International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - ICSCAAIT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

Vehicle Detection and Counting Method Based on Digital Image Processing in Python

Reha Justin[1], Dr. Ravindra Kumar[2]
[1]Intern, [2]Principal Scientist, CSIR-Central Road Research Institute, Transportation Planning Division Delhi, India
[1]reha.justin96@gmail.com, [2]ravindra261274@gmail.com

Abstract: Vehicle counting process provides appropriate information about traffic flow, vehicle crash occurrences and traffic peak times in roadways. An acceptable technique to achieve these goals is using digital image processing methods on roadway camera video outputs. This paper presents a vehicle counter-classifier based on a combination of different video-image processing methods including object detection, edge detection, frame differentiation and the Kalman filter. An implementation of proposed technique has been performed using python programming language. This paper describes the methodology used for image processing for traffic flow counting and classification using different library and algorithm with real time image.

Keywords: Vehicle Counting, Vehicle Detection, Traffic Analysis, Object Detection, Video-Image Processing.

I. INTRODUCTION

Vehicles detection and counting are done a non-intrusives sensor/manual method/video infrared, magnetic, radar, ultrasonic, acoustic, and video imaging sensors and intrusive sensor sensors include pneumatic road tube, piezo-electric sensor, magnetic sensor, and inductive loop . Non inrusive technology has advantages over intrusive technology which requires closing of traffic lanes and put construction workers in harm's way, stop traffic or a lane closure and non-intrusive sensors are above the roadway surface and don't typically require a stop in traffic or lane closure. Both types of sensors have advantages and disadvantages . But the accuracy from video or digital counting manually is very high as compared to other technology.

However the image processing is time consuming and requires some automation to save the time for image count and classification. In current era of python type programming language has much addition of image processing and time saving for vehicle detection, counting and classification. The paper states about way to image process, type of filter used and proposed technique are able to detect, count and classify the image accurately.

II. BACKGROUND INFORMATION

A. Video Processing:

Video processing is a subcategory of Digital Signal Processing techniques where the input and output signals are video streams. In computers, one of the best ways to reach video analysis goals is using image processing methods in each video frame. In this case, motions are simply realized by comparing sequential frames[7]. Video processing includes pre-filters, which can cause contrast changes and noise elimination along with video frames pixel size conversions[6]. Highlighting particular areas of videos, deleting unsuitable lighting effects, eliminating camera motions and removing edge-artifacts are performable using video processing methods[29]. OpenCv library of python is equipped with functions that allow us to manipulate videos and images. OpenCV-Python makes use of Numpy, which is a library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.[34]

B. RGB to Grayscale Conversion:

In video analysis, converting RGB color image to grayscale mode is done by image processing methods. The main goal of this conversion is that processing the grayscale images can provide more acceptable results in comparison to the original RGB images[11]. In video processing techniques the sequence of captured video frames should be transformed from RGB color mode to a 0 to 255 gray level. When converting an RGB image to a grayscale mode, the RGB values for each pixel should be taken, and a single value reflecting the brightness percentage of that pixel should be prepared as an output[2].

C. Power-Law Transformation:

Enhancing an image provides better contrast and a more detailed image as compared to a non-enhanced one. There are several image enhancement techniques such as power-law transformation, linear method and Logarithmic method. Image enhancement can be done through one of these grayscale transformations. Among them, power-law transformation method is an appropriate technique which has the basic form below.

$$V = A\, v\, \gamma \quad (1)$$

Where V and v are output and input gray levels, γ is Gamma value and A is a positive constants (in the common case of A=1). The python code that implements power law transformation is-

power_law_transformation=cv2.pow(gray,0.6)

The second argument is the gamma value. Consequently, choosing the proper value of γ can play an important role in image enhancement process and preparing suitable details identifiable in image.

D. Canny Edge Detection:

Object detection can be performed using image matching functions and edge detection. Edges are points in digital images at which image brightness or gray levels changes suddenly in amount.[33] The main task of edge detection is locating all pixels of the image that correspond to the edges of the objects seen in the image. Among different edge detection methodologies, Canny algorithm is a simple and powerful edge detection method. Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy)[9]. From these two images, we can find edge gradient and direction for each pixel as follows:

$$\text{Edge\_Gradient}(G)=\sqrt{G2x+G2y} \quad (2)$$

$$\text{Angle}(\theta)=\tan^{-1}(Gy/Gx) \quad (3)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions. After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. OpenCV puts all the above in single function, cv2.Canny() [12].

