MACHINE VISION ASSISTED AUTOMATED DRIVING

TEAM AUTOMATON:

Krishnasai Bandarupalli

Karthik Sajeev

IE 590 – Robotics and Machine Vision

Team Automaton

Tuesday, 12/04/2018

ABSTRACT:

The present project explores the idea of powering an autonomous vehicle through machine vision techniques using OpenCV and Python. The main objective of the project is to recreate the function of a Lidar to detect speed, distance and position of an object within the vision range of our vehicle using just a camera, which can be used as the sensor inputs for the onboard control system. In addition to distance and speed calculation, the other inputs that an autonomous vehicle should consider include the lanes and traffic signs which are also tackled in the project. Several techniques such as Hough Transform, Haar Cascade and distance calculating algorithms were considered to effectively achieve our objective. We developed our own algorithm to detect distance and speed and used video properties to convert it to real world distance units (meters). The main challenging aspect of the project is to make the program robust to climatic conditions and terrain. We have achieved our objectives of detecting lanes, objects, distance and speed of vehicles successfully.

Youtube Link: https://www.youtube.com/channel/UC4uPWFxUrwoaRvbeUb2alRw

Tools Used: Opencv 3.4.2
Opencv Contrib 3.4
Python 3.6.4

TABLE OF CONTENTS:

LIST OF FIGURES:

LIST OF TABLES:

1. INTRODUCTION:

The biggest challenge of the 21st century in the automobile industry has been a race against time and competition to create a perfect autonomous vehicle. There are a lot of technical and philosophical challenges in achieving this goal one of which is the vision system. Vision system is the most important part of an autonomous vehicle as it determines the safety and efficiency of entire system. In our quest to learn more about the vision system, we have come up with an alternative method of perceiving the surroundings through cameras alone instead of currently used popular method of Lidar and RADAR systems. We aim to create a system which is cost effective and invariant to different weather conditions and emergency situations. Our project began with an initial plan of detecting the lanes, objects including vehicles, signals, license plates and people followed by detection of relative speed and position of objects. Upon effective detection of these parameters we aim to conduct intensive testing to make the system robust and accurate. The methods and research into this area are explained in the literature review.

2. LITERATURE REVIEW:

2.1 Development of autonomous vehicles:

The development of machine vision techniques in automotive industry has been going on since the early 1980s. According to the level of autonomy in a vehicle, autonomous driving is classified into 5 levels. Level 1 is an assistance level of autonomy where the major functions are still controlled by the driver. An early example of this type of automation is the road recognition algorithm used in the Mercedes-Actros trucks in the early 2000s which gives out audio warnings when the truck moves beyond the center lane by some threshold distance. This process was done by recognizing edge features of roads with lanes based on spacio-temporal models of continuity. This also involves distance detection of objects in front of the vehicle and sending a warning when the object is too close.

Level 2 automation is the amount of automation being used in current crop of commercial vehicles which involves acceleration and steering assistance. It takes a temporary responsibility from the driver but cannot function on its own. The best example of this type of autonomy is the adaptive speed control. Initially RADAR systems were used to detect the objects and their distances for cruise control. This method involves transmitting sinusoidal waves with periodically varying frequencies. When these waves are reflected and detected by the receiver, the received frequency is compared to the current transmitting frequency from which the time period between the signals is calculated. As this time is proportional to the distance, the object distance is determined. This proved to be inefficient due to false detections caused by multiple reflections of the waves. This system doesn't consider the lane curves which is later tackled by introducing multiple cameras. Thus, a lane detection algorithm coupled with RADAR system was developed to determine the object distance effectively to improve the performance of adaptive speed control.

Level 3 automation is a shift from 2 where the vehicle takes control with some human interaction. This is done using sensors like Lidar and machine vision techniques. Improvement in object detection lead to a shift in the preference to Lidar over RADAR due to its high accuracy and efficiency. This method involves in transmitting several thousand pulses of light at the target and measuring the time taken for them to reflect to the receiver. As speed of light and time of travel are known, the distance of the object can be easily determined. Although Lidar is accurate, it is expensive and bulky to operate.

Level 4 and 5 are improvements in the control of the vehicle rather than improvements in the sensors. In these levels, the control system can handle all operations of a vehicle based on the input from the sensors. The only difference is that the driver is first notified and consulted in Level 4 whereas level 5 is complete automation.

2.2 Use of machine vision in autonomous vehicles:

Lin and Wang [2] designed a vehicle parking assistant system by making use of machine vision techniques. A camera installed on the rear bumper of the vehicle is used to capture video images in the perspective view. A model called top-view transformation is proposed to convert this to bird's eye view of the area behind the vehicle. A 'fitting parameters searching algorithm' is used to predict the coordinate transformation needed, thus avoiding the need for manual setting of camera parameters. Experimental results show that the proposed approach is fast and generates a clean and accurate bird's eye view of a perspective view.

Inverse perspective mapping is a concept which is used to isolate and get a clear picture of the region of interest. Lane detection usually involves in the detection of lines which can be effectively done through Hough transform. Hough transform takes an input from an edge detector (For example Canny) and detects location of all the lines. It's based on the logic that when a line in (X,Y) plane; which has a polar form of $\rho = X\cos\Theta + Y\sin\Theta$; is transformed to (ρ,Θ) plane, the intersection point of ρ and Θ lines represents the whole length of the original line. Thus, every point in Hough plane represents a line.

Escalera and Mata [4] propose a traffic sign recognition technique good enough to be used for driver support systems. The areas of the image that satisfy certain color restrictions are first identified and genetic algorithm is applied for sign detection. Next, information extraction is done on the detected areas and neural network is used for traffic sign classification. The developed approach for traffic sign recognition is found to be invariant to perspective transform, lighting changes, partial occlusion, object deformations and shadows.

