The International Journal of Engineering and Science (IJES)
|| Volume || 7 || Issue || 9 Ver.III || Pages || PP 64-72|| 2018 ||
ISSN (e): 2319 – 1813 ISSN (p): 23-19 – 1805

Vehicle Speed detection by using Camera and image processing software

Hakan Koyuncu[1], Baki Koyuncu[2]
[1] *Computer Engineering Dept., Istanbul Gelisim University*
[2] *Electrical and Electronics Engineering Dept., Istanbul Gelisim University*
Corresponding Author;Hakan Koyuncu

ABSTRACT: *It is an increasing trend to detect vehicle speedssimply by using a simple camera and image processing software. In the proposed technique, avehicle which is passing through the camera field of view (FOV) is detected. Two techniques named linear motion and discrete motion speed detection are employed. In the first technique, the vehicle speed was calculated as the ratio of real distance covered by the FOV and the time duration between the vehicle entering and exiting the FOV. Time duration is determined as the time between the time stamps at the beginning and at the end of the FOV.In the second technique vehicle speed is calculated at different time stamps within the FOV with respect to initial startup time stamp. In order to avoid the unwanted vehicle within the camera field, a rectangular area is generated in the middle of FOV and image processing techniques are deployed to capture only the related vehicle in this area in both techniques. Vertical distance between the camera and the vehicle and the vehicletravelling distance in the FOV are determined by using trigonometry. Accurate vehicle speeds are calculated below 50 km/hr. These speeds were compared against the speeds obtained with car speedometers and it was found that the detected speeds with the video system was close to speedometer speeds by ±1.2m/sec below 50km/hr.*

KEYWORDS*: Field of view (FOV), Camera field, travelling distance, vertical distance, trigonometry, time stamp, image processing*

Date of Submission: 07-09-2018 Date of acceptance: 24-09-2018

I. INTRODUCTION

Vehicle speed monitoring is one of the most important enforcements of the traffic laws, [1,2]. With the increasing urban life in the cities, the number of people and the number of vehicles are increased drastically. As a result, over speeding became the major reason for accidents. Hence controlling the speeds of vehicles has become very important issue for traffic management. During last decades, Doppler radar was employed to measure the speed of moving vehicles [3,4]. It was a hand held device which sent a radio beam to a moving vehicle and then calculated the vehicle speed by measuring the change in reflected wave frequency. It was a reliable device as long as there was no other vehicle in the field of view. See Figure 1.

Figure 1: hand held Radar Gun

There were some difficulties with this radar system. Firstly, Cosine error, [5], must be taken into account if the radar gun is not in line of sight. Radio interference must also be considered due to error caused in speed detection. Shadowing which is the radar wave reflection from 2 vehicles with different heights also causes speed detection errors.

Previously, video processing and image processing techniques were deployed for vehicle speed detection[6,7].These speed measurements are based on image frame difference, calibrated cameras and optical and digital video images.

In proposed study, any moving vehicle inside a video from any video camera or mobile source is utilized. The algorithms are implemented in C++using openCV and visual studio. Initial system is developed with a laptop and webcam. The aim was to carry the developed software to a mobile platform such as a smart phone or to a simple computer such as Raspberry Pi to generate a real time mobile vehicle speed detector.

II. METHODOLOGY

In this study,a mobilevehicle is considered. A video camera and side view images of the vehicle taken with it are deployed [8].

It was assumed that the viewed image scene is flat. Perspective distortions on the acquired images are considered negligible. Furthermore, since only one camera is used, the velocity vectors can only be formed in 2 dimensions. Hence the scale of the images along the 2D velocity vectors should be defined in a precise way. The real distance on the road covered by the FOV, [9], along the horizontal velocity vector must be measured precisely. Furthermore, the length of a line between 2 points on the road parallel to horizontal velocity vector is also measured by a hand held laser distance meter (Bosch PLR 50), [10], withinan accuracy of ±0.1 millimeter. The camera is set up so that the direction of the vehicle movement is from left-to-right or right-to-left parallel to camera plane within the camera FOV. The camera receives the side view of the vehicle.

The camera deployed in this study has a frame rate of 30 frames per second (fps) and an effective area of 640x480 pixels, [11]. A frame is captured at every 33.3 milliseconds. This means the speed calculations must be carried out within this time limit. The pixel size of the camera is 9 microns. The focal length of the camera is 6 mm.

