Alexandria Engineering Journal (2017) xxx, xxx–xxx

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

A novel approach in real-time vehicle detection and tracking using Raspberry Pi

Mallikarjun Anandhalli *, **Vishwanath P. Baligar**

Department of Computer Science and Engineering, BVB College of Engineering and Technology, Hubballi 580031, India

Received 25 October 2016; revised 15 May 2017; accepted 10 June 2017

KEYWORDS

Motion analysis;
Moving vehicle detection;
Tracking;
Raspberry Pi and urban environment

Abstract Real-time vehicle detection, tracking and counting of vehicles is of great interest for researchers and is a need of the society in general for comfortable, smooth and safe movements of vehicles in cities. We propose a video image processing algorithm which detects, tracks and finds the number of vehicles on a road. We convert RGB video frame to HSV color domain, which helps in differentiating the colors of the vehicles more absolutely. The noise is removed from each frame. Detection of the vehicles is purely carried on color features of the vehicles. Vehicle tracking is done using Kalman filter with the data association. The number of vehicles running in a video or in a particular lane is determined. We propose a novel idea to detect, track and count the vehicles on a road and it has been implemented on Raspberry Pi 3 using OpenCV and C++. We have compared the results of the proposed method with that of rear-view vehicle detection and tracking method (Bin et al., 2014) and morphological operation method (Zezhong et al., 2013), and found that the proposed algorithm is more effective in terms of accuracy of vehicle detection and cost.

1. Introduction

Video processing systems use high performance computers and application oriented circuits in the current technology. It is because of the amount of data to be processed is large and processing this data in real time is a tedious task. Surveillance of traffic area is essential nowadays, because of rapid increase in the number of vehicles but road area remains the same causing congestion. Determining the density of the vehicles in the road and analysis of the surveillance of the video help a lot in traffic management systems [1–3].

Different techniques have been proposed to detect the vehicles and track them on desktop computers. The authors in the paper [1] detect vehicles based on rear lamp and license plate with dedicated traffic surveillance camera. The system uses stationary camera to capture the traffic video. They extract frames from the video sequences and work on each frame of the video. Different techniques are being considered to detect rear lamp and license plate. Once after the localization of the vehicle parts, these parts are combined using Markov Random Field to model the relationship. Later Kalman filters are used to track the vehicles to get the better results.

The authors of the paper [4] propose a model based on the Hybrid image template to detect vehicles. A template is made by extracting various features using 50–1000 frames. Vehicles

* Corresponding author.
E-mail addresses: malliarjun71@gmail.com (M. Anandhalli), vpbaligar@bvb.edu (V.P. Baligar).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

http://dx.doi.org/10.1016/j.aej.2017.06.008

Please cite this article in press as: M. Anandhalli, V.P. Baligar, A novel approach in real-time vehicle detection and tracking using Raspberry Pi, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.06.008

are detected in three stages using SUM-MAX procedure [4]. In each step the local frame region with maximal score is computed. The results of the paper demonstrate detection of all vehicles under certain conditions.

The authors of the paper [5] proposed an algorithm in detection of lane using Raspberry pi, Obstacle avoidance and remotely controlling the car using Raspberry Pi along with other hardware devices, such as HD camera with ultrasonic sensor. But still the algorithm needs a lot of improvement with respect to real time, accuracy of lane detection. Authors proposed a method to build a self-driving car. The technique was tried on a small autonomous car. We have tried to build a system to detect, track and count vehicles using Raspberry Pi and a Pi camera. The Pi camera has a resolution 5 Mp which supports 1080P30 to capture videos. The camera has a small circuit board which connects to the PI's Camera Serial Interface (CSI) via flexible ribbon cable. The proposed system is capable of operating online and offline. In offline mode, a prerecorded video streams are analyzed to generate a traffic results. In online mode it directly captures the video and process the video in real time.

Many systems have been developed to detect, track and count the vehicle on a road, which are of high cost and hence limits the number of systems to be used at different locations [6,7]. Hence, we come up with the new system which is portable, can change its capture rate at any given time and can access video data in real time through Mobile or Desktop computers in remote place. We introduce Raspberry Pi for detection of vehicles and track them. It also analyses the input video and can provide the vehicle count at any given time.

The main contributions of the paper are as follows:

1. A novel method is proposed to detect vehicles based only on the color of the vehicle and does not consider any other features to detect them thereby reducing the time cost without sacrificing the accuracy.
2. The proposed system model makes use of Raspberry Pi interfaced with USB or Pi camera instead of using powerful workstations, traffic surveillance cameras. This system is portable.
3. Raspberry pi is accessed by remote computers, android devices or laptops and makes provision for view selection. Later section explains about the system in detail.

