Lane Following and Obstacle Detection Techniques in Autonomous Driving Vehicles

Phanindra Amaradi[1], Nishanth Sriramoju[2], Li Dang[3], Girma S. Tewolde[4], Jaerock Kwon[5]

Electrical &computer Engineering, Kettering University

1700 University Ave, Flint, MI, 48504, United States

{[1]venk2168, [2]srir3757, [3]dang8220, [4]gtewolde, [5]jkwon}@kettering.edu

***Abstract*-This paper presents techniques developed for lane following and obstacle detection capabilities for autonomous vehicles. The proposed system is based on a single 120 degrees fish eye camera and a 180 degrees scanning LIDAR sensor. For the lane detection a modified lane marking technique was used representing the position of the lines making the lane with respect to the position of the camera mounted on the vehicle. The paper describes implementation of the lane detection system using Hough transform to detect the lanes. Based on the detected lane markings, the position of the relative vehicle is determined as to whether it is in its lane or if it has drifted outside the lane. The system shows the offset amount, as a percentage of distance, it has drifted from the centre of the lane. The SICK LIDAR LMS-291 S-O5 laser range sensor is adopted for the obstacle detection.**

***Keywords*— Lane Detection, Obstacle Avoidance, Image Processing, Video Processing, Hough Transform and Computer Vision.**

I. Introduction

Lane and Obstacle detection techniques play one of the most important roles in Advanced Driver Assistance Systems (ADAS). ADAS acts as an assistant to the driver while driving. Most of the major car manufacturers, universities, research institutes have recently been focusing on using various kinds of ADAS systems such as ACC (Adaptive cruise control), CACC (Cooperative Adaptive Cruise Control), LDW (Lane Departure Warning), FCW (Forward Collision Warning) [1], BLIS (Blind Spot Information System), AFS (Adaptive Front lighting System), PCW (Pedestrian Collision Warning), V2V and V2I (Vehicle to Vehicle and Vehicle to Infrastructure) communications, and Start/Stop systems have been presented.

An Accident can occur within a split second due to driver's inattention or drowsiness. During the course of a long drive returning back from a holiday or after a stressful and tiring day at work; drivers tend to fall asleep for a few seconds, which could potentially result in accidents. To reduce the rising mortality rate due to traffic accidents, a system needs to warn the driver if the vehicle is drifting out of its lane. Therefore many research teams are working to develop and deploy intelligent vehicle systems to assist the driver, to avoid collisions or to control the vehicle autonomously [2]-[4]. Lane Detection System is one of the most studied areas among intelligent vehicle systems. The lane detection system involves the process of identifying the lane markings and estimating the position of the vehicle from the centre of the lane. Fig. 1 illustrates an example of lane detection scenario.

In addition to lane detection, obstacles detection is an important feature for ADAS and autonomous vehicles. There are a few options for obstacle detection, such as RADAR, SONAR, and LIDAR. LIDAR is a laser measurement system that emits infrared laser beams and measures the angle of incidence and distance of objects that are obtained by the beams. Using these measured angles and distances the vehicle can scan points that are within a buffer zone around itself to detect objects. In the project presented in this paper the SICK LMS-291 LIDAR sensor is utilized for obstacle detection.

The main objective of this paper is to implement an effective lane following and obstacle detection system. The work which develops image processing based lane detection system is comprised of the following steps: filtering, edge detection, lane detection and offset distance calculation based on position of the vehicle. After the processing is performed the control system issues commands to the vehicle to adjust its steering according to the offset so it stays close to the centre of the lane. Besides the lane following, the system also performs obstacle detection by using the LIDAR measurements of distances to nearby objects.

The Rest of the paper is arranged as follows, Section II describes the related literature. Section III presents the design details of the proposed system. Section IV provides results of various experiments conducted on the system. Finally, section V offers conclusion and directions for future work.

