ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 13, March 2018

Vehicle Speed Measurement In Video Sequence Using SIFT And KLT Methods

Dr.P.Vasuki[1]
Professor,ECE department
K. L. N. College of Information Technology
vasakime@gmail.com[1]

M.Renugadevi[4]
ECE department
K. L. N. College of Information Technology
srivini.44@gmail.com[4]

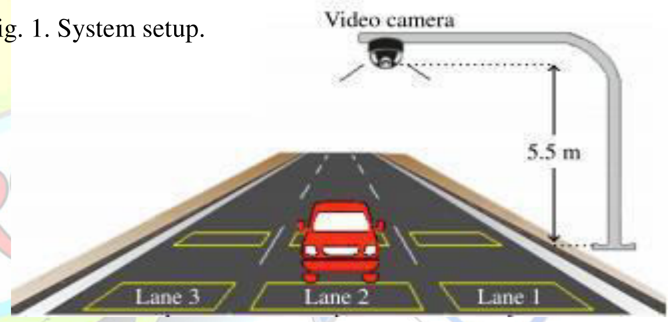
K.S.Poornimadevi[2]
ECE department
K .L. N. College of Information Technology
ksdevi008@gmail.com[2]

K.Vidya[4]
ECE department
K. L. N. College of Information Technology
ntkvidya@gmail.com[4]

***Abstract*—Speed Measurement of vehicle has been identified as a key risk factor in road traffic injuries and influencing the risk of a road crash. We need an efficient method to identify the speed of vehicle and detection of License plate. In this paper we proposed method consist of three steps, to identify the speed of vehicle. First step, the motion detection used MHI algorithm and blob analysis. In second step, feature of license plate region has been extracted using SIFT and KLT. In third step, the vehicle speed is determined by calculating centroid point difference between two frames divide by time. The proposed method gives the average accuracy of 93.8%. The proposed method is outperformed than existing method by increasing 0.8%.**

***Keywords*—Motion detection, License Plate detection, Motion Prediction and tracking, Speed Measurement.**

Fig. 1. System setup.



ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 13, March 2018

I. INTRODUCTION

In the contemporary world, we need for intelligent traffic management systems due to the increasing rate of traffic. According to the Global Status Report on road safety 2015 by the WHO, the speed limit on a section of road takes account of safety, mobility and environmental considerations.
Indeed, the WHO proposes to increase emphasis on enforcement of speed limits in most countries for successfully developing safer driving behavior. Video processing systems require a stream processing architecture, in which video frames
[2] from a continuous stream are processed one or at a time. Video is the technology of electronically capturing, recording, processing, storing, transmitting, and reconstructing a sequence of still images representing scenes in motion. Moving object detection as shown in fig.2. is the basic step for further analysis of video. Video sensors play an important role in traffic applications due to their fast response, ease of installation, operation and maintenance in large areas. The approach [2] of using video image processing in vehicle speed detection was introduced to replace the use of RADAR and LIDAR, MLD, LASER, etc. Magnetic Loop Detector(MLD) are used to count the number of vehicles using magnetic properties. Inductive Loop Detectors provide a cost effective solution. However, they are subject to a failure rate, when installed in poor road surfaces, leading to obstruct traffic during maintenance and repair. Infrared Sensors are affected to a greater degree by fog that cannot be used for traffic surveillances. Radar technology has
made its way into production-mode vehicles. It depends upon the frequency

Fig. 2 input image captured by our system.

modulation technique. Its reflections are received and demodulated, the frequency content is analyzed. The frequency shift in the received signal is used to measure the distance to the detected object. Detected objects are then tracked and filtered based on motion characteristics to identify vehicles and other obstacles. These measurements are quite noisy, requiring extensive filtering and cleaning.

II. RELATED WORK

Several video based approaches were proposed for estimating or measuring the speed of vehicles in roadways.Most methods include a background/foreground segmentation step to detect image regions containing motion. Common approaches for this task include simple frame differences [7]– [10], as well as statistic models based on medians [11]–[13], Gaussian distributions [14] or other measures [15]. Vehicle speeds are estimated by tracking image features or regions, including blobs [12], image patches [14], edges [7], [11], corners [8], [9], the license plate region [16]–[17], or a combination of such features [13]. Rectification for perspective distortion is also a step found in most methods, and may occur before or after feature tracking. Although the cited methods have some steps in common, they also have fundamental differences, not only on the way they measure vehicle speeds, but also on the type of scenario they can be used. Methods based on direct blob analysis [7], [8], [10], [12], [14], [15] are sensitive to conditions such as shadows, perspective, and illumination variations.