2. Proposed methodology

The proposed algorithm is built on Raspberry Pi with USB Camera to capture the traffic scene. Also the pre-captured traffic videos can be run on the Raspberry Pi and can be analyzed. The Raspberry Pi along with its camera kept at a remote place to capture the traffic videos and it can be controlled by the desktop or laptop or through the android devices and can analyze the traffic data. To access the remote Raspberry Pi from computers or any devices, a static IP address is assigned to Raspberry Pi and is connected to the private network, so that we can access information about the traffic from any remote place.

Advantages of proposed system are as follows:

1. Raspberry Pi is accessible anywhere in the network.
2. Live streaming of video is accessed from any of the remote devices.
3. Reduces physical space and storage space by capturing the video, only when the movement is found.
4. Power efficient and cost-effective.
5. Possible to connect multiple cameras and process them in parallel.

The flow chart of the proposed algorithm is as shown in Fig. 1.

2.1. Raspberry Pi with Pi camera

The Raspberry Pi is a small credit card sized processor with 1.2 GHz 64-bit quad-core ARMv8 CPU with 1 GB of RAM and 16 GB (expandable to 128 GB) of Hard Disk. It needs approx 6 GB of Hard disk space to install the operating system called "Raspbian Jessie". It's capable of multiprogramming and helps to explore many things to users. It has the ability to communicate with other peripheral devices such as sensors, motors, and LED's. Raspberry Pi has 802.11n Wireless Lan.

Raspberry Pi can be connected to Pi camera or USB camera. The Raspberry Pi camera and USB Camera are used in the proposed system to explore and take the advantages of Raspberry Pi and dedicated camera in analyzing the traffic flow. An operating system called "NOOBS - New Out Of Box Software" is installed on the Pi. The Pi Camera supports 1080p30, 720p60 and VGA90 video modes as well as image capture. Camera Serial Interface (CSI) port is available on the Raspberry Pi to connect the Pi camera as shown in Fig. 2.

2.2. Conversion and filtering

The Pi camera and USB Camera capture a video in a standard color format (RGB) at a given frame rate. In order to segment the vehicle colors more absolutely we convert RGB format to HSV. To segment absolute colors of vehicles HSV color space is well suited because of wide range of color space and can differentiate easily. H defines Hue (color), S defines the saturation (intensity of the color) and V defines the (Luminosity) [8]. After conversion, the HSV format image is split into three separate channels and is processed. Each channel is processed separately and vehicle colors are segmented, filtering out the

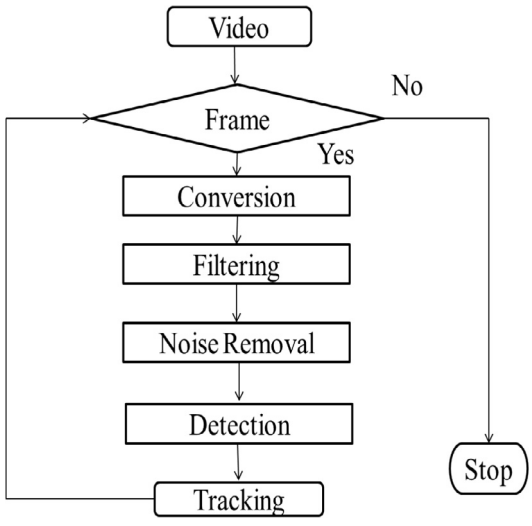

Figure 1 Flow diagram of the proposed algorithm.

Figure 2 Picture of Raspberry Pi and Pi Camera connected.

environmental colors. The algorithm for the color conversion from RGB to HSV format is shown below:

```
Step1: [Find the max and min values]
   M = max(R, G, B), m = min(R, G, B)
Step2: [normalized the RGB values to be in the range [0, 1]]
   r = (M-R)/(M-m)
   g = (M-G)/(M-m)
   b = (M-B)/(M-m)
Step3: [Calculate V value]
   V = max(R, G, B)
Step4: [Calculate S value]
   if M = 0 then S = 0 and H = 180 degrees
   if M <> 0 then S = (M - m) / M
Step5: [Calculate H value]
   if R = M then H = 60(b-g)
   if G = M then H = 60(2+r-b)
   if B = M then H = 60(4+g-r)
   if H >= 360 then H = H - 360
   if H < 0 then H = H + 360
      Where H in the range [0,360], S and V in the range [0,100]
Step6: [output HSV]
```

The results of each channel of HSV format are in binary, and OR operation is performed on these channels to get final resultant image as shown in Fig. 3.

2.2.1. Chain codes

Once the vehicle colors are segmented from the background colors, the white blobs in the image define that they are vehicles with noise included. Contour detection gives better localization of vehicle region when a binary image is given as input. Based on 8-connectivity, chain code method [9,10] is used for finding the contours. After detecting each of the objects with chain codes they are analyzed.