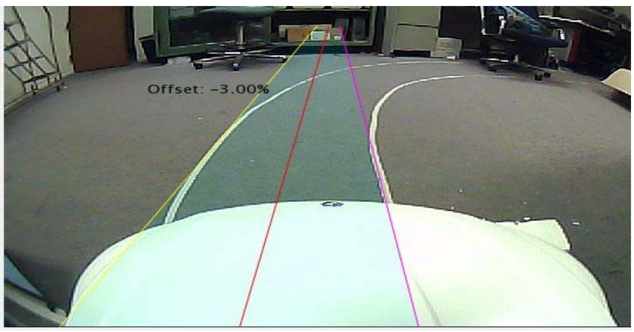


Figure 1: Lane detection scenario

II. Related Work

The related literature in autonomous vehicles covers research in the area of autonomous navigation through waypoints to reach target locations from given start points by using global-positioning system (GPS) to track locations. The accurate location of the vehicle is obtained from the GPS device in a standardized NMEA0183 string format. These strings are passed through a parser that calls a specified function when a complete set of sentence is received. This function then updates the current position of the vehicle. While using the GPRMC string, it is also possible to get the current heading using only the GPS [5]. To perform waypoint navigation, the desired heading value is calculated using trig functions based on the current position of the vehicle and the current target waypoint. There are many systems which are based on using single/dual cameras to detect the lanes. Sensors especially LIDAR and RADAR are used in detecting obstacles. In some of the works, 3D modelling and simulation of the road environment is generated for testing the autonomous driving vehicles. Other related works are focused on observing the driver's behaviour using camera facing the driver to detect possible drowsiness, sleepiness, and fatigue based on the eye-movement, detection of yawn and head rotation.

Other related works are based on traffic sign detection and recognition, and simulating a 3D model of road and environment (e.g. [6], [7], and [8]). For detecting and avoiding collision, they focused on LIDAR sensor data for classifying objects as static or dynamic, and tracking targets using extended Kalman filter. In lane tracking for example, once the lane markers are detected in the first frame, a Kalman filter could be used in order to predict where the lane marks will be in the next video frame, then evolve step-by-step and frame after frame. Every Kalman prediction identifies a very thin region of the next frame, in which we can perform a local search in a very efficient way. Srinivasa et al. [7] present forward collision warning system using forward looking camera and radar data.

Urban as well as countryside locations are used by many researchers to demonstrate the example of lane detection and departure warning system. Most of these works have used image processing, computer vision, and machine learning approaches for detecting and extracting the lane features. Many of the works assume straight and plane roads and work only on highways. This happens because of assumption of a strong lane marking, clear visibility, and low traffic. Another work based on vanishing point detection [10] gives a good result on countryside roads.

III. Design of the Proposed Lane and Obstacle Detection System

Lane Detection

The Lane detection algorithm is a mechanism which detects the driving lane boundaries and estimates the geometry and position of the lane in 2D coordinates, with respective to the vehicle position. Lane detection algorithm is critical for building various applications, such as: lane keeping and lane departure warning system.

Lane keeping actively applies torque to the steering wheel, to avoid unintentional drift of the vehicle out of the lane. Based on the lane boundary geometry, the steering angle decides how much torque is required to be applied in order to steer the vehicle, which prevents the vehicle from veering off the lane. Lane departure warning system uses the information from the lane detection algorithm, to estimate whether the vehicle is about to cross the lane boundary within a short time. In that situation, and if the driver did not plan to signal his intent to switch the lane, a warning is produced to alert the driver.

Lane detection is a most researched area in the field of computer vision with applications in autonomous driving vehicles and ADAS. A lane detection algorithm must be capable of detecting all manners of lane markings from possibly cluttered roadways and filter them to produce an estimate of position of the vehicle and trajectory respective to the lane markings. To develop a solid lane detection algorithm that can work in a variety of environmental conditions, our proposed system integrates the lane markers detection with lane tracking algorithm.