E. The Kalman Filter:

Images typically have a lot of speckles caused by noise which should be removed by the means of filtration. The Kalman filter is a powerful and useful tool to estimate a special process using some kind of feedback information[14]. The Kalman filter is used to provide an improved estimate based on a series of noisy estimates. This filter specifies that the fundamental process must be modeled by a linear dynamical structure:

$$x_k = F_{k-1}x_{k-1} + w_{k-1} \quad (4)$$

$$y_k = H_k x_k + v_k \quad (5)$$

Where x_k and y_k are the state and measurement vectors, w_k and v_k are the process and measurement noise, F_k and H_k are the transition and measurement values and k is desired time step[28]. The Kalman filter also specifies that the measurements and the error terms express a Gaussian distribution, which means in vehicle detection each vehicle can only be tracked by one Kalman filter [22],[31]. Therefore the number of Kalman filters applied to each video frame depends on the number of detected vehicles.

III. PREVIOUS WORKS

Using image/video processing and object detection methods for vehicle detection and traffic flow estimation purposes has attracted a huge attention for several years. Vehicle detection/tracking processes have been performed using one of these methodologies[8]:

- Matching
- Threshold and segmentation
- Point detection
- Edge detection
- Frame differentiation
- Optical flow methods

It can be said that one of the most important researches in object detection fields, which has resulted in the auto-scope video detection systems is introduced in [15]. In some works such as [21], forward and backward image differencing method used to extract moving vehicles in a roadway view. Some studies like [17] and [4] proved that the use of feature vectors from image region can be extremely efficient for vehicle detections goals. Some others represented the accurate vehicle dimension estimation using a set of coordinate mapping functions as it can be seen in [16]. Furthermore, some studies have developed a variety of boosting algorithms for object detection using machine learning methods which can detect and classify moving objects by both type and color such as [18] and [19]. Named approaches have both their advantages and disadvantages.

IV. PROPOSED TECHNIQUE

Different from previous works, the method proposed in this paper uses a combination of both "Frame Differentiation" and "Edge Detection" algorithms to provide better quality and accuracy for vehicle detection. By using the Kalman filter, position of each vehicle will be estimated and tracked correctly. This filter also used to classify detected vehicles in different specified groups and count them separately to provide a useful information for traffic flow analysis. The

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - ICSCAAIT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

flowchart of the method is represented in Figure 1.

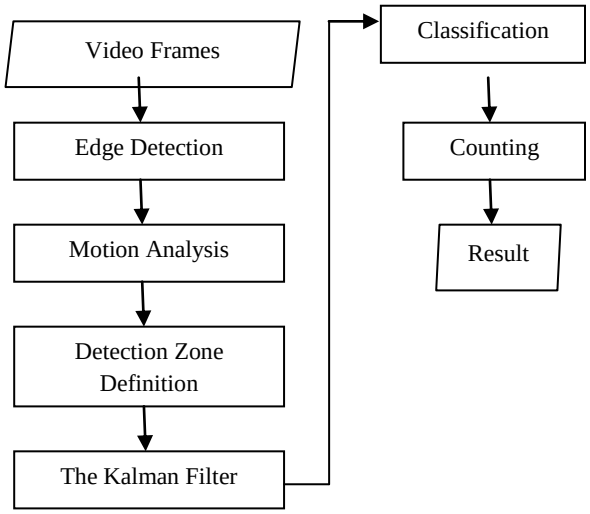

Fig. 1. Flowchart of the Technique

Based on Figure 1, the technique includes these steps: image enhancement process, edge detection, motion analysis using a combination of different techniques, detection zone definition, the Kalman filter, vehicle type classification and counting. It is necessary to say that some assumptions made in this work:

- No sudden changes of directions are expected
- No car accidents and crashes are expected
- There is both physical and legal limitations for vehicles
- motion scenes are captured with a view from above to the roadway surface

The proposed technique to detect and count vehicles is presented as below:

A. Grayscale Image Generation and Image Enhancement:

To get better results, vehicle detection process should be performed in the grayscale image domain. Hence a RGB to grayscale conversion is performed on each video frame. To achieve an appropriate threshold level and make results more suitable than the input image, each frame should be brought in contrast to background. Among several grayscale transformations, power-law method has been used in this work. For color conversion we use the function cv2.cvtColor(input_image,flag) where flag determines the type of conversion. To convert to grayscale we use flag cv2.COLOR_BGR2GRAY. Experimental results in different situations showed that the best results appear when γ value is set to 0.6 as it can be seen in Figure 2. This figure shows the result of applying different γ values to grayscale converted image, where section A is the input RGB color frame and B, and C are grayscale versions with gamma values 0.6 and 0.9, respectively. The implementation of Figure 2 results can be obtained by using the python code shown in Figure 3.

