracking is performed using a Kalman filter, while classification is handled via principal component analysis (PCA) [8]. A further method applies camera calibration to correct for perspective distortion, subsequently tracking centroids to estimate speed. However, this method assumes minimal occlusion, as the tested datasets contain few such instances.

Additionally, a technique tailored for low-light conditions has been introduced, in which vehicle headlights are detected, paired, and tracked. Speed estimation is then achieved using the pinhole camera model in conjunction with Euclidean distance computations [9–10]. In this paper, a straightforward and efficient approach is presented for estimating the average speed of moving vehicles using image sequences captured by a stationary wide-angle camera positioned on a two-lane road. The captured frames are taken from an oblique angle to provide comprehensive lane coverage. The proposed algorithm is designed for real-time implementation, without the need for specialized hardware or complex calibration procedures. As illustrated in the flowchart in Figure 1, the estimation process involves several stages. Initially, the system receives a video input. In the first phase, the user performs a one-time interactive setup to define essential parameters such as lane boundaries and reference distances. Once configured, the system begins the estimation process by converting each RGB frame to grayscale. Motion is then detected by subtracting consecutive frames and applying a threshold to generate a binary mask highlighting moving objects. Centroids of the resulting binary foreground are extracted within predefined regions of interest (ROIs). By calculating the time required for each vehicle to travel between two virtual lines, and knowing the real-world distance between those lines, the system can compute the vehicle's average speed.

The remainder of this paper is structured as follows: Section 2 details the interactive setup process, including lane border and distance definition, and the creation of ROIs. Section 3 explains the procedure for generating the foreground mask. Section 4 describes the centroid calculation method. Section 5 outlines the complete speed estimation algorithm. Section 6 presents the experimental results, and Section 7 concludes the paper.

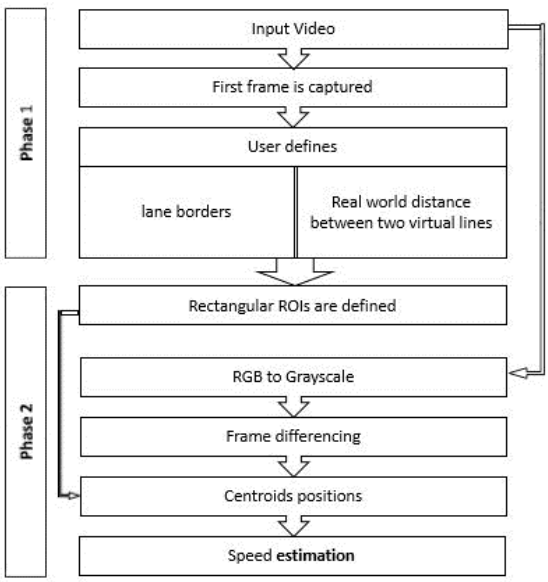

Fig. 1. Flow chart of the speed measurement steps

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.

Fig. 2. User-defined parameters

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.

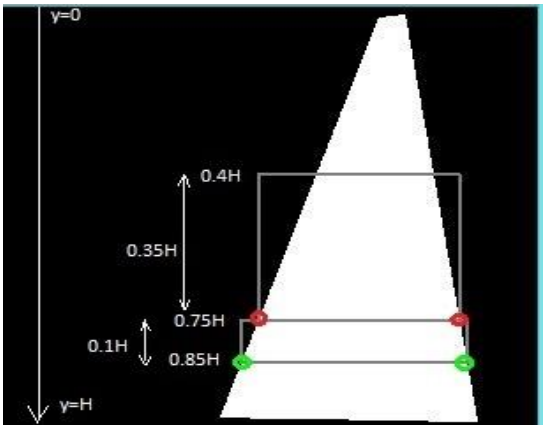

Fig. 3. Right lane binary mask, red circles as lower vertices of the upper rectangle, green circles as lower vertices of the lower rectangle

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).

$$Grayscale = 0.299\,R + 0.587\,G + 0.114\,B \tag{1}$$

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).

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

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

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.

$$x = m_{10}/(\sum_x \sum_y\ image(x,y)\ ) \tag{4}$$

$$y = m_{01}/(\sum_x \sum_y\ image(x,y)\ ) \tag{5}$$

$$Centroid = (x,y) \tag{6}$$

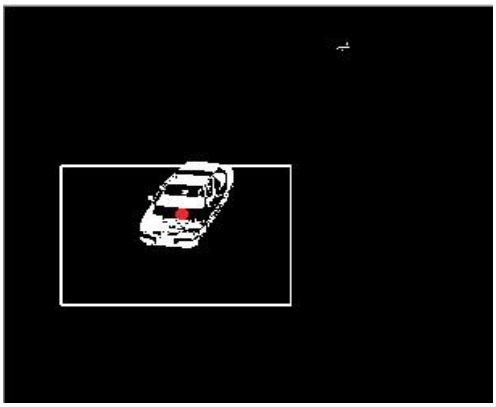

Fig. 4. Centroid of the binary foreground in the ROI

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.

$$Speed_{km/h} = (d \times frame\ rate \times 3.6)/n \tag{7}$$

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

```
1) Initialize n =0 , get d
2) IF (upper centroid) passed (border1) AND (upper norm) > (threshold1)
3)    n + 1 ⟶ n
4)    IF (lower centroid) passed (border2) AND (lower norm) > (threshold2)
5)        GOTO step 8
6)    ELSE
7)        GOTO step 3
8) RETURN n
9) RETURN (d . frame rate . 3.6 )/n
```

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.