Lienhart and Maydt [6] made use of rotated haar-like features to achieve faster face detection. They emphasized working with features instead of raw pixels, enabling easier classification. They also introduced a novel post optimization technique for improving the false alarm rate. The experimental study compared two face-detection systems, one using basic and the other a haar-like feature set showed a 10% lower false alarm rate. A third system with post optimization and haar managed to achieve 24% improvement over the basic system.

Many methods to detect speed through image processing were developed for stationary cameras as shown in [5]. This paper explains a novel method of detecting vehicles through license plates recognition. Since the camera used here is stationary, speed was calculated by taking the time difference between entry and exit points of the vehicle and calculating change in position in consecutive frames. License plate was detected through text detection algorithm and training the system like Haar cascade. The experimental results shown in Table 3.1 prove that the speed calculated by this method is as accurate as the one measured by Lidar and RADAR systems.

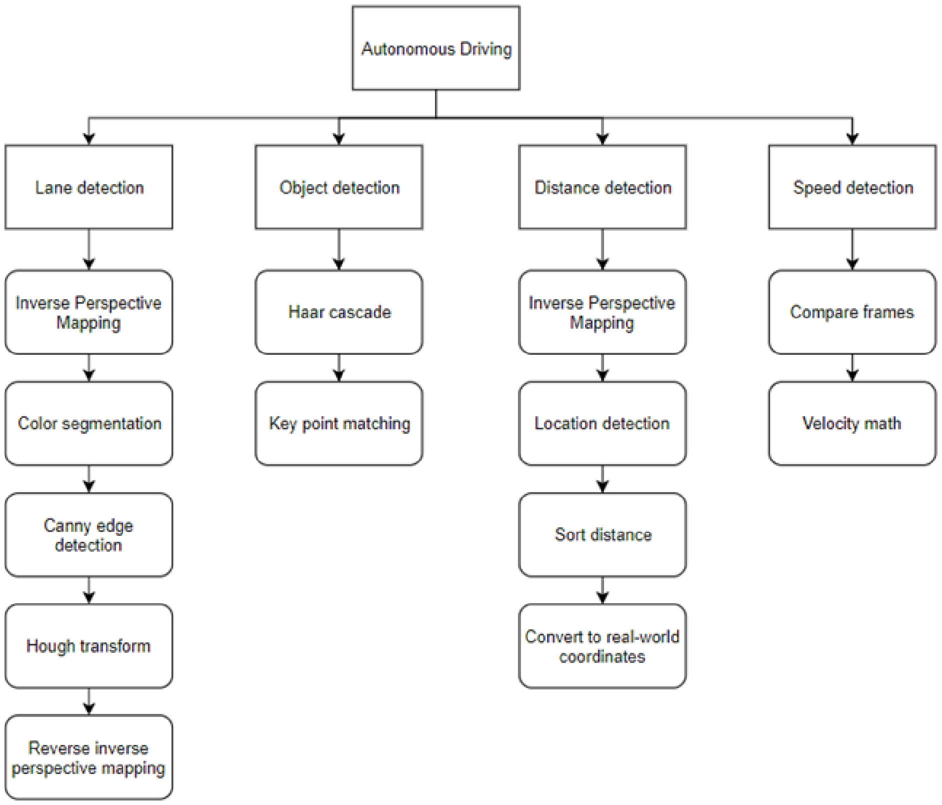
Actual velocity	Measured velocities
43 ± 1	43.3; 44.0; 42.9; 44.1; 42.6; 44.0
55 ± 1	54.9; 56.0; 55.9; 56.2; 56.9
71 ± 1	70.7; 71.1; 71.0; 69.1

Table 1: Experiment results for speed [5]

UAVs used for remote sensing purposes may be used for detecting and counting cars. Moranduzzo and Melgani examine this possibility in their paper [8] by using SIFT to extract features and to successively discriminate between cars and other objects using support vector machines. Their approach was two-fold: first they extracted a set of points invariant to affine transforms. Next, the classifier is trained and used to identify points belonging to the car class.

To detect distance, a combination of object tracking techniques and calibration techniques can be used. In the research paper by Christian Kollmitzer [7], objects were detected and tracked using a dual stereo camera setup that was independently calibrated. The centers of these objects were detected and tracked across multiple frames. From the information on focal length of the camera, object size and time between frames, the distance and speed of the object being tracked was measured.

3. METHODOLOGY:



4. DETECTION OF LANES:

4.1 Inverse Perspective Mapping:

The output obtained from a dashcam is in perspective view, which makes it difficult to detect lanes. Applying Hough transform on the perspective image results in a lot of false detections. Also, the perspective image distorts the objects in the image, which makes future measurements and calculations erroneous. Consider the white, dashed lane lines in the two images below. Looking at the perspective view in Fig. 4.1, the algorithm might be tempted to believe that the lines are of different lengths, while they are in fact of the same length. This becomes clear when we look at the bird’s-eye view in Fig. 4.2. Thus, inverse perspective mapping is applied to convert perspective view to bird’s eye view. This is done by manually choosing four points in the perspective view to be the four corners of a quadrilateral that we need to transform to rectangle. Thus, the area in the interior of this quadrilateral alone will be considered for lane detection. This method thus helps to avoid false positives present in the background by strategically choosing the region of interest as road alone.