Moreover, these methods produce satisfactory results only when the camera is positioned high above the roadway, with the blobs being tracked for many frames. The same issues affect methods which use other types of features, but still compute them from blobs, such as those proposed by Zhiwei et al. [11], which detects edges near the limits of each blob, or Palaio et al. [13], which extracts from each blob features such as derivatives, Laplacian and color. As discussed in Section VII, we have compared our system with a blob tracking approach based on a particle filter, similar in concept to the one proposed by Maduro et al. [12]. The method from Dogan et al. [9] avoids the problems associated with blob analysis by directly tracking distinctive features using the Lucas–Kanade optical flow algorithm [4]. However, their method assumes that all the tracked features belong to the same vehicle — thus it can handle only a single vehicle at a time. Moreover, they do not take perspective into account, and require a side view of the vehicles. The work from Garibotto et al. [16] relies on how characters detected by an optical character recognition (OCR) algorithm vary in size and position. Their method demands a very robust OCR, and did not produce acceptable results even in a controlled environment — average errors for a single-camera setup ranged from 3% to 13%. A similar issue was observed in the work from Czajewski and Iwanowski [17], which is also based on license plate recognition. Note that, although our system has a license plate detection step, it does not require the characters to be precisely segmented or recognized.

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 13, March 2018

III. PROPOSED METHOD

Vision-based vehicle detection uses one or more cameras as the primary sensor suite. Unlike LIDAR and radar, cameras do not emit electromagnetic energy but rather measure the ambient light in the scene. This system has the following steps as shown in the fig. 3 in which motion detection is the first step, there by obtaining the Region of Interest. The next step is license plate detection using edge extraction , edge filtering , region grouping and region classification. We have to predict (SIFT) [20] the motion vectors in the consecutive image frames to select the features in that particular frame. The features of the vehicle is tracked by using the Kanade-Lucas-Tomasi (KLT) algorithm [20]. The speed is calculated by finding the centroid of the two successive frames in the image region with respect to time.

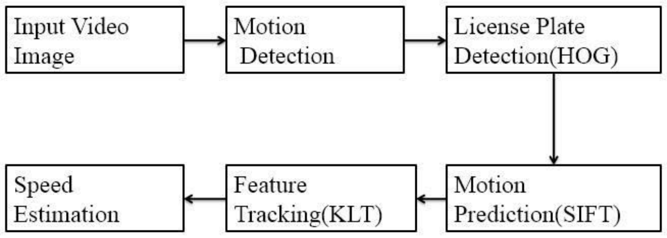


Fig. 3. Pipeline of proposed system.

IV. MOTION DETECTION

The Initial step in our project is to detect the moving vehicles, resulting in a set of region of interest. Each region of interest must contain the entire license plate from a single vehicle. The motion region was detected by using MHI algorithm.

The Motion History Image (MHI) is a static image template which helps in understanding the motion location and path propagates. In MHI, the temporal motion information is collapsed into a single image template where intensity is a function regency of motion.

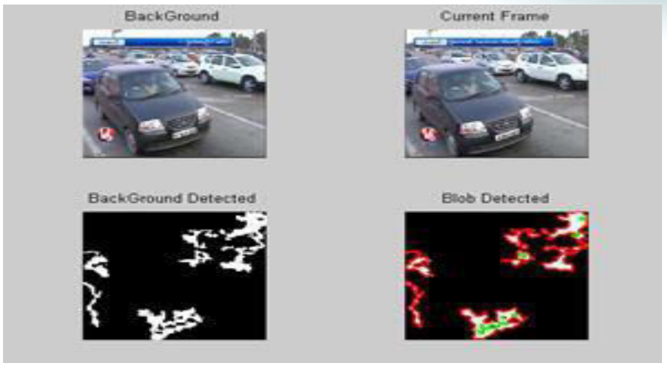


Fig. 4. foreground and background segmentation.

Thus the MHI pixel intensity is a function of the motion history, where brighter values correspond to a more recent motion. Using MHI, moving parts of a video sequence can be engraved with a single image, from where one can predict the motion flow as well as the moving parts of the video action. Some important features of the MHI representation are: It represents motion sequence in a compact manner. In this case, the silhouette (the dark shape [13] and outline of someone or something visible in restricted light against a brighter background) sequence is condensed into a grayscale image, where dominant motion information is preserved. MHI can be created and implemented in low illumination conditions where the structure cannot be easily detected otherwise. The MHI representation is not so sensitive to silhouette noises, holes, shadows, and missing parts.

The gray-scale MHI is sensitive to the direction of motion because it can demonstrate the flow direction of the motion. It keeps a history of temporal changes at each pixel location, which then decays over time. The MHI expresses the motion flow or sequence by using the intensity of every pixel in a temporal manner.

MHI is extracted from the input image that express the degree of change of motion, the brightness of the image becomes bright from dark and it is proportional to time for motion by expressing MHI where the action occur, we can calculate how much time by which direction. It is possible to obtain the temporal and spatial information whether the motion is occurred.