Fig. 2. Input RGB Video Frame (A) and Grayscale Converted With Different Γ Values (B and C)

```
while True:

    ret, frame = cap.read()

    if ret == True:

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray =gray/255.0
        power_law_transformation=cv2.pow(gray,0.6)
        cv2.imshow('tranformed',power_law_transformation)
```

Fig. 3. Code for Conversion from RGB to Grayscale and Image Enhancement

B. Edge Detection:

Each image (video frame) has three significant features to achieve detection goals. These features include: edges, contours and points. Among mentioned features, an appropriate option is to use edge pixels. Processing of image pixels enables us to find edge pixels, which are the main features of passing vehicles in a roadway video frame. One of the most common ways to find the edges of an image is to use Canny operator which has been used in this work. The result is presented in Figure 4 and the corresponding code is presented in Figure 5. As it can be seen the output result of edge detection process is demonstrated in a binary image (threshold) with the detected edge pixels.

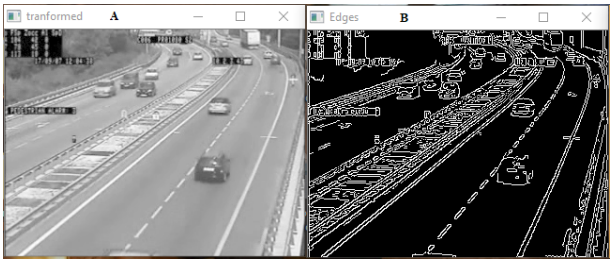

Fig. 4. A: Original Image B: Edge Detection Result

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - ICSCAAIT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

```
hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
lower_red=np.array([30,150,50])
upper_red=np.array([255,255,180])
mask=cv2.inRange(hsv,lower_red,upper_red)
res=cv2.bitwise_and(frame,frame,mask=mask)
edges=cv2.Canny(frame,100,200)
cv2.imshow('Edges',edges)
```

Fig. 5. Code for Canny Edge Detection

The next step is to extract moving edges from sequential video frames and process the resulting edge information to obtain quantitative geometric measurements of passing vehicles.

C. Background Subtraction:

Using provided threshold, the static parts of sequential video frames should be cleaned. The main challenge here is that the performance of image analysis algorithms suffers from darkness, glare, long shadows or bad illumination at night, which may cause strong noises [3], [13]. Therefore, the grayscale image might be unspecified in these situations and make the detection task a bit more complex. Edges essentially separate two various regions which are static region (the roadway) and dynamic region (moving vehicles). The static background is then deleted to locate moving objects in each frame. The result zone leaves only vehicles and some details as moving objects in sequential images which are changing frame to frame. A combination of forward and backward image differencing method and Sobel edge detector has been used in this work. According to this method, three sequential frames are chosen and the middle one should be compared to its previous and next frames. Consequently, extracted edges of each frame detected by Canny edge detection achieved from previous section are used here. Then the differences of frames can be obtained by subtracting each two sequential pair of generated binary images, as in equation 6:

BinaryImage (Canny (F_{n-1}) ∩ Canny (F_n)) – **BinaryImage** (Canny (F_n) ∩ Canny (F_{n+1})) (6)

Where F_{n-1} is previous frame, F_n is current frame and F_{n+1} is the next frame. This process continues to the last three sequential video frames. The output result is demonstrated in Figure 6. The python code is represented in Figure 7. In this figure A, B and C represent three sequential frames, where D demonstrates the output background subtraction method. Using this technique moving vehicles are detected in three sequential frames.