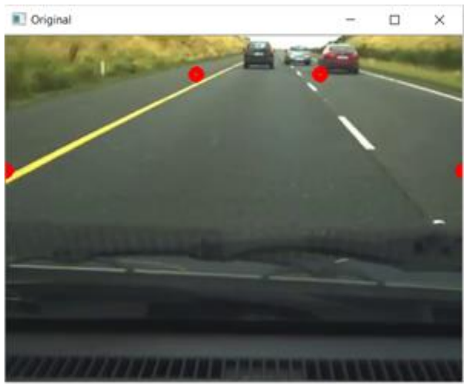


Fig. 4.1: Perspective view

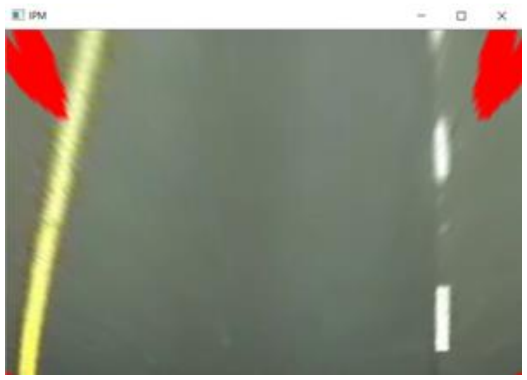


Fig. 4.2: Bird’s-eye view

4.2 Color Segmentation:

Once bird’s eye view is obtained, we can use color segmentation to further isolate regions of interest. In our case, we know that lanes are either white or yellow in color. We thus specify the color ranges for these colors and through trial-and-error, find the values that give the best segmentation. The output is shown below:

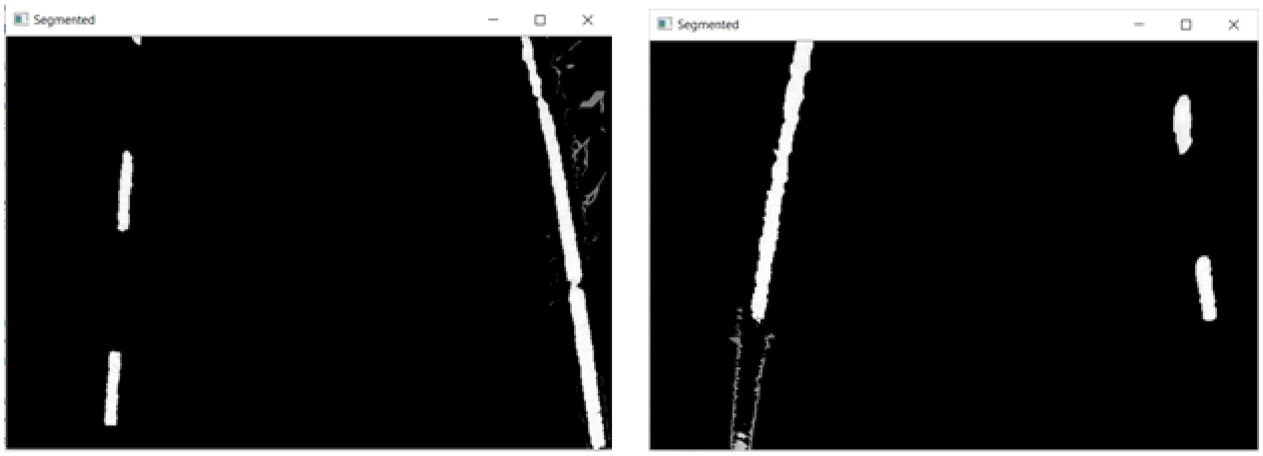


Fig. 4.3: Color-segmented images

As seen above, while we are able to separate white and yellow lanes, it does not always have a perfect detection. Also, in case there are white or yellow cars in the bird’s-eye view, they would also appear in the color-segmented image. We need to thus apply further processing in order to separate the lanes.

4.3 Canny Edge Detection:

From the color-segmented image, the lanes are extracted using edge detection techniques. This is achieved by performing gaussian blur on the image, followed by Canny edge detection. This also acts as a preparatory step for Hough transform, which needs an edge-detected image to work correctly.

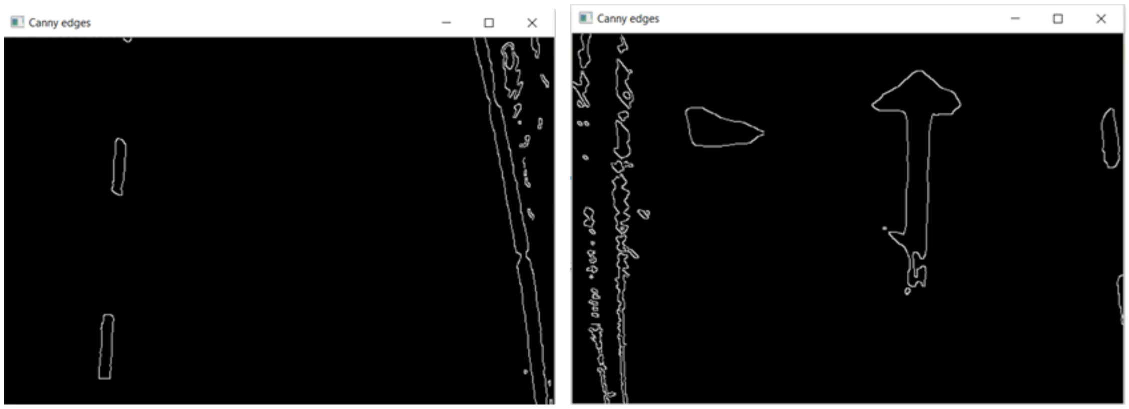


Fig. 4.4: Canny Edge Detection

It is seen that in addition to lane lines and white/yellow cars, our color-segmented image also highlighted road markings, which happen to be in the same specified color range. This will be dealt with later.

4.4 Probabilistic Hough Transform:

Probabilistic Hough Transform is a method that detects lines in an image. Usually Hough transform is computationally intensive since it takes all the points in the image. Probabilistic Hough Transform is an approximation where it considers a random subset of points to find the lines. The output of this function consists of a matrix with coordinates of all the detected lines. These coordinates can then be used to draw smooth lines, in place of short and disjoint lines that Canny returned.