The motion detection gives a region of interest (ROI).The region of interest are limited based on a foreground image [20] mask as shown in fig. 4. and a vertical projection profile. The MHI (H) for time t is given by,

H(*x, y, t*) ={max(0, (, , − 1) − 1) *h* ----(1) (, ,) = 1

Where H means MHI, D means binary images obtained from the threshold frame difference and parameter represents the duration of the expected motion in frame units. In our tests, we used $\tau = 10$.

The binary segmentation mask (M) is obtained by

M(*x, y, t*) = { (, ,) > 0 0 *h* ------------ (2)

To reduce the processing time the generated image M and in subsequent steps, the images are sub sampled i.e. the values of *x* and *y* in (1) are restricted to a regular sparse grid, with pixels outside the grid being skipped. The segmentation may become less precise, but this is acceptable as long as the vehicle license plate remains entirely within its corresponding region. In tests involving the complete system, we observed that a sub sampling factor of 4 in both axes (i.e. processing 1 of each 16 pixels) reduced the processing time of the original time, without any loss in detection performance. After, the sub sampled binary segmentation M is obtained.

A. *Blob Detection:*

Blob detection [1] refers to modules that are aimed at detecting points or regions in the image that differ in properties like brightness or color compared to the surrounding, it provide complementary information about regions, it is usually done after

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 13, March 2018

color detection and noise removal to find the required object from the image. In general, Blobs [7], [8], [9], [10] are bright on dark or dark on bright regions in an image.

To implement this algorithm, a wide range of threshold values are considered to detect blobs, which are containing the Region of Interest of the frame. It depends on the threshold value starts from 10. Each blob has some key specifications like the height, the width, coordinates of center position, and the threshold value as shown in fig. 4 used for extraction.

The vertical and horizontal distance between the center positions of two neighboring characters must be minimized. Iftwo neighboring blobs with the center distance d are within the permitted range, then they can be considered as correct candidates for part of the numeric characters. The blob extraction module uses most of the computational time, because the threshold values are changed repetitively.

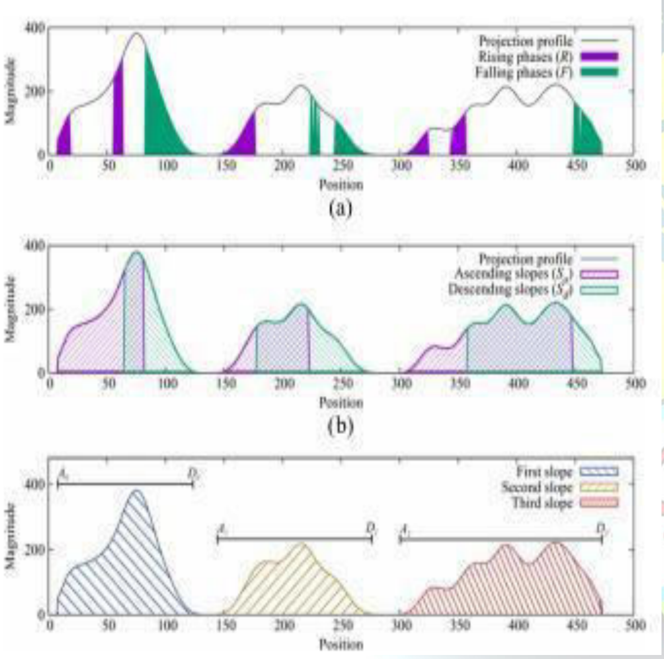
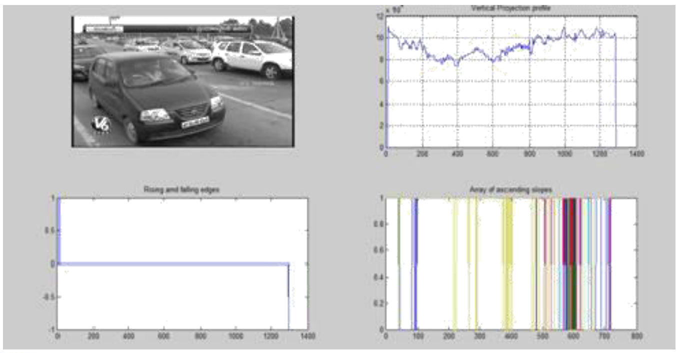


Fig. 5. Internal steps of the FIND-HILLS routine. (a) Rising and falling phases according to a given threshold . (b) Ascending and descending slopes. (c) Three slope regions delimited by the rising edge of ascending slopes and the falling edge of descending slopes.