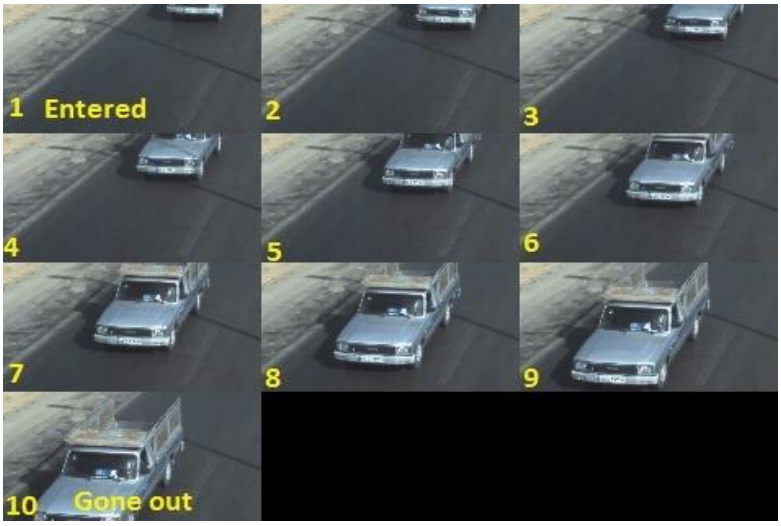

Fig. 6. Passing car in several frames

Table 1. Estimated and real speeds of 6 vehicles

Vehicle Number	Estimated Speed	Real Speed
# 1	78.55 km/h	78.54 km/h
# 2	86.40 km/h	86.40 km/h
# 3	96.00 km/h	96.00 km/h
# 4	54.00 km/h	61.71 km/h
# 5	54.00 km/h	61.71 km/h
# 6	Not detected	96.00 km/h

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.

ACKNOWLEDGEMENT

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

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] Calitz, A., & Hill, M. (2020). Automated license plate recognition using existing university infrastructure and different camera angles. In Proceedings of the 12th International Conference on Computer Recognition Systems CORES 2020 (p. 4).

[2] Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y., & Morishima, K. (2017). KEGG: New perspectives on genomes, pathways, diseases and drugs. Nucleic Acids Research, 45(D1), D353–D361. https://doi.org/10.1093/nar/gkw1092

[3] Wang, K., Peng, X., Yang, J., Meng, D., & Qiao, Y. (2019). Region attention networks for pose and occlusion robust facial expression recognition. IEEE Transactions on Image Processing, 29, 4057–4069. https://doi.org/10.1109/TIP.2019.2956143

[4] Ghatak, S., Rup, S., Didwania, H., & Swamy, M. (2021). GAN-based efficient foreground extraction and HGWOSA-based optimization for video synopsis generation. Digital Signal Processing, 111, 102988. https://doi.org/10.1016/j.dsp.2021.102988

[5] Liu, S., Zhang, Y., Hu, Q., Liu, M., & Zhao, J. (2016). SAR image de-noising based on GNL-means with optimized pixel-wise weighting in non-subsample shearlet domain. Computer and Information Science, 10(1), 16–22. https://doi.org/10.5539/cis.v10n1p16

[6] Simon, G., & Tabbone, S. (2021). Generic document image dewarping by probabilistic discretization of vanishing points. In International Conference on Pattern Recognition.

[7] Lee, J., Go, H., Lee, H., Cho, S., Sung, M., & Kim, J. (2021). Ctrl-c: Camera calibration transformer with line-classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 16228–16237). https://doi.org/10.1109/ICCV48922.2021.01592

[8] Xu, W., & Zhang, F. (2020). FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter. IEEE Robotics and Automation Letters, 6, 3317–3324. https://doi.org/10.1109/LRA.2021.3064227

[9] Xia, Z., Gharbi, M., Perazzi, F., Sunkavalli, K., & Chakrabarti, A. (2020). Deep denoising of flash and no-flash pairs for photography in low-light environments. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2063–2072). https://doi.org/10.1109/CVPR46437.2021.00210

[10] Ramdania, D., Andrian, R., Irfan, M., Abidin, R., & Kaffah, F. M. (2021). On designing application of finding nearby Islamic boarding schools in West Java using Haversine formula and Euclidean distance algorithms. In 1st International Conference on Islamic Science and Technology (ICONISTECH 2019). https://doi.org/10.4108/eai.11-7-2019.2297517