2.3. Convex hull

Each object is defined with the polygon shape, called Convex Hull [11]. This defines a definite shape to a blob in a minimal area [12,13]. The result of convex hull construction is shown in Fig. 4 corresponding to the irregular blobs of Fig. 3. Area of the objects is considered as a constraint to check whether the

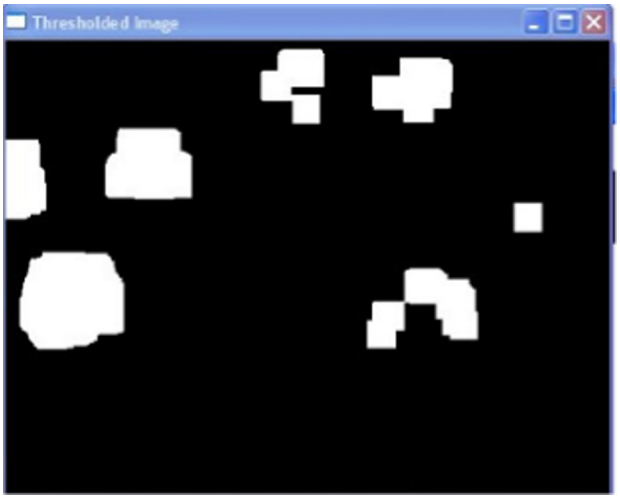

Figure 3 Result after image conversion and filtering.

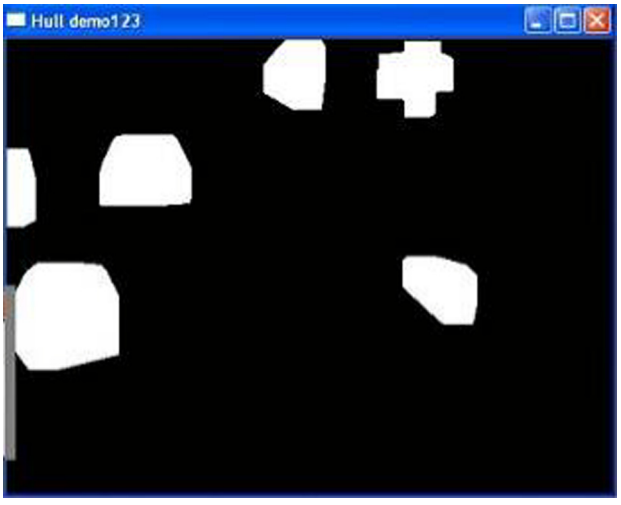

Figure 4 Result of Convex Hull.

blob is a vehicle or noise based on the threshold area set. In Fig. 3 and Fig. 4, we see that the noise is removed. Each object in the image is analyzed and processed.

2.4. Calculate the centroid position

The centroid of the each object is calculated and these centroid points are important in tracking the blobs. Kalman filter is implemented to track these centroids. Each centroid is tracked by a tracker (Kalman filter). Hence multiple Kalman filters are implemented in tracking the centroid points. Because of multiple Kalman filters, it becomes very important to allocate the same blob centroid point to the same Kalman filter. A technique called data association is used to allocate the centroid points to the Kalman filter.

2.5. Data association

Data association with respect to the proposed system is a process of associating a centroid points to Kalman filters. A cost matrix is built by the detected centroid points [14,15]. The cen-

troid points detected from the previous frame are subtracted with the centroid points of the current frame using Euclidean distance formula. Rows and columns of the matrix define distance and tracker. The problems faced during the data association are as follows:

1. Track maintenance.
2. Target detection.
3. Single or multiple targets.

Overall procedure:

1. Make observations

Association of centroid points from the previous frame and current frame in a matrix from.

2. Predict the measurements

From the associated matrix, check all the points whether they are in the area. This is used to narrow the search.

3. Check whether a measurement lies in the gate.

If yes, then it is a valid centroid point for pairing with current centroid frame. The algorithm is almost similar to the Hungarian algorithm but differs in preparation of cost matrix.

2.6. Kalman filter

Prediction and measurements are the two step process in Kalman filter. Prediction step estimates the current state and the measurement state evaluates the predicted estimate and updates the equation accordingly. The algorithm is recursive and it works for real-time data [16]. The Kalman filter tracks the associated centroid points.

The Kalman filter model assumes that the state of a system at a time 't' evolved from the prior state at time '$t-1$' according to the equation

$$X_t = F_t x_{t-1} + B_t u_t + w_t \quad (1)$$

where

- X_t is the state vector containing the terms of interest for the system (e.g., position, velocity and heading) at time 't'.
- u_t is the vector containing any control inputs (steering angle, throttle setting, braking force).
- F_t is the state transition matrix which applies the effect of each system state parameter at time '$t-1$' on the system state at time 't' (e.g., the position and velocity at time $t-1$ both affect the position at time t)
- B_t is the control input matrix which applies the effect of each

Measurements of the system can also be performed, according to the model

$$Z_t = H_t x_t + v_t \quad (2)$$

where

- Z_t is the vector of measurements.
- H_t is the transformation matrix that maps the state vector parameters into the measurement domain.
- v_t is the vector containing the measurement noise terms for each observation in the measurement vector. Like the process noise, the measurement noise is assumed to be zero mean Gaussian white noise with covariance R_t.