A video signal describes a sequence of time varying images, [12]. A still image is a spatial distribution of intensities remaining constant in time. On the other hand, a time varying image has a spatial intensity distribution varying in time.

Videos can be in different formats based on the camera types. The video format used in this study is in AVI. Since the video had 30 fps, extracting all the frames would cause unwanted redundancy and this would increase the delay time to execute the program. Hence, frame sampling of 2 frames per second was chosen for the computations. Camera used in the study was a color camera. Colored images are converted into gray scale images. This reduced the amount of computations.

The field of view (FOV) of the camera used in this study is termed as '2α' degrees. The camera sees a horizontal distance 'D' within FOV on the road parallel to the camera. See Figure 2. The vertical distance from the camera lens to the distance D on the road is taken as 'L'. Known parameters in Figure 2 geometry are the camera distance L from the road and the FOV angle of 2α.

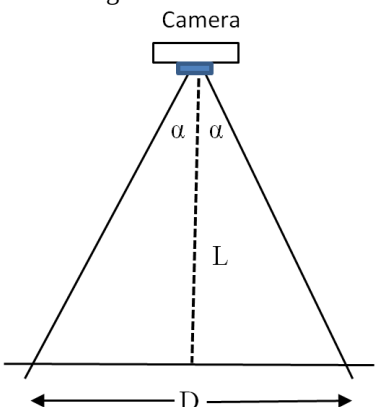

Figure 2: Camera FOV geometry

Hence the real distance D on the road is

$D = 2*L*\tan(\alpha)$

If the camera FOV is 50 degrees and if the vertical distance L between the road and the camera lens is 34 meters than the FOV real distance D on the road seen by the camera is 32 meters. Hence the real distance seen by the camera across the road and the vertical distance between the camera and the road are roughly equal.This is taken as a thumb rule while arranging the camera and the distances during the speed measurements.

1. Image Processing

Once the camera is positioned with respect to road, image frames of the vehicles are captured by the camera at 30 frames per second. Several image processing tasks are carried out to obtain the speed information.

a) Noise Reduction

Initially, Salt and pepper noise, [13], is generated in images when it is transmitted over noisy channels as in video cameras. The images can also be degraded due to electrical sensor noise. Secondly the convolution noise (blurring) can appear due to misfocus of the camera lens, camera motion and atmospheric conditions. All these noise sources increase the contributions to high frequency noise components. Hence median filtering is employed to reduce this high frequency noise. This way object edge information is saved for the detection algorithm. Edges in the image are important features and they stay significant among the traffic scenes and other lightening conditions.

b) Background Subtraction

The image obtained by the camera must be calibrated so that the object image must be brought forward for processing. A reference image frame is required to subtract from the current image frame. This way background of the image is subtracted and the region of interest of the object image becomes prominently visible.

c) Edge Detection

Once the background subtraction is carried out, edges of the object image is detected. This detection is deployed in several steps.

Initially, a smoothing operation is made to blur the image and reduce the noise further.

Secondly, the edges are marked where the gradient of the images has large amplitudes.

Thirdly edges are determined by using CANNY edge detection algorithm, [14].

Final edges are determined by suppressing all the other edges which are not connected to a strong edge.

d) Enhancement

Although the edges of the images are determined gaps are observed along the edges. To provide the continuity and large features of interest, morphological operators dilation and erosion are employed. These operators remove the specific image features smaller than the main structure of the interest without effecting the larger features.

e) Masking

Due to heavy background interference, simple masking techniques are applied to reduce these effects. Correlation between pixels in the neighborhood of the object and thresholding, [15], areapplied before the edge detection to improve the image quality. Similar masking is also deployed to remove the multiple objects on different lanes.

f) Speed Calculation

Once the vehicle edges are clearly displayed on the image, speed algorithm is applied to determine the vehicle speed.

A rectangular test area in the FOV is chosen and drawn on the image screen. Figure 3.