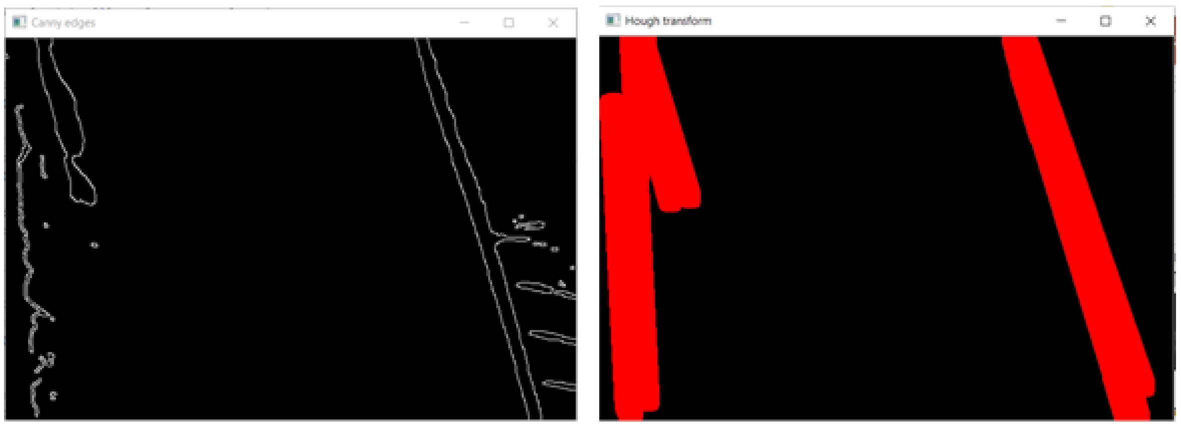


Fig 4.5: Canny edges and the corresponding Hough lines

We also came up with a strategy to get rid of grid-lines that appear in the middle of the road. We realized that it is possible to derive the slopes of Hough lines using their end-points. Legitimate lane lines would thus be within a specific range of slope (assuming the car moves approximately parallel to the lanes). We determined the range of slope variation through trail-and-error and specified this as a condition to eliminate stray detections while drawing Hough lines. The results can be seen below:

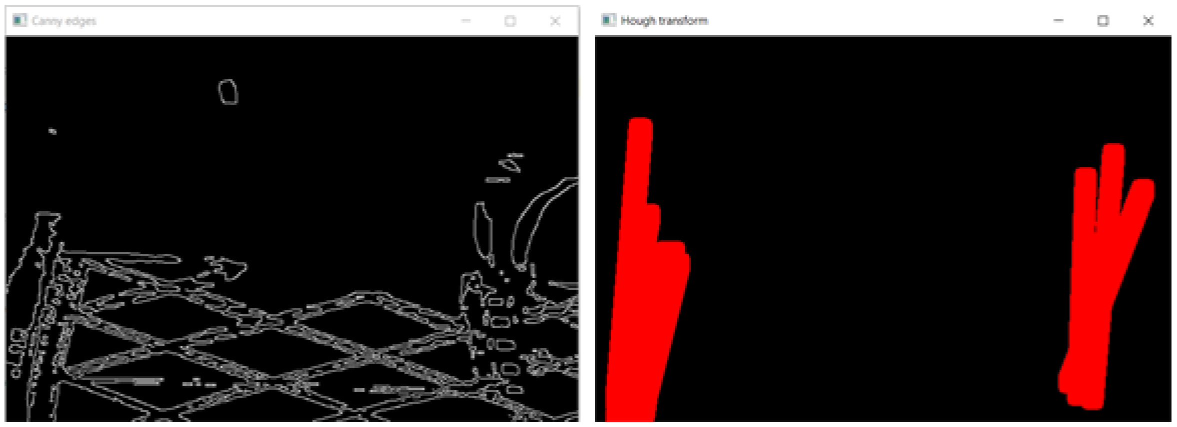


Fig 4.6: Filtering lines based on slope

4.5 Reverse Inverse Perspective Transform:

To transfer the detected lanes from mapped image to the original image, inverse perspective transform is performed to convert bird’s-eye view back to perspective view. The results are shown:

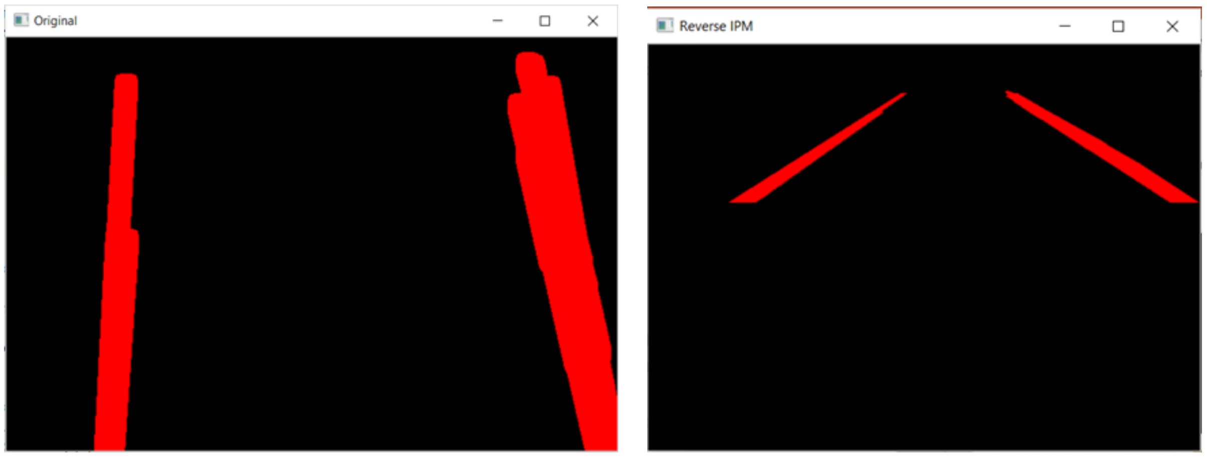


Fig 4.7: Hough lines in bird’s-eye and perspective views

4.6 Draw Final Lanes

The Hough lines in perspective view may directly be superimposed on our original image, but we realized that the results obtained are not always smooth. After all, Hough transform returns a set of lines which would hence draw multiple lines over a single lane. In order to avoid this, we decide to find the four extreme points of the Hough lines in the perspective view: top-left, top-right, bottom-left, bottom-right. This is done by traversing through every pixel in a specific direction, ignoring black pixels and making note of the first red pixel that we come across. After four points are obtained, smooth lane lines may be drawn by joining the top-left point with the bottom-left, and top-right point with the bottom-right. This technique also helps avoid the road markings (arrows) seen in 4.1.2, which could not be eliminated even after specifying the slope range.

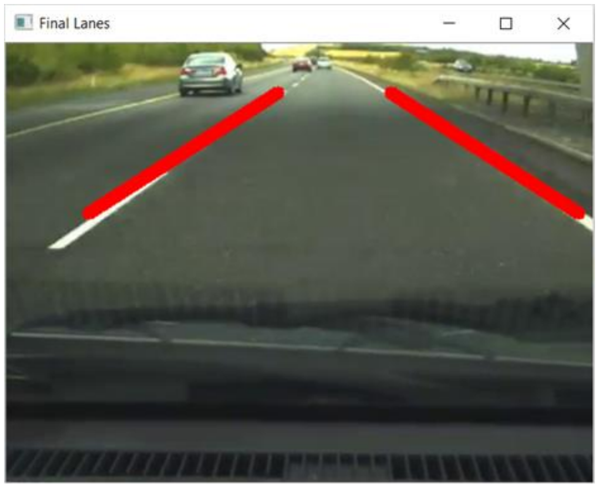


Fig. 4.8: Final lanes

5. DETECTION OF OBJECTS:

A Haar cascade is a commonly used classifier used to identify particular objects. The method used is to superimpose positive images containing the object of interest over negative images which do not have the object. By comparing the two scenarios, the presence/absence of the object and its location on the source image is determined.

For common objects like cars, license plates, people, sign boards, faces and eyes, Haar cascades exist online in the form of xml files. These files may be downloaded and linked to OpenCV in Python to determine the corresponding object in our source image. In our case, each frame of the input source video is considered as an image and the Haar cascade for the required object is used to identify the same. The detected item is shown by a box surrounding the object.

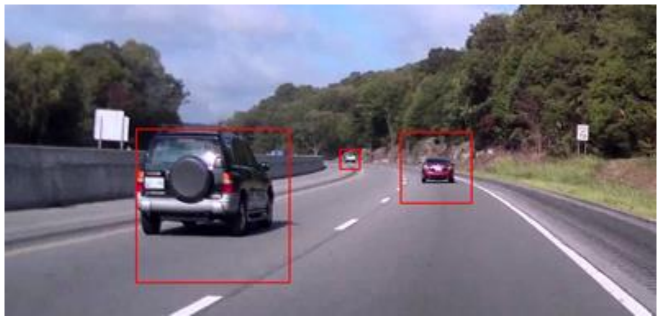


Fig 5.1: Car Detection

5.1 Car detection:

While Haar cascades helped us in detecting cars, we realized that the detection was not consistent. This meant there were many frames where the Haar cascade failed to detect anything despite the presence of a car in the frame. In order to improve car detection, we made use of key point matching to identify the presence of cars in consecutive frames and were thus able to achieve consistent detection. The steps to achieve the same have been detailed below:

1. Use Haar cascade for initial car detection and mark the returned rectangular area as the region of interest (ROI).
2. Use SIFT to obtain feature descriptors within the ROI.
3. Compare features between consecutive frames using FLANN matcher.
4. Use k-nearest neighbors to identify the 2 closest neighbors and use Lowe's ratios test to determine if one or none should be added to the list of good matches.
5. If number of good matches is greater than a threshold, draw rectangle around the detected object.
6. Otherwise, print 'Not enough matches are found'.

The above-mentioned algorithm thus helps to consistently detect cars in consecutive frames, provided the initial detection is done using a Haar cascade.

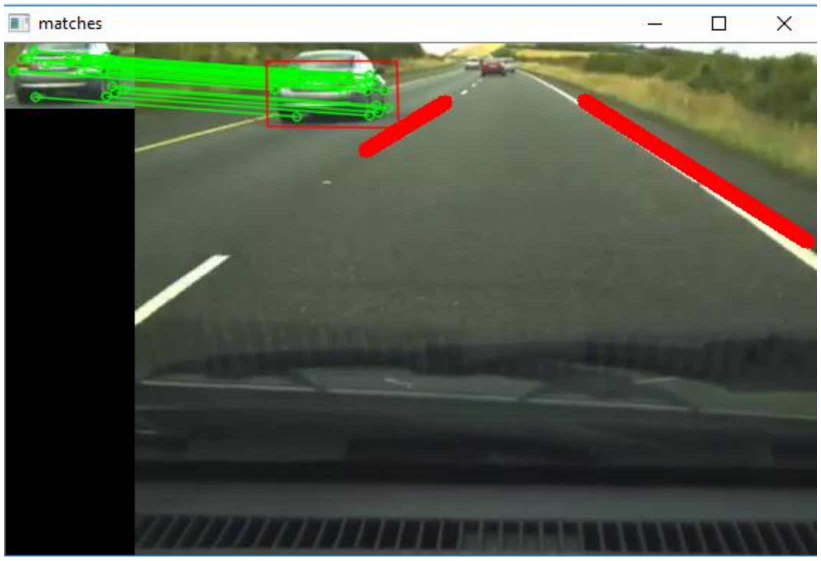


Fig 5.2: Feature matching

5.2 Traffic sign detection:

Traffic signals and signs are detected by training the Haar Cascade similar to car detection. Separate xml files are obtained for traffic signs are included in the algorithm.