3. Results and discussion

In this section, we present experimental results of the approach using the data set captured and live streaming video from the Raspberry Pi camera. The Pi camera requires proper parameter values for good performance such as fps and resolution.

The Raspberry pi can be accessed by any of the computers, laptop and Mobile etc. remotely and can get to know the vehicle data at real time. Our algorithm is mainly implemented on Raspberry Pi with the Pi Camera and it is used to capture the video with C++ and OpenCV libraries. Integrating OpenCV with C++ IDE on Raspberry pi is the difficult part than Desktop. The Raspberry Pi is configured with a 1 GB RAM and a 16-GB memory device. The experiments were performed on our videos captured in urban traffic environment, highway traffic scenes and also on the standard datasets.

To collect the live data set, the Raspberry Pi is connected to the power bank and USB camera. The live streaming video input is shown in Figs. 5, 17a, 17b and 19. There is no wired connection between the desktop computer and Raspberry Pi. This live video stream can be accessed by the remote system with the static IP address assigned to Raspberry Pi. The Captured live video stream is processed to the proposed system to analyze the traffic video stream and results of the algorithm are shown (see Fig. 6).

The standard data set is considered for the test that has frame rate at 25 fps, 30 fps. To check the robustness of the algorithm the few videos were captured using mobile Phone at 31 fps (Samsung Smartphone of 5 Mp) (Figs. 7 and 8) and few other data sets captured were at 31 fps from the Raspberry pi USB camera of 8 Mp (Figs. 17a and 17b). The videos have different resolution with different angle, height of the Pi camera. The proposed work is on vehicle detection based on color features, so dynamically the color thresholds can be set at any time for better color feature extraction. The algorithm is not confined to the static cameras only but also to the dynamically moving cameras and shaking cameras as shown in Figs. 10, 12, and 13. While capturing the Video sequence 1 and Sequence 2

Figure 5 Photography of Frame of Live Video Sequence Capturing using Pi and Camera.

Figure 6 Output frame of Video Sequence 1 at 25 fps with side view and Top view of the vehicle captured on the Highway 1.

Figure 7 Output Frame of urban road captured using mobile sequence 1 at 31 fps with multi view.

from the mobile, it had a varying position slowly (Figs. 7 and 8). In the video sequence (Fig. 10) the camera was moving constantly with rapid change in the environment as well because of the movement of camera and vehicle as shown. The proposed system detects vehicles irrespective of their view point either it may be a rear or a front or top view (see Figs. 9, 11, 14–16).

The Fig. shown in [6–17] represents the proposed system that works with traffic input data sets. Ten traffic video data sets of different scenarios were collected to evaluate the algorithm with high resolution and the low resolution. For video sequence of different resolution the proposed algorithm works well.

To check the algorithm implemented on Raspberry pi a trial was given on urban roads in India. An Opencv and code blocks were interfaced on the Raspberry pi to implement the algorithm using C++ and Opencv. The Figs. shown in 17a and 17b are the live video sequence taken from Pi. With the remote computer we have accessed this data and have taken the screenshots of the results of the algorithm. The screenshot of the remote desktop is shown in Fig. 18. USB camera was also interfaced to the pi device and placed on edge of the road to capture the live video data and stream the same to the algorithm as shown in Fig. 19. The pi is powered with the power bank. The results were accessed from the remote computer.

The advantages of having Raspberry pi are as follows:

1. Real-time video process in the video network and density of vehicles can be broadcasted instantaneously.
2. Can access the real-time video data using any remote computers, laptops, mobiles etc.
3. Portable can be placed on any roads for traffic surveillance.

3.1. Performance

The results of vehicle detection run on the Raspberry pi are represented in Table 1. The performance of the proposed algorithm is good with respect to detection of vehicle and track them and to determine the number of vehicles. The system is compared with existing systems and is explained in the section Comparative Study.

Figure 8 Output Frame of urban road captured using mobile sequence 2 at 31 fps with multi view.