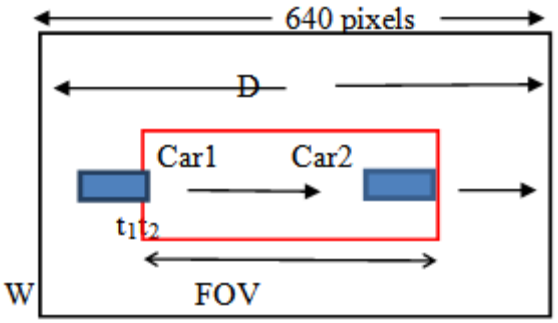

Figure 3. FOV image area

As the front end of the vehicle enters the left boundary of the test area, the front edge of the car is detected. When the vehicle leaves the right boundary of the test area the front edge of the vehicle is again detected. The number of pixels are determined between two edges. This number is multiplied with the pixel calibration factor (D meters / 640 pixels) to determine the distance traveled by the vehicle. Additionally, $\mathbf{T_1}$and$\mathbf{T_2}$ time stamps are also determined from the system CLK frequency. Finally, the distance found is divided by$\mathbf{(T_2\text{-}T_1)}$ time difference to calculate the speed of the vehicle going from left to right in kilometer/hour.

2. Software

In this study, a laptop webcam is employed to capture the images of the moving vehicles. The software captures an image from the webcam. The continuous input of an image stream forms a video stream. The software converts the color video stream to gray scale in order to increase the computation speed.

Two different techniques were deployed to determine the speed of the tracked vehicle. These are linear motion and discrete motion techniques.

A. Linear Motion Detection

In this technique, we track the motion of a vehicleat constant speed. Motion tracking is conducted inside an area chosen by the user across the FOV. This is done to eliminate undesired objects from being tracked and to increase the computation speed.

Figure 4 gives the flow chart of the algorithm which is used to determine the vehicle speed.

Video frames are capturedin a WHILE LOOP where the rectangular box is drawn and other calculations are carried out. Gaussianfilter, [16], as shown in the below code segment is applied on the video stream to smooth out the imperfections such as noisy pixels which can cause errors in the detection of edges.

```
blur(input,output,Size(15,15))
```

After pre-setting the video stream by converting to gray scale and blurring by Gaussian filtering, a rectangular image test area is allowed to be drawn by the user as shown in below segment:

```
staticvoid onMouse(intevent, intx, inty, int, void*)
{
        if (selectObject)
        {
                selection.x = MIN(x, origin.x);
                selection.y = MIN(y, origin.y);
selection.width = std::abs(x - origin.x);
selection.height = std::abs(y - origin.y);
selection &=Rect(0, 0, image.cols, image.rows);
        }
switch (event)
        {
        caseCV_EVENT_LBUTTONDOWN:
                origin =Point(x, y);
                selection =Rect(x, y, 0, 0);
                selectObject = true;
                break;
        caseCV_EVENT_LBUTTONUP:
                selectObject = false;
if (selection.width > 0 && selection.height > 0)
                        trackObject = -1;
                break;
        }
}
```

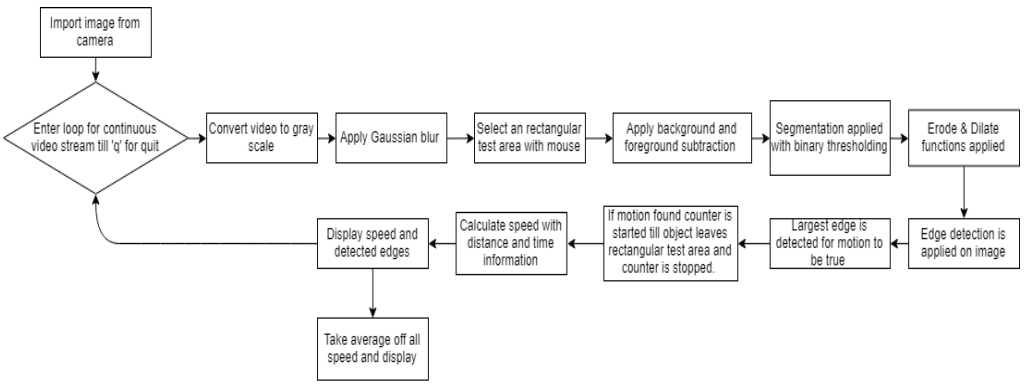

Figure 4: System Software Flow Chart

This image test area is drawn to simplify the calculations and reduce the computational load hence yielding faster results.
In this selected image area,image subtraction is employed between static background image and new foreground image to distinguish the vehicle using the below segment:

```
Ptr<BackgroundSubtractor> pMOG2;
pMOG2 = createBackgroundSubtractorMOG2();
pMOG2->apply(crop, fgMaskMOG2)
```