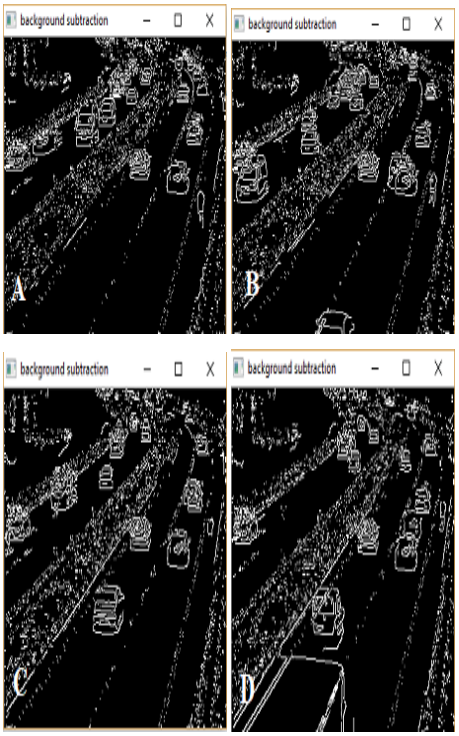

Fig. 6. Proposed Moving Vehicle Detection Technique and Background Subtraction (A,B,C and D)

```
fgbg=cv2.BackgroundSubtractorMOG()

fgmask=fgbg.apply(edges)
cv2.imshow('background subtraction',fgmask)
if cv2.waitKey(30) & 0xFF == ord('q'):
    break
```

Fig. 7. Code for Background Subtraction

D. Detection Zone:

As an observation (detection) zone, a region should be defined to display moving vehicle's edges in a bounding box at the time that the vehicle enters it. This zone is in the middle of the screen and covers 1/3 of its height and 3/5 of its width (considering minimum and maximum available size of detectable passing vehicles in pixels). This area which contains the most traffic can embed both small and long vehicles and the main goal of defining it is to avoid perspective challenges and wrong type counts. Based on proposed method in background subtraction level, a vehicle is detected in three sequential frames. When a moving vehicle is detected, a bounding box whelming vehicle borders in binary image is drawn.

E. The Kalman Filter:

The bounding boxes could also be used to count and classify passing vehicles. This can be done by the Kalman filtering technique. In roadway videos, the edge detection

function provides an inaccurate position of moving vehicles, but the knowledge of the vehicle's current position needs to be improved. Since perfect measurements cannot be guaranteed due to movement of objects, the measurements should be filtered to produce the best estimation of the accurate track. The Kalman filter can optimally estimate the current position of each vehicle and also predict the location of vehicles in future video frames by minimizing noise disorders. It is also used to stop tracking of vehicles proceeding in opposite direction in roadway captured video. Although edge detection can find moving objects, the Kalman filter makes an optimal estimate of positions based on a sequence of localization measurement.

The linear Kalman filter is simpler and used in proposed technique. Consider parameter A as area of vehicle's bounding box, which has been detected in frame differentiation phase and p(x, y) is the center point of the vehicle where x and y are its distances from horizontal and vertical edges. Now by integration of proposed parameter in (7) and (8) equations resulted in the following vectors [30]:

$$xk = [x, y, A, vx, vy, vA]^T \quad (7)$$

$$yk = [x, y, A]^T \quad (8)$$

Where v_A is the rate of changes in vehicle's bounding box, v_x and v_y are the speed of changes in the movement of vehicle's center point. Subsequently using the Kalman filtering technique, the position of each vehicle can be estimated and tracked better. Finally an identifier is allocated to each passing vehicle for counting and classification purposes.

The Kalman Filter is a unsupervised algorithm for tracking a single object in a continuous state space. Given a sequence of noisy measurements, the Kalman Filter is able to recover the "true state" of the underling object being tracked. It is implemented using the pykalman library of python.

Sample code-

```
from pykalman import KalmanFilter

kf = KalmanFilter(initial_state_mean=0, n_dim_obs=2)
```

The traditional Kalman Filter assumes that model parameters are known beforehand. The KalmanFilter class however can learn parameters using KalmanFilter.em() (fitting is optional). Then the hidden sequence of states can be predicted using KalmanFilter.smooth():

```
measurements = [[1,0], [0,0], [0,1]]

kf.em(measurements).smooth([[2,0], [2,1], [2,2]])[0]
array([[ 0.85819709],
       [ 1.77811829],
       [ 2.19537816]])
```

Common uses for the Kalman Filter include radar and sonar tracking and state estimation in robotics. This module implements two algorithms for tracking: the Kalman Filter and Kalman Smoother. In addition, model parameters which are traditionally specified by hand can also be learned by the implemented EM algorithm without any labeled training data. All three algorithms are contained in the KalmanFilter class in this module.