6. DETECTION OF DISTANCE:

6.1 Detection of object height in image plane (y):

After lane detection and object detection, the next part of our project involves in the distance and speed calculation. Since the camera outputs a perspective image, calculation of distance results in a perspective error. To eliminate the error due to perspective view, the image was converted to bird's eye view using warp perspective function in OpenCV. This image is filtered to remove all objects except the object box.

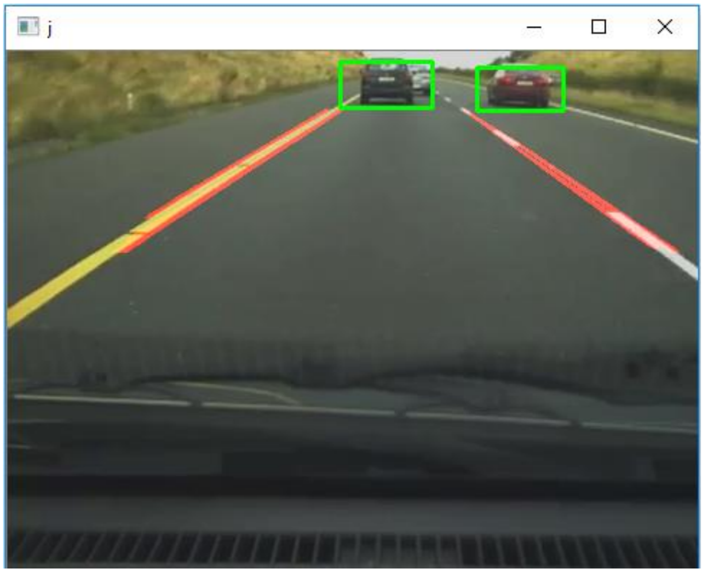


Fig 6.1: Output of Haar Cascade

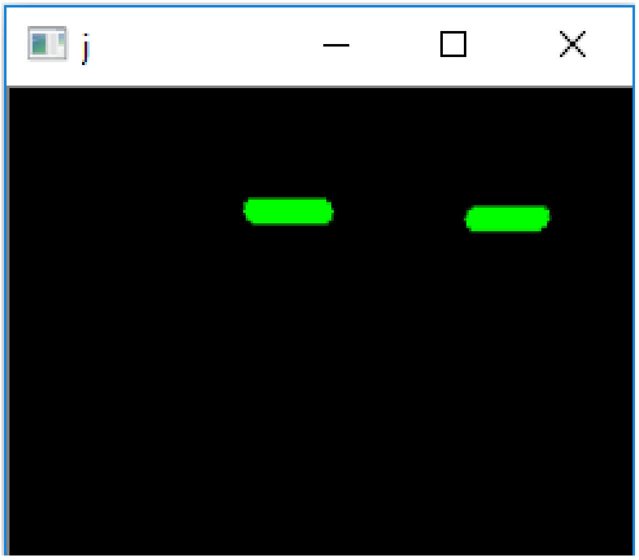


Fig 6.2: Bird's eye view

From the bird's eye view, the co-ordinates for the bottom edges are detected and their distance from the camera lens was determined in pixels. This process is done by scanning the screen (Shown in Fig 6.2) from bottom right corner and finding the first instance of green color. Multiple detections of the same line are avoided by thresholding the minimum width between points to be 40 px and minimum height to be 15 px. The distance of this point from bottom edge of the image was taken as height of object in image plane (y).

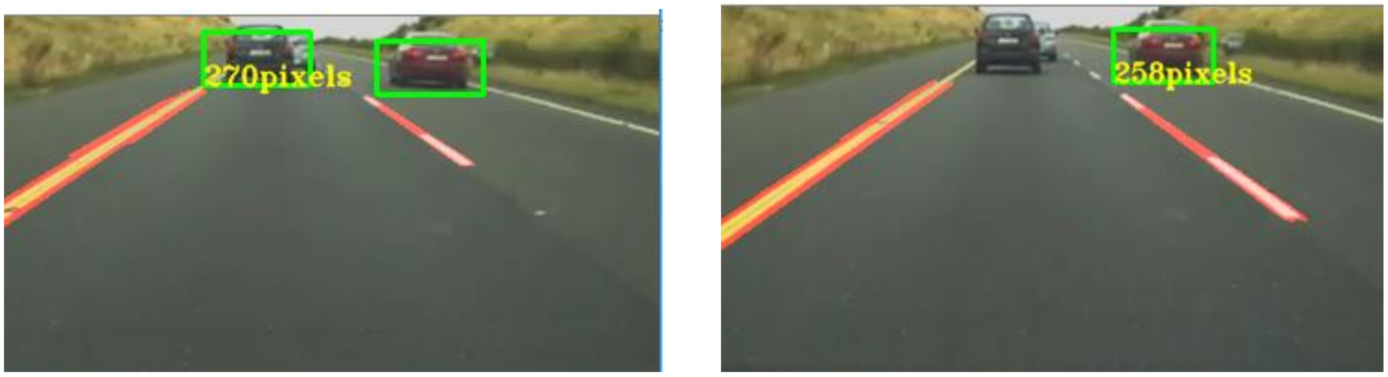


Fig 6.3: Distance Calculation in pixels

6.2 Finding distance in meters:

In order to determine the height of the car in object plane we had to establish a relationship between image plane and object plane. This relationship was established using the following transformation matrix.