Furthermore,thresholding is applied on the image for segmentation purposes by the following threshold function;

```
threshold(fgMaskMOG2, dst, 120, 255, THRESH_BINARY);
```

Erodefunction, [17],is used to eliminate any other noisy pixels from the video shown below:

```
erode(dst, imgfinal, Mat(), Point(-1, -1), 2, 1, 1);
```

Once the undesired pixels have been eliminated, other desired pixels are expanded with the help of **dilation** function, [18] shown below.

```
dilate(imgfinal, imgfinal, Mat(), Point(-1, -1), 2, 1, 1);
```

Edge detection is applied with **findcontour**(canny) function seen below. This allows for all the edges to be identified in a given scene.

```
findContours(imgfinal, contours, hierarchy, CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);
for (int i = 0; i < contours.size(); i++)
{
        double a = contourArea(contours[i], false);
if ((a > largest_area) && (a > min_area)) {
        largest_area = a;
        largest_contour_index = i;
        bounding_rect = boundingRect(contours[i]);
        motiondetect = 1;
}
}
```

In this code segment, the largest edge is also identified and motion is accepted to be triggered. As the vehicle enters from the vertical left side of test area, the motion is sensed and $\mathbf{T_0}$ initial time stamp is recorded. $\mathbf{T_1}$ time stamp is triggered and recorded when the vehicle reaches the vertical right side of the rectangular test area. $\mathbf{T_0}$ and $\mathbf{T_1}$ are recorded as the number of clock periods of the computer used. $\Delta\mathbf{T}= \mathbf{T_1}\text{-}\mathbf{T_0}$is the time stamp difference of clock periods**.** This difference is multiplied by the clock frequency of the computer and the time difference in seconds is obtained.
Itis the total time in seconds that thevehicle takesto cross the rectangular test area in horizontal direction.

```
if (motiondetect == 1)
{
if (state == 0)
        {
        state = 1;
        t0 = getTickCount();
}
elseif (t0 != 0 && motiondetect == 0)
        {
        t1 = getTickCount();
}
}
double secdiff(int64a, int64b)
{
        double totalsec;
        totalsec= (a - b) / getTickFrequency();
        return totalsec;
}
```

The real distance D in meters on the road represents the horizontal frame distance of FOV which is 640 pixels. Hence the pixel calibration factor (PCF) is
(PCF) = D / (640) meter/pixel
Once the calibration factor is known, the horizontal lengthW of the rectangular
horizontal test area can be calculated by using
W= (PCF)* S meters
where S is the number of pixels between two $\mathbf{T_0}$ and $\mathbf{T_1}$ time stamps. This can also be defined as the number of pixels between the two vertical edges of the rectangular test area along the horizontal direction.S is determined by using a special function

staticvoid onMouse(intevent, intx, inty, int, void*)

This function provides the coordinates of 4 corners of rectangular test area in pixel form. An example rectangular area and pixel coordinates of its cornersare given in Figure 5.

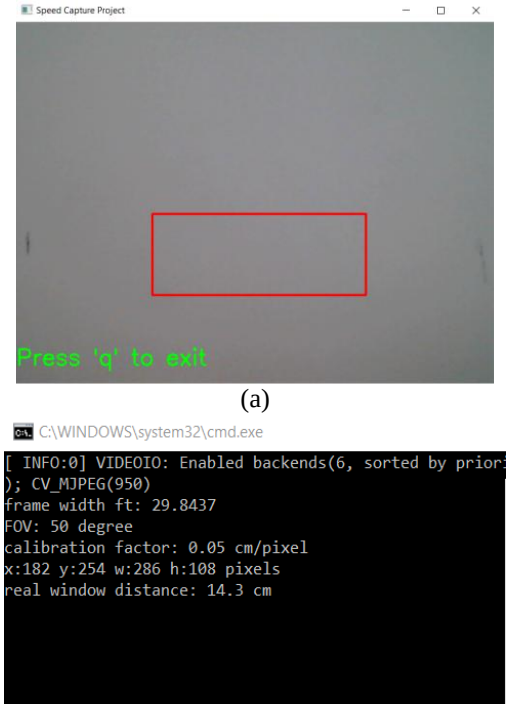

Figure 5: a) rectangular area b) pixel (x,y) top left point coordinates , w is the pixel distance of S

Once the horizontal pixel distance of S is known; W distance in meters can be calculated by the previous formula.
By knowing the horizontal distance in meters and the time in seconds the vehicle traveledin this distance, the speed can be calculated by using the universal law of speed = distance / time.
The calculated speed information is printed on the video stream indicating how fast the vehicle is travelling on the road.