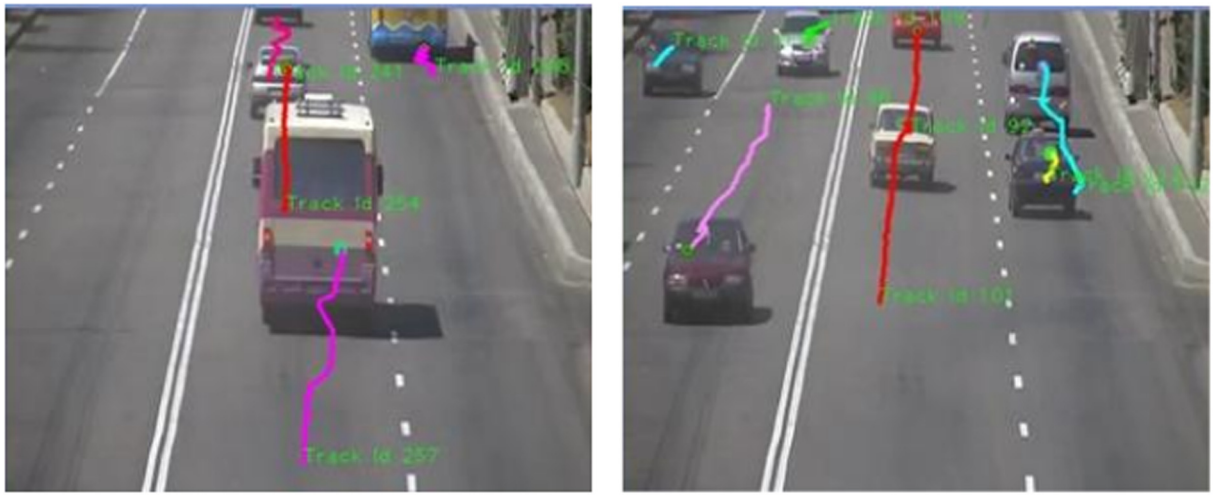

Figure 9 Output Frame of Video Sequence within the city limits at 25 fps with front and rear view.

Figure 10 Moving camera capturing rear view of the vehicle of Video Sequence at 30 fps.

Figure 12 Output Frame of Complex Background Video Sequence with less no. of time camera shakes.

Figure 11 Output Frame of Underpass Video Sequence captured at 25 fps with side view and top view.

Figure 13 Output Frame of Complex Background Video Sequence with more no. of time camera shakes.

Please cite this article in press as: M. Anandhalli, V.P. Baligar, A novel approach in real-time vehicle detection and tracking using Raspberry Pi, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.06.008

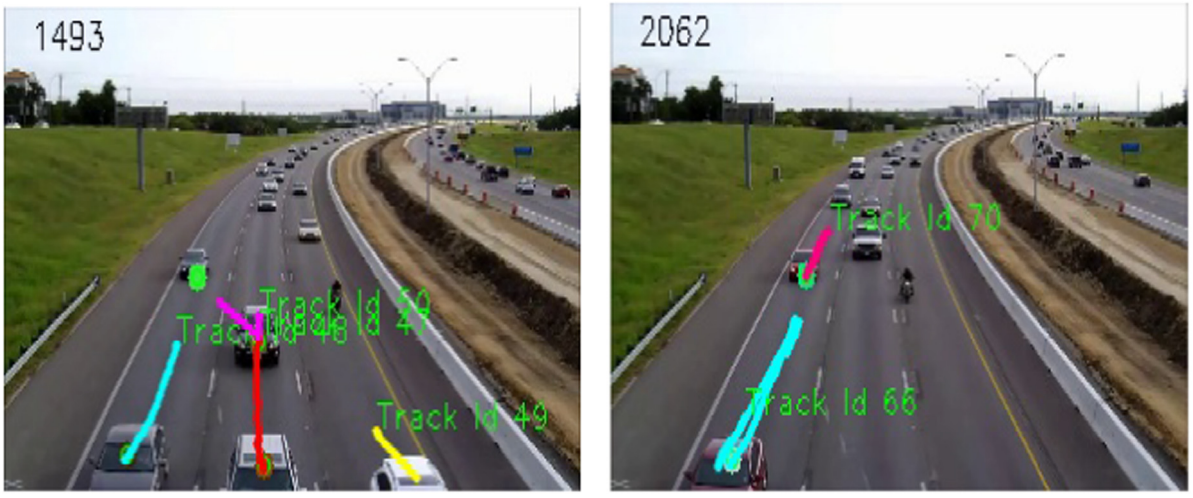

Figure 14 Output Frame of on Highway with fast moving vehicles.

Figure 15 Output Frame of Rainy Video Sequence.

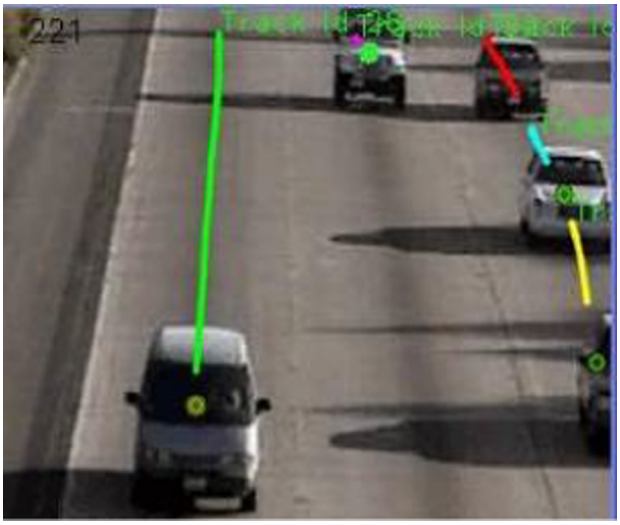

Figure 16 Output Frame of Video Sequence Vehicles with intense shadows.