$$s * \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f & 0 & Cx \\ 0 & f & Cy \\ 0 & 0 & 0 \end{pmatrix} * [I|0] * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

We chose the origin of world co-ordinate system and origin of camera co-ordinate system to be the same which results in the homogenous transformation matrix [R|t] to be [I|0]. To make this equation work we require the focal length of the camera which was determined from the frame shown in fig 6.3

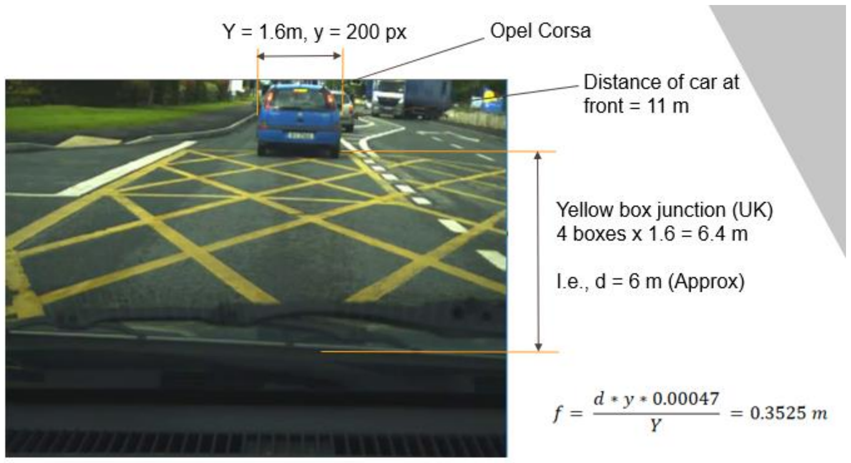

Y = 1.6m, y = 200 px
Opel Corsa
Distance of car at front = 11 m
Yellow box junction (UK) 4 boxes x 1.6 = 6.4 m
I.e., d = 6 m (Approx)
$f = \frac{d * y * 0.00047}{Y} = 0.3525\ m$

Fig 6.4: Calculation of focal length

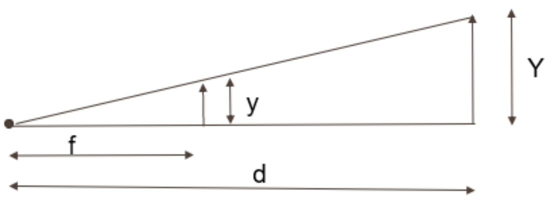
In fig 4.10, The width of the car in real world (Y) was found to be 1.6 meters which was determined from the spec sheet of Opel Corsa. The width of car in image plane was found to be 200 pixel which can be converted to meters by multiplying the size of a pixel which was found to be 0.00047 meters. The distance of car from the camera in real world was found by using the yellow box junction in the image. According to UK traffic sign manual, Yellow box junction is divided into equally sized squares. Since there are around 4 boxes between the camera and the blue car, we determined the distance to be approximately equal to 6 meters. From the data we calculated the focal length to be 0.3525 meters.

Since we know the focal length (f) and center of image (Cx,Cy), using the relation shown above we were able to establish a relationship between image plane and object plane.

$$s * \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f & 0 & Cx \\ 0 & f & Cy \\ 0 & 0 & 0 \end{pmatrix} * [I|0] * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2.836 & 0 & -0.32 \\ 0 & 2.836 & -0.24 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} * s$$

$$Y = (2.836 * y * s - 0.24)$$



$$\frac{y}{f} = \frac{Y}{d}$$

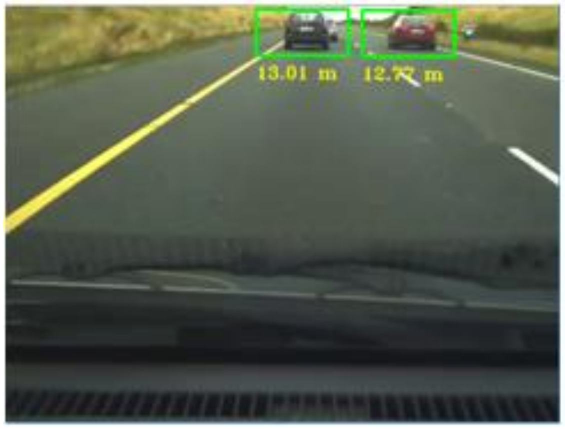
$f = \frac{d * y * 0.00047}{Y}\ m$

f – Focal length
d – Distance of object from camera
y – Height of object in image plane
Y – Height of object
Size of each pixel = 0.00047 m

Fig 6.5: Calculation of real-world height(Y)

Fig 6.6: Camera properties

Thus, we determined the height of object in image plane (y), the height of object in world co-ordinate system (Y) and focal length (f) from which the distance (d) of car in real world was calculated using the relation d = (f*Y)/y. An appropriate scale was chosen as 8.5 which gave us the distance of 6 meters as shown in fig 6.7.