B. Discrete Motion Detection

This detection technique has the same pre-setting process as in linear motion detection with the key difference of time calculations. Similarly, after the pre-setting phase a rectangular image test area is drawn by the user to reduce the computationalload.

In the selected test area, foreground subtraction from background is applied in order to distinguish the relatedvehicle insidethe static background. **Thresholding,Erosion,** and **Dilation** are applied on the image to eliminate noisy pixels and enhance the clear ones. Edge detection is applied with **findcontour** function. The largest edge is identified and motion is accepted to be triggered.

As soon as motion is detected an initial time stamp is taken. The progress of the vehicle passing through the rectangular test area is tracked with a continuous WHILE loop. On every iteration of the loop, the vehicle is checked if it is still in the test area. If it is still in, an edge is detected and the time stamp is taken. The vehicle can move from right-to-left or left-to-right.

When the vehicle's largest horizontal edge is detected, a bounding box is arranged around this edge. If it is moving left-to-right in horizontal direction, the right side of the bounding box is used for detections. The position of the bounding box is computed by using width, height and top left corner coordinates of the bounding box. Hence, location, width and height information are known in each frame at all times. Since the vehicle movement is along the x axis, only the xcoordinates will be considered for the calculations. As soon as the vehicle partially entered in the test area a bounding box is formed around this partial section. The x pixel distance between the right side of bounding box and the left side of the rectangular test area is calculated. Code segment for this calculation is shown below.

```
if (bounding_rect.x >= last_x)
{
abs_chg = bounding_rect.x + bounding_rect.width - initial_x;
}
else{
abs_chg = initial_x - bounding_rect.x;
}
```

The top left corner of the bounding box has known coordinates of (x,y) as it enters into the test area. In order to draw the bounding box, width and height information arealso needed. The right side of the bounding box will enter the rectangular area first in a left-to-right motion. Hence, x+width will give the right most corner of the bounding box.

Last_x coordinate value in the above code is the previous position of bounding_rect.x. As the object moves from left-to-right, the bounding box's, x coordinate value increases. Bounding box is identified asbounding_rect in the code. If this is not the case, then the object is moving right-to-left.

Once the direction of the bounding box motion is determined, it is possible to calculate the distance which the boundary box, consequently the vehicle, has traveled after entering the rectangular test area.

When moving left to right, the vehicle hence the bounding box right most point is needed for calculations. On the other hand, moving right to left only requires the far left corner of the vehicle. Calculations are carried out similar to left to right motion.

Time stamps are taken at each iteration of the WHILE LOOP with respect to initial start time stamp. The difference between the initial time stamp and the other discrete time stamps are taken as the time travelled by the vehicle. Once the distances and the time differences are determined, the speed is easily calculated at each time stamp. These calculated speed values are averaged out and a single speed value is defined for one vehicle pass in front of the camera.

III. RESULTS AND CONCLUSİONS

In this study, speeds of vehicles on urban roads are detected by using video cameras. Two measurement techniques are employed to determine the speeds of vehicles. First technique has employed simple detection of vehicles by entering and exiting a rectangular test area in camera FOV. As the vehicle entered the test area entrance time stamp is recorded. When the vehicle exited the test area exit time stamp is again recorded. The time difference between them is used to calculate the vehicle speed.

In the second technique, time stamps are determined at each loop iteration of the program. The vehicle being tracked can have different speed readings at each iteration. The time difference between these time stamps and initial time stamp are used in speed calculations. Each time stamp is described as a discrete time and the distance of the vehicle from this time stamp to the initial time stamp is also determined in pixel form as the vehicle distance. Once the pixel distance is calibrated and converted into real distance on the road, adiscrete speed calculation is carried out at each time stamp across the test area. Since the vehicle has a linear motion across the road, average of these speeds gave an average speed value for the vehicle.