Vertical projection profile analysis is performed after sub sampling the binary images [20], to separate vehicles horizontally. Vertical projection profile is used to perform word segmentation as the valleys are created corresponding to word gaps. These word boundaries can be identified with the help of minimum points. The novel method based on decision tree construction and on the use of projection profile histogram has been used. Projection profile is a data structure used to store the number of non-background pixel when the image is projected over normal X-Y axis.

The vertical projection profile is divided into two region. The upper part is ascending region and lower part is descending region. We take the lower part of image M, which shows the region closer to the camera generating a histogram with n image columns.



(a)

(b)

Fig. 6(a). Motion detection: regions of interest are delimited based on a foreground image mask and a vertical projection profile. 6(b).Region of interest.

We take the lower part of image M, which shows the region closer to the camera, and count the foreground pixels in each column, generating a histogram with *n* bins (for *n* image columns). This histogram is smoothed, to reduce noise, and interpreted as an array Ψ. It can be seen that the interval containing a vehicle is delimited by an ascending and a descending slope, corresponding respectively to the left and right boundaries of the vehicle.

The FIND-HILLS [20] routine is used to determine these boundaries for each vehicle as shown in fig. 5. It receives as parameters the projection profile array Ψ, and a threshold ρ (0.1, in our tests) that defines the minimum angle of inclination for a boundary. It returns a pair of lists *{A,D}*, such that each list element represents a hill's ascending and descending border, respectively. In step of FIND-HILLS, we call the FIND-INCLINATION routine, outlined. The purpose of this routine is to determine the rising and falling phases of Ψ , given as arrays *R* and *F*, respectively. The threshold $0 \leq \rho \leq 1$ is used to discard false phases: since vehicle regions are represented by high values in the projection profile as shown

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 13, March 2018

in the fig. 6. , it prevents against incorrectly dividing a vehicle in two regions. By pairing the ascending and descending boundaries produced by FIND-HILLS in arrays *A* and *D*, we can determine the left and right boundaries of each region of interest ,to detect the region of interest in order to find the license plate region of vehicle.

V. LICENSE PLATE DETECTION

The motion detector produces one region of interest for each moving vehicle present in the scene at a given time. The license plate detector finds, for each region of interest, an axis-aligned rectangle, which is an approximate bounding box of the vehicle's license plate region. Our detector follows the *hypothesis generation and validation* [16]. In the hypothesis generation phase, we use *edge extraction*, *edge filtering* and *region grouping* modules as shown in the fig. 7(b) to provide coarse candidate regions based on the edge attribute that makes up the license plate. At this phase, we aim to isolate the license plate region and prevent false negatives, even at the cost of several false positives. In the *hypothesis validation* phase, we use a *region classification* module as shown in the fig. 7(g) to refine the candidates. For this classification we use the *Text HOG* (T-HOG) descriptor [17], which is based on the observation that the license plate textual information can often be characterized by the distribution of the directions of the image gradients.

A. *HYPOTHESIS GENERATION:*

I. *EDGE EXTRACTION*

Edge extraction detects discontinuities in brightness of the image. Edge detection is used for image segmentation and extraction in areas such as image processing, computer vision, and machine vision as shown in the fig. 7(b).

II. *EDGE FILTERING*

Edge filtering is performed to remove noise from edge, we used median filter, the 2D median filtering system object to filter out slight speckle noise introduced due to segmentation.

The median filter is a non-linear filtering technique used to remove noise from image under consideration. While it helps in removing the impulse noise it preserves the edges. After segmentation filtering was used for remove all lines expect characters. It is take consider noise. It is widely used and it is very effective at removing noise while preserving edges. It is particularly effective at removing 'salt and pepper' type noise. Neighboring vertical edges that remain after filtering are merged by performing a morphological operation.

III. *MORPHOLOGICAL FILTERING*

Binary images may contain numerous imperfections. In particular, the binary regions produced by simple threshold are distorted by noise and texture [6]. Morphological image processing pursues the goals of removing these imperfections by accounting for the form and structure of the image. Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. Morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images. Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels.

The structuring element is a small binary image, i.e. a small matrix of pixels, each with a value of zero or one. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. By choosing the size and shape of the neighborhood, you can construct a morphological operation that is sensitive to specific shapes in the input image. In morphology, the image is processed based on shape of an image. In order to remove gaps obtained along the edges, we need to enhance the moving edges. This enhancement uses the morphological operator's dilation and erosion [20] with an appropriate structural element as shown in the fig. 7(d). Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image**.**

IV. *REGION GROUPING*

The MAKE-SET operation makes a new set by creating a new element with unique identification text as shown in the fig. 7(e). In Region grouping, Region bounding boxes of a sample image are selected and MAKE-SET routine [19] applied to all regions, the beginning each regions a disjoint set created by the MAKE-SET algorithm and The UNION-FIND routine then tries to group two *compatible* candidate regions. Union –find algorithm tracks a set of elements partitioned into a number of disjoint (non-overlapping) subsets, it provides near-constant-time operations to add new sets as shown in the fig. 7(f) to merge existing sets and to determine whether elements are in the same set.