Figure 17a Live Video Sequence 7 at 25 fps.

Figure 17b Live Video Sequence 7 at 25 fps.

Please cite this article in press as: M. Anandhalli, V.P. Baligar, A novel approach in real-time vehicle detection and tracking using Raspberry Pi, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.06.008

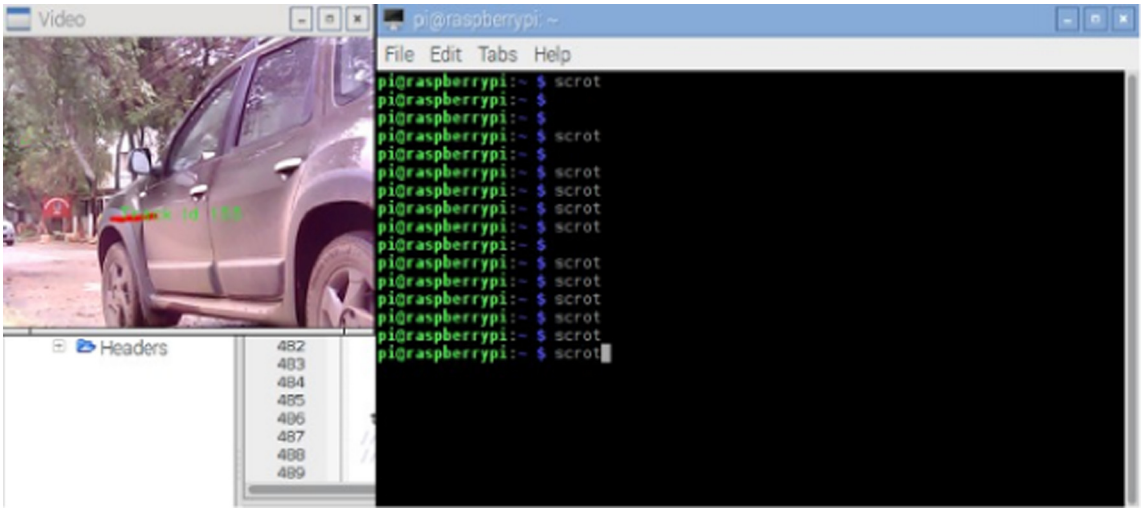

Figure 18 Screenshot of remote computer accessing Raspberry Pi.

Figure 19 Photography of Pi taken along with USB camera powered with power bank placed along road side at remote place.

Table 1 Tabular representation of results.

Sl. no	Resolution/video sequences/time duration	TP	FP	FN	Correctness	Completeness	Quality
1	320 * 240/static camera highway 1(2.57 min)	18	0	0	100	100	100
2	640 * 480/mobile sequence 1(1 min)	2	0	0	100	100	100
3	640 * 480/mobile sequence 2 (1 min)	3	0	0	100	100	100
4	640 * 480/moving camera (1 min)	1	0	0	100	100	100
5	320 * 240/underpass video sequence (3 mins)	23	1	0	95.83	100	95.83
6	640 * 480/within the city limits (12 mins	230	27	0	89.4	100	89.4
7	720 * 640/complex background with less no. of time camera shake (3:25 s)	75	4	3	94.93	96.15	91.64
8	720 * 640/complex background with more no. of time camera shake (2:25 s)	45	3	5	93.75	90	84.09
9	720 * 640/on highway with fast moving vehicles	71	0	0	100	100	100
10	Live Pi and USB camera video sequence	5	0	0	100	100	100
11	320 * 240/rainy video sequence (20 s)	13	0	0	100	100	100
12	640 * 480/vehicles with intense shadows (30 s)	93	3	0	96.875	100	96.875

Please cite this article in press as: M. Anandhalli, V.P. Baligar, A novel approach in real-time vehicle detection and tracking using Raspberry Pi, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.06.008

Wiedemann proposed three different categories in paper [17] to check the performance. The parameters to measure correctness, completeness and quality of the system are as follows:

1. True Positive (TP): The number of correctly detected true vehicles.
2. False Positive (FP): The number of correctly detected false vehicles.
3. False Negative (FN): The number of vehicles that are not detected.

$$Correctness = \frac{TP}{TP + FP} * 100\%$$

$$Completeness = \frac{TP}{TP + FN} * 100\%$$

$$Quality = \frac{TP}{TP + FP + FN} * 100\%$$

3.2. Computational cost

The algorithm is implemented with Opencv [19] and C++ on the processor configured with 2.5 GHz, dual core processor and a 4-Gb ram. The algorithm is evaluated based on four steps

1. Vehicle color extraction.
2. Filtering and shaping of blob.
3. Data association.
4. Vehicle tracking this gives the time taken for each frame by the system.