Speed measurements of two techniques are checked out with a car speedometer. A Hyundai i20 car is employed to check the speed measurements with the developed video system. Table 1 and Table 2 are given for both techniques. Absolute speed differences are compared in these tables. It was found that the video system has a speed detection accuracy of ±1.2 km/hrup to 50 km/hr. After 50km/hr the video system accuracy starts to degrade and speed error margins increases. See Figure 6. Both techniques show the similar performances and they both determine the speeds of vehicles approximately the same.

The developed system will be a very useful system to measure the low speeds accurately and without using expensive instruments. Eventually a smart phone application will be developed and low speeds up to 50 km/hr speeds can be measured easily and efficiently.

REFERENCES

[1]. R. K. Kodali and M. Sairam, "Over speed monitoring system," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 2016, pp. 752-757.doi: 10.1109/IC3I.2016.7918061

[2]. Hafeez, Farrukh& Al Shammrani, Mohammad & Al Shammary, Omar. (2007). Smart Vehicles Speed Monitoring System Using RFID. 3297. 10.15662/ijareeie.2015.0404002.

[3]. Doviak, Richard &Zrnic, D.S. & S. Sirmans, Dale. (1979). Doppler Weather Radar. Proceedings of the IEEE. 67. 1522 - 1553. 10.1109/PROC.1979.11511.

[4]. Eryildirim, Abdulkadir&Onaran, Ibrahim. (2011). Pulse Doppler Radar Target Recognition using a Two-Stage SVM Procedure. Aerospace and Electronic Systems, IEEE Transactions on. 47. 1450 - 1457. 10.1109/TAES.2011.5751269.

[5]. M. E. Goodson, "Technical Shortcomings of Doppler Traffic Radar," Journal of Forensic Sciences, JFSCA, Vol. 30, No. 4, Oct. 1985, pp. 1186-1193.

[6]. K. V. K. Kumar, P. Chandrakant, S. Kumar and K. J. Kushal, "Vehicle Speed Detection Using Corner Detection," 2014 Fifth International Conference on Signal and Image Processing, Bangalore, India, 2014, pp. 253-258. doi: 10.1109/ICSIP.2014.46

[7]. Ibrahim, Osman &ElGendy, Hazem&Elshafee, Ahmed. (2011). Speed Detection Camera System using Image Processing Techniques on Video Streams. International Journal of Computer and Electrical Engineering. 3. 771. 10.7763/IJCEE.

[8]. Doğan S, Temiz MS, Külür S. Real Time Speed Estimation of Moving Vehicles from Side View Images from an Uncalibrated Video Camera. Sensors (Basel, Switzerland).2010; 10(5):4805-4824. doi:10.3390/s100504805.

[9]. P. Pretto, M. Ogier, H.H. Bülthoff, J.-P. Bresciani, Influence of the size of the field of view on motion perception, Computers & Graphics, Volume 33, Issue 2, 2009, Pages 139-146, ISSN 0097-8493, https://doi.org/10.1016/j.cag.2009.01.003.

[10]. Kaškonas, Paulius&Meškuotienė, Asta. (2012). Vehicle Speed Meters Validation and Verification System. Electronics and Electrical Engineering. 119. 10.5755/j01.eee.119.3.1372.

[11]. Handa A., Newcombe R.A., Angeli A., Davison A.J. (2012) Real-Time Camera Tracking: When is High Frame-Rate Best?. In: Fitzgibbon A., Lazebnik S., Perona P., Sato Y., Schmid C. (eds) Computer Vision – ECCV 2012. ECCV 2012. Lecture Notes in Computer Science, vol 7578. Springer, Berlin, Heidelberg

[12]. Mahboubi, A., Benois-Pineau, J. & Barba, D. EURASIP J. Adv. Signal Process. (2002) 2002: 750583. https://doi.org/10.1155/S1110865702000902

[13]. S. Deivalakshmi and P. Palanisamy, "Improved tolerance based selective arithmetic mean filter for detection and removal of impulse noise," 2010 5th International Conference on Industrial and Information Systems, Mangalore, 2010, pp. 309-313, doi: 10.1109/ICIINFS.2010.5578687

[14]. Eshaghzadeh, Ata & Kalantari, roghayehsadat. (2017). Canny Edge Detection Algorithm Application for Analysis of the Potential Field Map. Earth Science India. 10. 0974-8350.