B. *HYPOTHESIS VALIDATION:*

The last stage of our license plate detector is a region classification, which discards regions that do not seem to contain any textual information. We use for this task the T-HOG text descriptor [16] which is a texture classifier specialized for capturing the gradient distribution characteristic of character strokes in occidental-like scripts. We first estimate a center line for each candidate image region, by taking, at each column, the center point between the upper most and the bottom most pixels from the filtered and dilated edge image. For each window, we compute the T-HOG descriptor and use it as an input for a SVM classifier [17]. For the SVM classifier we used a Gaussian $\chi 2$ kernel, whose standard deviation parameter was optimized by

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 13, March 2018

cross-validation on a training set of text and non-text samples. The classifier output is threshold to give a binary text/non-text region classification. The final license plate detection is performed by taking regions containing a large number of windows classified as text as shown in the fig. 7(g). classifier output is threshold to give a binary text/non-text region classification. The final license plate detection isperformed by taking regions containing a large number of windows classified as text as shown in the fig. 7(g).

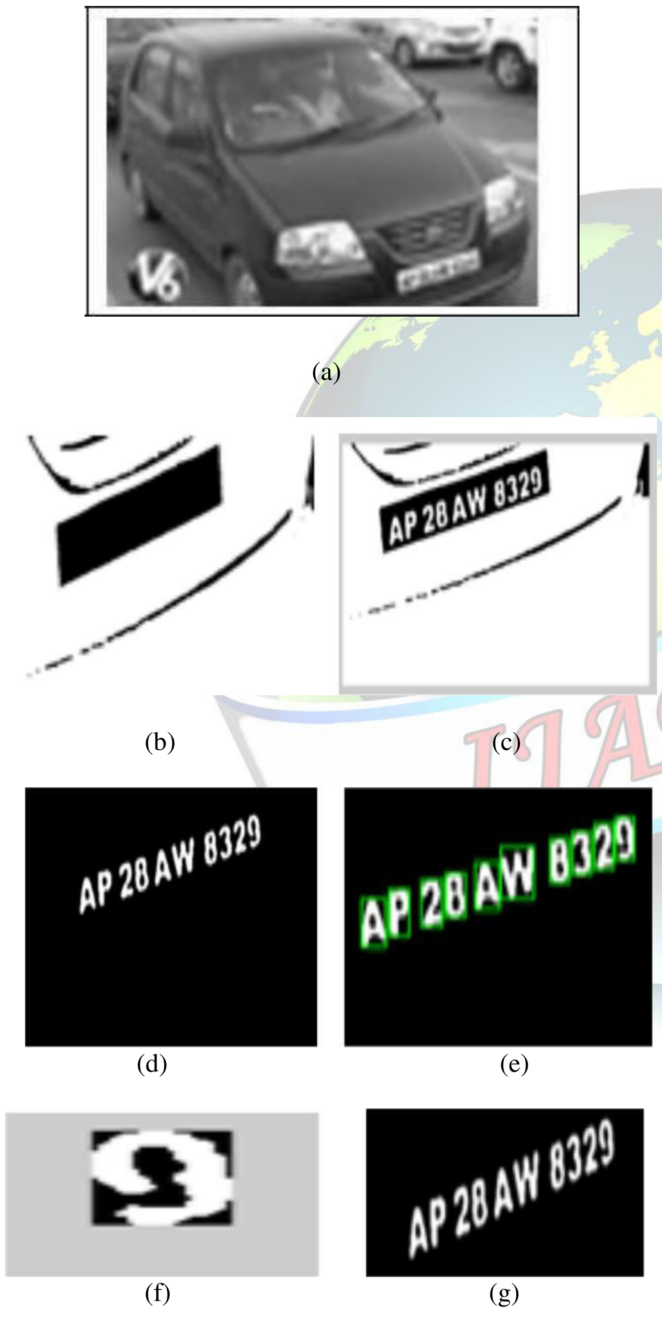


Fig. 7. (a)Input image (b) Edge Extraction (c) Edge filtering (d) Morphological dilation (e)Make-Set (f)Union-Find (g)Region classification.