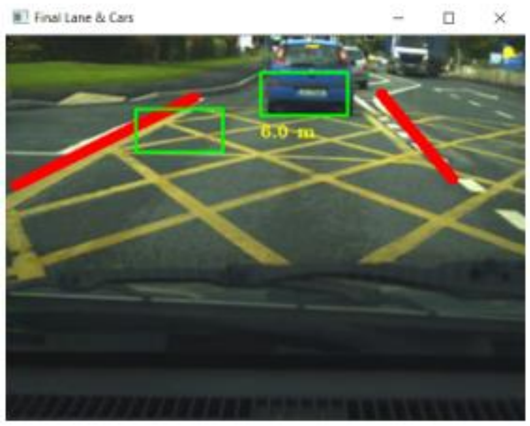


Fig 6.7: Distance Calculation in meters

7. DETECTION OF SPEED:

Speed was calculated using conventional formula of differentiating distance over time. Cartesian co-ordinate system was used for both distance and speed calculation with top left corner as origin. Since the cars are detected in consecutive frames, their co-ordinates are obtained. Noise was reduced by taking a threshold value of 15 pixels for change in distance along both X and Y directions. Once the condition is satisfied, the change of height between the two frames was calculated. Since we are using a video which is 15 frames per second, time was calculated by multiplying the number of frames by 66 milli seconds. Thus, speed was calculated by the following formula:

$$Speed = \frac{(D1 - D2)}{\left(\frac{1}{fps}\right)} = \frac{D1 - D2}{0.066}$$

Where D2 – Distance of object in frame F2 and D1- Distance of object in frame F1.

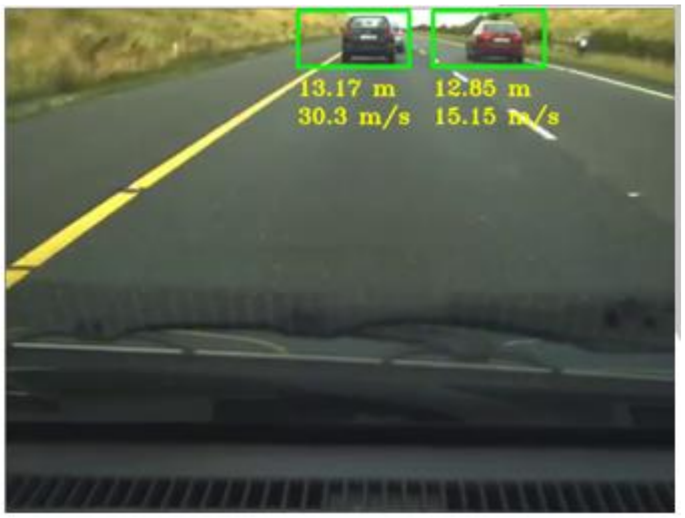


Fig 7.1: Final Speed

8. EVALUATION

:

The performance of the developed lane and vehicle detection techniques is evaluated by making use of a confusion matrix. The matrix illustrates the accuracy of detection of lanes, cars, distance and speed by looking at 200 frames in the video clip. The results obtained are shown below:

	Cars detected by algorithm	**Cars missed by the algorithm**	**Distance detected by algorithm out of total detected cars**	**Speed detected out of total detected cars**	**Lanes detected out of 200 frames**	**Lanes missed out of 200 frames**
Cars in video (Total –421)	114	307	108/114	88/114	-	-
No cars in video	1	-	0	0	-	-
Lanes in video (Total – 200)	-	-	-	-	193	7

Table 2: Confusion Matrix

To summarize the performance of our model, we can look at the following statistics:

Lane detection: 96.5 %

Vehicle detection: 27.1 %

Distance detection given car is detected- 94.7%

Speed detection given car is detected- 77.2%

It can be seen that lanes are detected in most frames. In fact, the only frames where our model failed to detect lanes was when our car was shifting lanes, and hence there were no lanes within the specified region of interest. The low value of vehicle detection can be attributed to the fact that most cars missed by the algorithm were considerably far away from our camera, and hence went undetected by the Haar cascade we used. For cars that were reasonably close, the detection rate was much higher. Distance detection seemed to be pretty robust as most frames which had identified a car also had a distance value printed. Some distance values went missing however, due to the fact that the detected car was outside the region of interest used for inverse perspective mapping while calculating distance. The lower success rate of speed detection compared to distance detection can be attributed to the fact that calculation of speed requires the presence of distance in two consecutive frames.

9. FUTURE WORK:

We have successfully achieved our goal of detecting lanes, objects, distance of objects and calculation of relative speed of the cars. The next steps involve extensive testing and elimination of false positives. We are planning on the following steps to refines our program:

- The current program requires us to enter the 4 corners of the road for inverse perspective mapping which limits the region of interest. This resulted in loss of information. We are planning to make the IPM process more dynamic by creating an algorithm to detect the 4 corners of the entire road automatically.
- The algorithm needs extensive testing under different road and weather conditions to make it more robust and accurate.

10. REFERENCES:

1. E.D. Dickmanns, "The development of machine vision for road vehicles in the last decade", *Intelligent Vehicle Symposium, 2002. IEEE.*
2. Lin, Chien-Chuan & Wang, Ming_shi. , "A Vision Based Top-View Transformation Model for a Vehicle Parking Assistant Sensors" *(Basel, Switzerland), 12. 4431-46. 10.3390/s120404431.*
3. Minxian Li, Chunxia Zhao, Yingkun Hou, Mingwu Ren, "A New Lane Line Segmentation and Detection Method based on Inverse Perspective Mapping", *doi: 10.4156/jdcta.vol5. Issue4.28*
4. Escalera, Arturo de la, Jose M. Armingol and Mario Mata, "Traffic sign recognition and analysis for intelligent vehicles." *Image Vision Comput. 21 (2003): 247-258.*
5. Witold Czajewski and Marcin Iwanowski, "Vision-Based Vehicle Speed Measurement Method", *DOI 10.1007/978-3-319-46418-3_41*
6. R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection" *Proceedings. International Conference on Image Processing*, Rochester, NY, USA, 2002, pp. I-I.
7. Christian Kollmitzer, "Object Detection and Measurement Using Stereo Images", *DOI: 10.1007/978-3-642-30721-8_16 (2012)*
8. T. Moranduzzo and F. Melgani, "A SIFT-SVM method for detecting cars in UAV images," 2012 IEEE International Geoscience and Remote Sensing Symposium, *Munich, 2012, pp. 6868-6871. DOI: 10.1109/IGARSS.2012.6352585*