F. Counting and Classification Functions:

Vehicle counters are used in computing capacity, establishing structural design criteria and computing expected roadway user revenue [10]. Typically in proposed technique vehicles are classified as four common types:

- Type1: bicycles, motorcycles
- Type2: motorcars
- Type3: pickups, minibuses
- Type4: buses, trucks, trailers

It is necessary to have the width and length of each vehicle's bounding boxes in pixels to diagnose that the passing vehicles belongs to which of the mentioned types. The area of each bounding box shows that which type should be allocated for the vehicle. Each vehicle type can be shown by a special rectangle color. Type 1 has been represented by red, where Type2, Type 3 and Type 4 have been characterized by green, blue and yellow rectangles, respectively.

In counting step, four isolated counters used for each vehicle type and a total counter is needed to store the sum value of them. All counters should count just the vehicles which are passing in a specific direction. So if a vehicle stops, turns or moves in wrong direction in the detection zone, it should not be counted. In this technique, counting is according to the number of moving vehicles detected in the detection zone and classified in one of mentioned groups.

Total passed vehicles, which will be shown in yellow, help to analyze traffic flow in a period of time. Also by calculating the bounding boxes height and width in pixels, vehicle types can be distinguished and counted by related counters. Furthermore, in both counted vehicles, edges will be covered with green rectangles, which shows that they belong to Type 2 (even the green numbers inside bounding boxes confirm this result).

V. CONCLUSION

In this paper a methodology based on python language programming has been proposed. Python is very good library like numpy, matplotlib, scipy, which can help to count traffic, classify the traffic and save the time of engineer. Traffic flow is basic data for transportation planning process and its accuracy and processing within limited time frame is challenging task for transportation and highway engineer. This tools will be very useful for their field application road construction design, traffic planning point of view.

VI. REFERENCES

[1] D. Beymer, P. McLauchlan, B. Coifman, J. Malik, "A Real-time Computer Vision System for Measuring Traffic Parameters," IEEE Conference on Computer Vision and Pattern Recognition, 1997.

[2] M. Fathy, M. Y. Siyal, "An Image Detection Technique, Based on Morphological Edge Detection and Background Differencing for Real-time Traffic Analysis," Pattern Recognition Letters, Vol. 16, pp. 1321-1330, 1995.

[3] V. Kastrinaki, M. Zervakis, K. Kalaitzakis, "A Survey of Video Processing Techniques for Traffic Applications," Image and Vision Computing, Vol. 21, pp. 359-381, 2003.

[4] D. A. Forsyth, J. Ponce, "Computer Vision: A Modern Approach," Prentice Hall, 2003.

[5] T. R. Currin, "Turning Movement Counts. In Introduction to Traffic Engineering: A Manualfor Data Collection and Analysis," Stamford Wadsworth Group, pp. 13–23, 2001.

[6] W. Yao, J. Ostermann, Y. Q. Zhang, "Video Processing and Communications," Signal Processing Series, ISBN: 0-13-017547-1, Prentice Hall, 2002.

[7] P. Choudekar, S. Banerjee, M. K. Muju, "Real Time Traffic Light Control Using Image Processing," Indian Journal of Computer Science and Engineering, Vol. 2, No. 1, ISSN: 0976-5166.

[8] N. Chintalacheruvu, V. Muthukumar, "Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems," Journal of Transportation Technologies, Vol. 2, pp. 305-314, 2012.

[9] R. Milances Gil, S. Pun, T. Pun, "Comparing Features for Target Tracking in Traffic Scenes," Pattern Recogition, Vol. 29, No. 8, pp. 1285-1296, 1996.

[10] E. Atkociunas, R. Blake, A. Juozapavicius, M. Kazimianec, "Image Processing in Road Traffic Analysis," Nonlinear Analysis: Modelling and Control, Vol. 10, No. 4, pp. 315–332, 2005.

[11] X. Fu, Z. Wang, D. Liang, J. Jiang, "The Extraction of Moving Object in Real-Time Web-Based Video Sequence," 8th International Conference on Digital Object Identifier, Vol. 1, pp. 187-190, 2004.

[12] V. Khorramshahi, A. Behrad, N. K. Kanhere, "Over-Height Vehicle Detection in Low Headroom Roads Using Digital Video Processing," World Academy of Science, Engineering and Technology, 2008.