V. MOTION PREDICTION AND TRACKING

The selection and tracking of a set of distinctive features from the license plate region is the basis of our vehicle speed estimation. The input data of this module are a license plate region, detected at video frame and a sequence of consecutive frames where the features being tracked are visible. The output is a list of motion vectors for each tracked feature in a pair of consecutive video frames. We select distinctive features, only once for each vehicle. The selected features are tracked along the video frames using the Kanade-Lucas-Tomasi (KLT) algorithm [4]. To cope with large feature displacements, e.g. from a vehicle moving at a high speed, the movement is initially estimated by matching SIFT (Scale- Invariant Feature Transform) features extracted from the first two frames in the sequence. Incorrect motion vectors obtained by KLT and the SIFT matching are filtered out by an outlier rejection.

A. *MOTION PREDICTION (SIFT):*

The maximum pixel displacement d that the pyramidal KLT algorithm [20] can handle is given by $d = (2^{l+1} - 1)\delta$, where l is the maximum number of pyramid levels and δ is the pixel motion allowed by elementary optical flow computation. That is, for l= 3 the maximum displacement allowed is about 15 pixels. Increasing the number of levels may allow for large displacements, but the window Ω at a higher pyramid level will include data from a larger neighborhood of I, making the similarity property of Equation (1) harder to hold, and the system more susceptible to incorrect matches.

As the KLT algorithm can use a point's previous motion to estimate its position in the next frame, it can handle large displacements well, as long as the previous results were accurate. However, for the first frame in a sequence (*i.e.* the moment a license plate is detected), there is no known motion. Let I = V (i) be the image where the license plate was detected, and J = V(i+ 1) be the next image from video sequence. If a vehicle is moving fast, the displacement of the tracked points between I and J may be too large for us to use the common

is unknown. To overcome the limitation described above, an initial value for ˉ ˉis estimated by using a different method for matching features extracted from I and J. We have used SIFT (Scale Invariant Feature Transform) [20]. SIFT is a popular method [15] for detecting and describing image key points, being invariant to scale and rotation, as well as robust to illumination variations, and to affine and perspective distortions up to a certain extent. SIFT was chosen for our system mostly because it is a popular and tested method, with implementations readily available.

We define two windows $\Omega 1 \in$I and $\Omega 2 \in$J, which are obtained by expanding the license plate boundaries respectively by $\gamma 1$ and $\gamma 2$ pixels. Here, $\gamma 1$ is a constant used only to allow SIFT features to be properly extracted from the license plate region, since SIFT discards regions too close to the image borders. The value of $\gamma 2$ was selected so that the license plate still appears inside $\Omega 2$, even for fast moving vehicles. SIFT features extracted from $\Omega 1$ are matched to features extracted from $\Omega 2$. We have used the "nearest neighbor distance ratio" matching strategy in which a match occurs if the ratio between the distance from a feature to its nearest neighbor and the distance to its second nearest neighbor is below a given threshold (10, in our experiments). The obtained feature matches can be used to compute the displacement of each SIFT feature between frames I and J. The average feature displacement is used as the initial value for . This process occurs only once for each detected license plate, after the motion is roughly predicted, the system relies on the KLT algorithm [14], which allows for a faster and more accurate computation for the optical flow.

B. FEATURE TRACKING (KLT):

The computation of 2D image velocities, or optical flow, is based on the fact that the intensities of pixels in a small region in two consecutive frames remain constant but the position may change, that is

$I(u,t)=J(u+\vec{\ },\vec{\ }t)+\Delta t$ -----------------(1)

Where $\vec{\ }\vec{\ }$ = (x_d,y_d) denotes the displacement of a point u = (x_u;y_u), and Δt is the time interval to the next frame in sequence. In this context, the goal of the Lucas-Kanade algorithm is to minimize the sum of the squared error between these two images, for a small image region Ω, that is

$E=\sum_{\Omega}[\ (\ )-(\ +\ )]^2$ ---------- (2)

where E is the residual error to be minimized.

to perform an exhaustive search for all possible values of $\vec{\ }$ in

the domain DJ, but this is an expensive brute force approach that requires high computational effort.

To the Lucas - Kanade algorithm [4] assumes that a current estimate of is known and then iteratively solves for increments Δ , namely

$E=\sum_{\Omega}[\ (\ )-(\ +(\ +\Delta\ ))]^2$ -------(3)

Updating the parameter at each iteration

$= \ +\Delta$ ----------------------------- (4)

These steps are interacted, with sub pixel accuracy, until the . The derivation of the Lucas-Kanade algorithm is out of the scope of this dissertation, but it uses the image derivatives of I. An assumption of the LK algorithm is that the motion is small (order of one pixel between the two images). To overcome this limitation, described a pyramidal feature tracking, also known as KLT [20]. More precisely, let I^0, I^1,...,I^l be a multi-scale *image pyramid* [20]. The base I^0 of the pyramid is the highest resolution image I, and each subsequent image (*level*) I^k is a copy of the preceding one I^{k-1}, reduced in width and height by a constant factor, usually of 1⁄2. Therefore level I^k has $1/2^{2k}$ as many pixels as level I^0. The maximum level *l* depends on the size of the original image and on the maximum allowed displacement. The optical flow is computed at the deepest pyramid level I^l. The result of that computation is propagated to the upper level I^{l-1} in the form of a guess for the pixel displacement. Then, the refined optical flow is computed at level I^{l-1}, and the result is propagated to levelI^{l-2}, and so on up to level 0 (the original image. In our system the KLT algorithm is used to align a set of *n*
emplate regions, over the license plate region, to an input image.