The computational cost of the algorithm is shown in Table 2. As in the table presented the algorithm is time effective and needs 72.7 ms to process one frame and Raspberry Pi needs 81.589 ms which are considered for real time. This is the average time taken to complete processing of a frame in Raspberry pi and desktop computer. The advantage of Raspberry pi over desktop computer can reduce the storage memory by capturing those videos where the movement is found, portable and any dynamical changes in the program and camera can be made at real time using remote computers or laptops.

The same algorithm with same input video run on the Raspberry Pi to compare the performance and the average time taken by Raspberry Pi and Dual core desktop processor is shown in Table 2.

3.3. ROC

The receiver operating characteristic (ROC) curve of vehicle detection is drawn in Fig. 20. We tested all data sets in different scenarios to check the reliability of the proposed algorithm. To determine the ROC curve the experimental results are

Table 3 Probability of TP and FN.

Sl. No.	P(TP)	P(FN)
1	1	0
2	1	0
3	1	0
4	1	0
5	0.958	0.042
6	0.895	0.105

Table 2 Computational cost of the proposed algorithm.

Sl No.	Steps of proposed algorithm	Run time (in s)	
		Desktop computer	Raspberry Pi
1	Color conversion	0.01690	0.017579
2	Filtering and shaping of blob	0.0163	0.01909
3	Data association	0.022	0.0249
4	Vehicle track (Kalman filter)	0.0175	0.02002
5	Total time	0.0727	0.081589

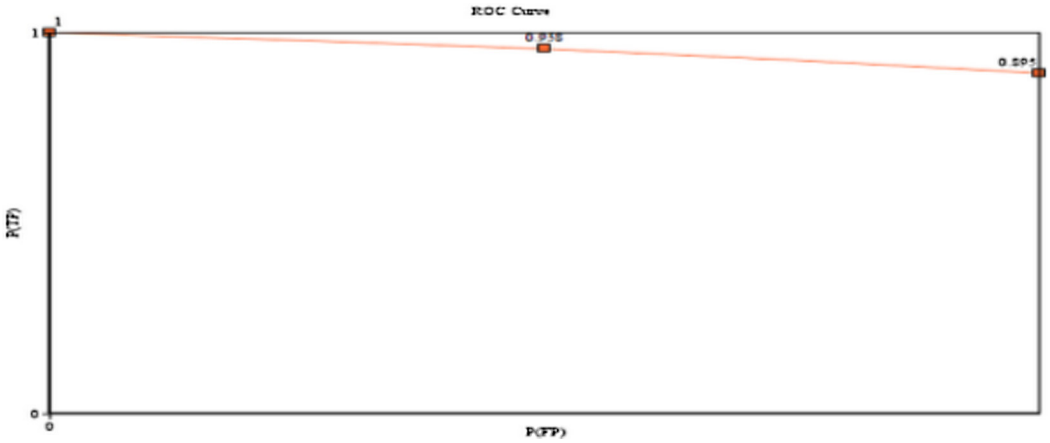

Figure 20 ROC curve for vehicle detection.

Table 4 Comparative analysis with different algorithms.

View algorithm	Detection accuracy		
	Side view and top view	Front and rear view	Multi view
Proposed algorithm	96%	92.8%	93%
Rear-view vehicle detection and tracking by combining multiple parts for complex urban surveillance [1]	Not considered	92%	Not considered
Air-borne moving vehicle detection for video surveillance of urban traffic [18]	89%	Not considered	Not considered
Morphological operation [19]	91.5%	Not considered	Not considered

required shown in Table 3. As shown in Fig. 20 the probability TP rate decreased when the probability FP rate exceeded a certain limit. With the ROC curve, we can determine how good the algorithm is for all scenarios. Since the system did not have any miss detection true negative is always zero. To determine the ROC curve we have to calculate the probability of true positive and probability of false negative.

3.4. Comparative study

The proposed algorithm is compared with different algorithms [1,18,9] in terms of detection accuracy with respect to different views: Side-Top view, Front-Rear view and Multi view. The proposed algorithm is more effective when compared to other algorithms in terms of accuracy and view. The proposed system detects vehicles irrespective of their view point either it may be a rear or a front or top view. System determines the number of vehicles running on the road in real time. The vehicle detection accuracy achieved is 93%, 96% and 92% for Multi view, Side-Top view and Front-Rear view respectively. It also addresses the partial occlusion problems. Table 4 represents the comparative study made with different algorithms in terms of accuracy and view.

4. Conclusion

A system with Raspberry Pi and USB camera is being used for real-time vehicle detection, tracking and counting. The density of the vehicles running in the particular road is determined in real time. The results of the proposed method in terms of its accuracy and time taken are better compared to rear-view vehicle detection and tracking method [1] and morphological operation method [19]. Because of the static IP address assigned to the Raspberry system it enables us to communicate it with other remote computers. The performance of the proposed system is found superior by 6% to 8% when compared to rear-view vehicle detection and tracking method [1] and morphological operation method [19]. It has been experienced that the cost of the proposed system is much less than the existing systems. Detection of the vehicle and tracking made by the system is reliable. The proposed method considered only color features of the vehicle and made it replace existing systems. Numbers of vehicles present in the video are calculated in real time.