[13] J. Zhou, D. Gao, D. Zhang, "Moving Vehicle Detection for Automatic Traffic Monitoring," IEEE Transactions on Vehicular Technology, Vol. 56, NO. 1, 2007.

[14] G. Welch, G. Bishop, "An Introduction to the Kalman Filter", The University of North Carolina at Chapel Hill, 2006.

[15] P. G. Michalopoulos, "Vehicle Detection Video Through Image Processing: The Autoscope System," IEEE Transactions on Vehicular Technology, Vol. 40, No. 1, 1991.

[16] A. H. S. Lai, G. S. K. Fung, N. H. C. Yung, "Vehicle Type Classification from Visual-Based Dimension Estimation," IEEE Intelligent Transportation Systems Conference, pp. 201-206, 2001.

[17] D. G. Lowe, "Distinctive Image Features from Scaled-Invariant Keypoints," International Journal of Computer Vision, pp. 91-110, 2004.

[18] P. T. Martin, G. Dharmavaram, A. Stevanovic, "Evaluation of UDOT's Video Detection Systems: System's Performance in Various Test Conditions," Report No: UT-04.14, 2004.

[19] O. Hasegawa, T. Kanade, "Type Classification, Color Estimation, and Specific Target Detection of Moving Targets on Public Streets," Machine Vision and Applications, Vol. 16, No. 2, pp. 116-121, 2005.

[20] R. Cucchiara, M. Piccardi, P. Mello, "Image Analysis and Rule-based Reasoning for a Traffic Monitoring System," IEEE Transactions on Intelligent Transportation Systems, Vol. 1, Issue 2, pp 119-130, 2000.

[21] Q. Cai, A. Mitiche, J. K. Aggarwal, "Tracking Human Motion in an Indoor Environment," International Conference on Image Processing, USA, Vol. 1, pp. 215-218, 1995.

[22] T. Le, M. Combs, Q. Yang, "Vehicle Tracking based on Kalman Filter Algorithm," Technical Reports Published by the MSU Department of Computer Science, ID: MSU-CS-2013-02, 2013.

[23] Sh. Agarwal, A. Awan, D. Roth, "Learning to Detect Objects in Images via a Sparse, Part-based Representation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

[24] Sh. Agarwal, D. Roth, "Learning a Sparse Representation for Object Detection," European Conference on Computer Vision, Vol. 1, ISBN: 978-3-540-43748-2, pp. 113-127, 2002.

[25] A. Torralba, K. P. Murphy, W. T. Freeman, "Shared Features for Multiclass Object Detection," Toward Category-Level Object Recognition, ISBN: 978-3-540-68794-8, pp. 345-361, 2006.

[26] L. Bohang, L. Qingbing, C. Duiyong, S. Hailong, "Pattern Recognition of Vehicle Types and Reliability Analysis of Pneumatic Tube Test Data under Mixed Traffic Condition," 2nd International Asia Conference on Informatics in Control, Automation and Robotics, ISSN: 1948-3414, pp. 44-47, 2010.

[27] L. Feng, W. Liu, B. Chen, "Driving Pattern Recognition for Adaptive Hybrid Vehicle Control," SAE 2012 World Congress and Exhibition, pp. 169-179, 2012.

[28] D. Simon, "Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches," Wiley-Interscience, 2006.

[29] R. Gonzalez, R. E. Woods, "Digital Image Processing," 2nd Edition, Prentice-Hall, 2002.

[30] S. Siang Teoh, T. Bräunl, "A Reliability Point and Kalman Filterbased Vehicle Tracking Technique," International Conference on Intelligent Systems, pp. 134-138, 2012.

[31] K. Markus, "Using the Kalman Filter to Track Human Interactive Motion Modeling and Initialization of the Kalman Filter for Translational Motion," Technical Report, University of Dortmund, Germany, 1997.

[32] D. Nagamalai, E. Renault, M. Dhanuskodi. "Implementation of LabVIEW Based Intelligent System for Speed Violated Vehicle Detection", First International Conference on Digital Image Processing and Pattern Recognition, ISSN: 1865-0929, pp. 23-33, 2011.

[33] I. E. Igbinosa, "Comparison of Edge Detection Technique in Image Processing Techniques", International Journal of Information Technology and Electrical Engineering, ISSN: 2306-708X, Vol. 2, Issue 1, 2013.

[34] Learning OpenCV: Computer Vision with the OpenCV Library By Gary Bradski, Adrian Kaehler.