C. OUTLIER REJECTION:

An *outlier* is a data point which appears to be inconsistent with the rest of the data [6]. For our system, an outlier is a motion vector that differs significantly from the other computed motion vectors, due to a mismatch. This kind of mismatch, in our system, may compromise the initial guess, resulting in a tracking failure, or may compromise the speed estimation. To prevent against such failures, we used an *outlier rejection* routine, which is applied for the motion vectors obtained by the KLT algorithm, as well as those obtained by matching SIFT features [15].

i i

motion prediction module and refined with sub-pixel accuracy by the KLT feature tracking module [20],

where $\mu x=\frac{\sum_{=1}\ (\ )}{\ }\ \mu y=\frac{\sum_{=1}\ (\ )}{\ }$ ----------------------(5)

$\mu x=\frac{\sum_{=1}(\ (\ )-\mu x)\text{^}2}{\ }\ \mu y=\frac{\sum_{=1}(\ (\ )-\mu y)\text{^}2}{\ }$ ---------- (6)

i = *{*1, 2*...n}* is the index of a set of individual features tracked or keypoints matched. In order to discard outliers, we compute the mean and the standard deviation of the displacements in the *x* and *y* axes.

Then, we discard the motion vectors outside the three-sigma deviation in any direction. This procedure is repeated until the standard deviation in both *x* and *y* axes become smaller than 0.5 pixel. The outlier rejection step applied to KLT tracked features and to SIFT keypoint matches.

VI. SPEED ESTIMATION

In our video we identify each vehicle and to find their corresponding distance covered in consecutive frames [10]. In our approach we track each vehicle and trace their centroid in upcoming frames to get the distance travelled by that vehicle. In this approach we use the array of structure to store centroids of each vehicle. In this as vehicle is arrived into the region of interest in the video their corresponding bounding box is created by that we generated the centroid of the bounding box
[20]. As new vehicle arrived we store its centroid value to the tracks. And updating its centroid value in upcoming frames, and updating its track value of centroid until that vehicle passes out by the video. If the video, we have number of vehicles then we have to track them all simultaneously. And we store their centroid values into the track structure that we created. And updating the centroid of all vehicles simultaneously as the new frame arrived. We keep on updating the centroid values until it is in the region of interest. For the updating of the centroid of the vehicles in consecutive frames [6]. We use the method in that tracked cars update their centroid in their next upcoming consecutive frame by calculating the minimum distance centroid of the current frame [10]. As we know the movement of the vehicle is major in the one direction only. So therefore we update tracked centroid to the minimum distance current

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

centroid. We used the loops for comparing each track distance o each current centroids. As we get the minimum distance centroid we assign that current centroid value to that track [6], and we run this process for each tracked cars previously. Here 3 cases arise as follows:

1) If (Tracked Cars==Current Frame Centroids):

In this we get to know that no. of vehicles remain same in consecutive frame. Therefore we update all tracked vehicle to the current centroid and measure their distance covered.

2) If (Tracked Cars< Current Frame Centroids):

In this we get to know that no. current frame centroid is more than the tracked cars that means new vehicles are arrived into the frame video. So firstly we assign and update the previous tracked cars after that we add new vehicles to the tracked structure.

3) If(Tracked Cars> Current Frame Centroids):

In this tracked cars are more than the current frame centroids, it means that some vehicles crossed their region of interest and passes by their video window frame. So we delete that tracked cars and update remaining tracked cars centroid to the current centroids. By measuring the distance travelled by a vehicle in a
particular time. Here, we calculated speed by measuring the distance covered by the cars from one frame to another by using the formula [10],

$$\text{Speed} = \frac{\text{distance covered in unit of pixels}}{\text{time}}$$

And as we know captured video has frame rate of 26 frames per second.
Therefore, rate of change of one frame to another consecutive frame is 1/15 seconds.

$$\text{Speed} = \frac{\text{distance covered in unit of pixels}}{\frac{1}{26}}$$

and the unit of speed is number of pixels transferred per second. The speed is calculated in the number of pixel travelled per second unit, and then we change it into the km/hr unit by taking the
actual distance measure of the area covered by the camera view as shown in the fig. 8.