5. Future work

The experimental results demonstrated on challenging data set considered under different camera angles, Rear View of the Vehicles and at different camera heights show flexibility and good accuracy of the proposed method. One future works that would make the system more reliable is classification of vehicles. Classification method improves the detection performance further, so it is useful to add a classification process. That is one of our future works.

References

[1] Bin Tian; Ye Li; Bo Li; Ding Wen, Rear-view vehicle detection and tracking by combining multiple parts for complex urban surveillance, in: IEEE Transactions on Intelligent Transportation Systems, vol.15, no.2, pp. 597–606 (April 2014).
[2] Ye Li; Bo Li; Bin Tian; Qingming Yao, Vehicle detection based on the and– or graph for congested traffic conditions, in: IEEE Transactions on Intelligent Transportation Systems, vol.14, no.2, pp.984–993 (June 2013).
[3] M. Anandhalli, V.P. Baligar, Improvised approach using background subtraction for vehicle detection, in: Advance Computing Conference (IACC), 2015 IEEE International, pp. 303–308, 12–13 (June 2015).
[4] Ye Li; Bo Li; Bin Tian; Fenghua Zhu; Gang Xiong; Kunfeng Wang, Vehicle detection based on the deformable hybrid image template, in: 2013 IEEE International Conference on Vehicular Electronics and Safety (ICVES), pp.114–118 (28–30 July 2013).
[5] F. Yamazaki, W. Liu, Vehicle extraction and speed detection from digital aerial images, Proc. IEEE IGARSS 3 (2008) 1334–1337.
[6] S. Li, H. Yu, J. Zhang, K. Yang, R. Bin, Video-based traffic data collection system for multiple vehicle types, IET Intell. Transp. Syst. 8 (2) (March 2014) 164–174, http://dx.doi.org/10.1049/iet-its.2012.0099.
[7] R. O'Malley, E. Jones, M. Glavin, Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions, Intell. Transport. Syst. IEEE Trans. 11 (2) (June 2010) 453–462, http://dx.doi.org/10.1109/TITS.2010.2045375.
[8] N.A. Ibraheem, M.M. Hasan, R.Z. Khan, P.K. Mishra, Understanding color models: a review, ARPN J. Sci. Technol. 2 (3) (2012).
[9] Michael B. Dillencourt, Hannan Samet, Markku Tamminen, A general approach to connected-component labeling for arbitrary image representations, J. ACM. 39 (2) (1992) 253.
[10] H. Samet, M. Tamminen, Efficient component labeling of images of arbitrary dimension represented by linear bintrees, IEEE Trans. Pattern Anal. Mach. Intell. (1988).
[11] A.M. Andrew, Another efficient algorithm for convex hulls in two dimensions, Inform. Proc. Lett. 9 (5) (1979) 216–219, http://dx.doi.org/10.1016/0020-0190(79)90072-3.
[12] S. Suzuki, K. Abe, topological structural analysis of digitized binary images by border following, CVGIP 30 (1) (1985) 32–46.
[13] Sklansky, J., Finding the Convex Hull of a Simple Polygon. PRL 1 $number, pp 79-83 (1982).

[14] W. Harold Kuhn, The Hungarian Method for the assignment problem, Naval Res. Log. Quart. 2 (1955) 83–97.
[15] R.E. Burkard, M. Dell'Amico, S. Martello, Assignment Problems (Revised reprint). SIAM, Philadelphia, PA, 2012.
[16] Greg Welch, Gary Bishop, An Introduction to the Kalman Filter, 1995.
[17] C. Wiedemann, C. Heipke, H. Mayer, O. Jamet, EmPirical evaluation of automatically extracted road axes, in: K. Bowyer, P. Phillips (Eds.), EmPirical Evaluation Methods in Computer Vision, IEEE Comput. Soc. Press, New York, NY, USA, 1998, pp. 172–187.
[18] Renjun Lin, Xianbin Cao, Yanwu Xu, Changxia Wu, Hong Qiao, Air-borne moving vehicle detection for video surveillance of urban traffic, in: Intelligent Vehicles Symposium, 2009 IEEE, pp. 203–208 (3–5 June 2009).
[19] Zezhong Zheng; Guoqing Zhou; Yong Wang; Yalan Liu; Xiaowen Li; Xiaoting Wang; Ling Jiang, A novel vehicle detection method with high resolution highway aerial image, in: IEEE Journal of Selected ToPics in Applied Earth Observations and Remote Sensing, vol. 6, no. 6, pp. 2338–2343 (Dec. 2013).

Please cite this article in press as: M. Anandhalli, V.P. Baligar, A novel approach in real-time vehicle detection and tracking using Raspberry Pi, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.06.008