[15]. L. Fang, Y. Zou, F. Dong, S. Sun and B. Lei, "Image thresholding based on maximum mutual information," 2014 7th International Congress on Image and Signal Processing, Dalian, 2014, pp. 403-409, doi: 10.1109/CISP.2014.7003814

[16]. S. K. Kopparapu and M. Satish, "Identifying Optimal Gaussian Filter for Gaussian Noise Removal," 2011 Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, Hubli, Karnataka, 2011, pp. 126-129, doi: 10.1109/NCVPRIPG.2011.34

[17]. R. van den Boomgaard and A. Smeulders, "The morphological structure of images: the differential equations of morphological scale-space," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 11, pp. 1101-1113, Nov. 1994, doi: 10.1109/34.334389

[18]. X. Lin and J. Chen, "Research and application of Mathematical Morphology algorithms on OSSC," 2009 IEEE International Workshop on Open-source Software for Scientific Computation (OSSC), Guiyang, 2009, pp. 172-178, doi: 10.1109/OSSC.2009.5416872

Linear Motion Detection (LMD)				
Exp. #	Vehicle Direction	Speedometer km/hr	Estimated Speed km/hr	Absolute Difference 1
1	L-R	5	5.8	0.8
2	R-L	10	11.4	1.4
3	L-R	15	13.7	1.3
4	R-L	20	19.2	0.8
5	L-R	25	26.3	1.3
6	R-L	30	31.5	1.5
7	L-R	35	33.8	1.2
8	R-L	40	39.2	0.8
9	L-R	45	46.2	1.2
10	R-L	50	51.4	1.4
11	L-R	55	60.6	5.6
12	R-L	60	66.5	6.5
Average absolute difference till 50 km/hr				1.17km/hr

Table 1. A sample of speed measurements with Linear Motion Detection

Average Discrete Motion Detection (DMD)				
Exp. #	**Vehicle Direction**	**Speedometer km/hr**	**Estimated Speed km/hr**	**Absolute Difference 2**
1	L-R	5	5.6	0.6
2	R-L	10	11.3	1.3
3	L-R	15	13.4	1.6
4	R-L	20	18.6	1.4
5	L-R	25	26.3	1.3
6	R-L	30	31.3	1.3
7	L-R	35	33.5	1.5
8	R-L	40	38.6	1.4
9	L-R	45	46.4	1.4
10	R-L	50	51.4	1.4
11	L-R	55	61.0	6.0
12	R-L	60	67.0	7.0
Average absolute difference till 50 km/hr				1.32km/hr

Table 2. A sample of speed measurements with Discrete Motion Detection

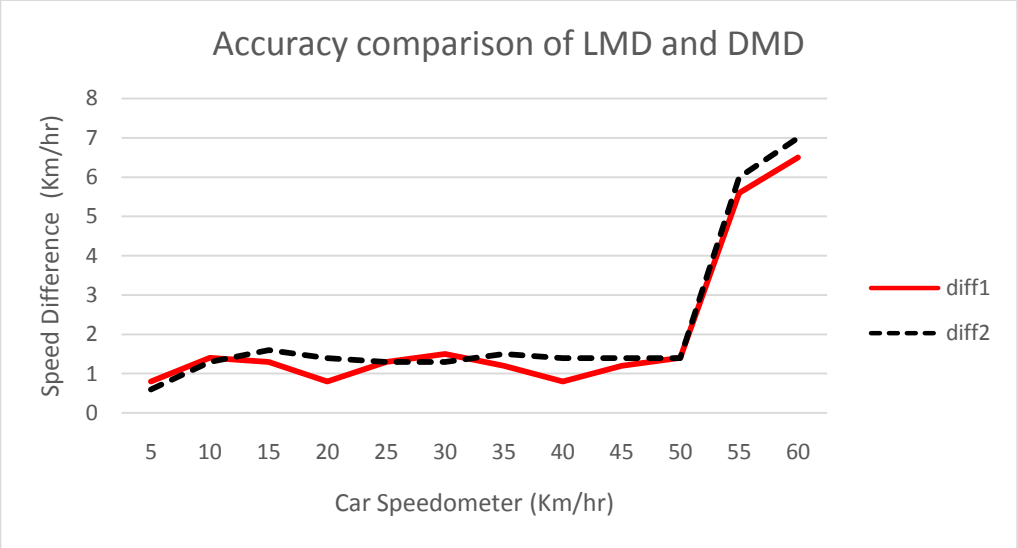

Figure 6: The accuracy graphs Diff 1 and Diff 2 for different speedometer speeds