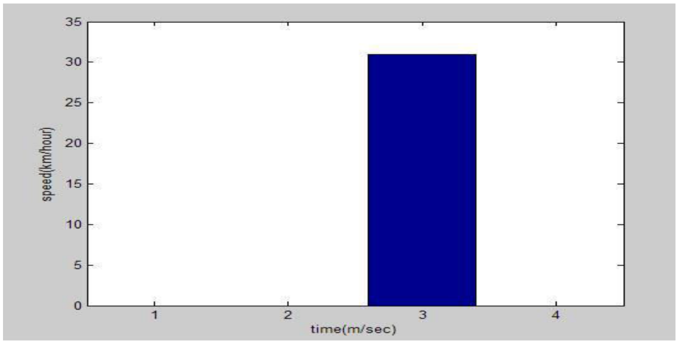


Fig. 8. Speed measurement

.VII. CONCLUSION

This project solves the problem of accidents due to rash driving. Speed and license plate detection plays an important role in traffic surveillances. This system is based on selection and tracking of distinctive features located within license plate region to measure speed. This system is tested for various videos of HD quality with associated ground truth distances from the lane. The distinctive features are extracted in license plate region to reduce complexity to measure speed. This module achieved accuracy of 93.8%. In our experiments from various videos, the measured speeds had an average error of 0.062.

As future work, we intend to verify if estimating the distance of the license plates from the ground can improve the results. We also aim to apply an OCR on the detected license plates in order to create a traffic speed control system with integrated surveillance tools, e.g. to compute the traffic flow, to identify stolen vehicles, etc. Another topic for future work is the implementation on a compact platform that allows local processing, including optimizations such as parallel processing on GPUs, thus reducing communication bandwidth requirements.

VIII. RESULT AND DISCUSSION

Table.1 Data set of Videos

RESOLUTION	Fps	REFEREN-CE SPEED	TEST SPEED	ERROR
720X1280	1/26	32.63	31	5%
600X1040	1/31	38.709	36	7%
720X600	1/28	43.617	41	6%
960X1060	1/24	48.421	46	5%
600X540	1/25	45.652	42	8%

where Fps refers to frames per second.

ERROR CALCULATION ANDACCURACY FORMULA:

£ = 100 − E --------------(1)

E= --------------------- (2)

whereE - Average Error

E - Total number of errors for given videos V -Total number of videos

$$\text{Error}(\%) = \frac{\text{Speed of Reference Video} - \text{Speed of test Video}}{\text{Speed of Reference Video}}$$

Accuracy= 100- Error(%).

CALCULATION FOR THE DATA SET:

E = (5+6+7+5+8)/5 = 6.2%

Accuracy= 100- 6.2 = 93.8%

IX. REFERENCES

[1] T. V. Mathew, "Intrusive and non-intrusive technologies," Indian Inst. Technol. Bombay, Mumbai, India, Tech. Rep., 2014.

[2] N. Buch, S. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," IEEE Trans. Intell. Transp. Syst., vol. 12, no. 3, pp. 920–939, Sep. 2011.

[3] J. Shi and C. Tomasi, "Good features to track," in Proc. IEEE Int. Conf. CVPR, 1994, pp. 593–600.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in Proc. Joint Conf. Artif. Intell., 1981, pp. 674–679.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, 2004.

[6] D. Luvizon, B. Nassu, and R. Minetto, "Vehicle speed estimation by license plate detection and tracking," in Proc. IEEE ICASSP, 2014, pp. 6563–6567.

[7] D. Dailey, F. Cathey, and S. Pumrin, "An algorithm to estimate mean traffic speed using uncalibrated cameras," IEEE Trans. Intell. Transp. Syst., vol. 1, no. 2, pp. 98–107, Feb. 2000.

[8] V. Madasu and M. Hanmandlu, "Estimation of vehicle speed by motion tracking on image sequences," in Proc. IEEE Intell. Veh. Symp., 2010, pp. 185–190.

[9] S. Dogan, M. S. Temiz, and S. Kulur, "Real time speed estimation of moving vehicles from side view images from an uncalibrated video camera," Sensors, vol. 10, no. 5, pp. 4805–4824, 2010.

[10] C. H. Xiao and N. H. C. Yung, "A novel algorithm for estimating vehicle speed from two consecutive images," in Proc. IEEE WACV, 2007, pp. 1–6.

[11] H. Zhiwei, L. Yuanyuan, and Y. Xueyi, “Models of vehicle speeds measurement with a single camera,” in Proc. Int. Conf. Comput. Intell. Security Workshops, 2007, pp. 283–286.
[12] C. Maduro, K. Batista, P. Peixoto, and J. Batista, “Estimation of vehicle velocity and traffic intensity using rectified images,” in Proc. IEEE ICIP, 2008, pp. 777–780.