In order for a lane detection system to be robust [11] it should satisfy the following requirements:

- It should be capable of detecting image features which correspond to road markings.
- It should ignore irrelevant road markings, i.e. text on road surfaces or road markings not parallel to the vehicles lane.

The process of the lane detection system includes the following operations: smoothening, edge detection, line detection, and lane detection.

Removing noise from the image is very important because the presence of noise in the system will affect the detection of the edges. The process of Smoothening involves removing of noise from the image by using one of the various [12] filters. Then an edge operator is applied to generate a binary edge map. This is followed by the application of the Hough Transform to the edge map to detect lines. Hough Transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough Transform is most commonly used for the detection of regular curves such as lines, circles and ellipses, whereas Hough Line is a transform used to detect straight line in lane boundaries.

Three fundamental steps performed in edge detection system are [13]:

- Smoothening : application of filters for noise reduction
- Edge point detection: this operation extracts all the points in the image that are potential members to become edge points.
- Edge localization: is a process to select a member from the edge points only, the points that are truly members of the set of points incorporated in an edge.

There are many edge detection methods [14] that can be used in MATLAB. Figure 2, below shows Canny, Sobel, Prewitt, Log (Laplacian of Gaussian) edge detection methods that operate on the intensity of an input image. The figure illustrates the results of operations of the different filters to find edges in an image of a stop sign board. When comparing the different edge detection filters, one can observe their differences in the detection of the edges. The Canny edge detection looks for local maxima of the gradient of intensity in finding edges and the edge function calculates the gradient using the derivative of Gaussian filter. The Sobel edge detection function uses Sobel approximation to the derivative. It displays edges at those points where the gradient of intensity is maximum. The Prewitt edge detection returns at those points, where the gradient of intensity is maximum. Log (Laplacian of Gaussian) edge detection finds edges by looking for zero-crossings after filtering the intensity value with a Laplacian of Gaussian filter. Comparing all the methods, we can observe that relatively speaking the Canny method is less impacted by noise, and more likely to detect true weak edges when compared with the other methods.

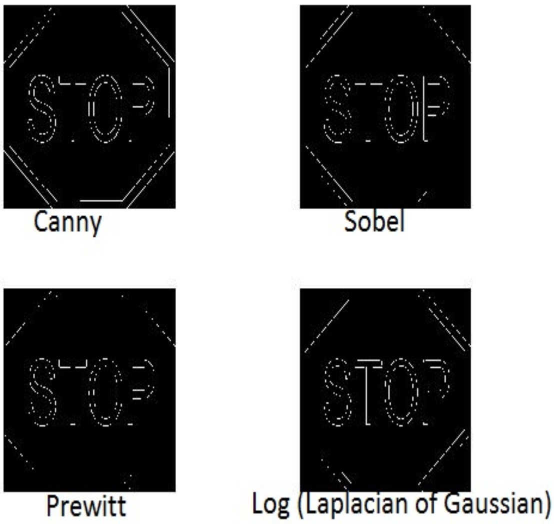


Figure 2 Example of an unacceptable low-resolution image

Line Detection

We used Hough Transform [15] to detect lines. With this transform, straight lines passing through edge points can be drawn, using a mapping from the Cartesian space to a parameter space. The advantage of using Hough Transform is that the pixels forming a line need not be contiguous with that of other pixels. Therefore, it is a useful tool for detecting lines with short breaks in them due to noise or partial occlusion by objects. The following steps have been used in detecting lines in an image [16].

- Finding all the edge points in the image using any edge detection method.
- Quantifying the (m, c) space into a two-dimensional matrix H with appropriate quantification levels.
- Initializing the matrix H to zero value.
- Considering each element of H matrix, H (mi, ci), which is found to correspond to an edge point is incremented by one. The result is a histogram or a vote matrix showing the frequency of the edge points corresponding to certain (m, c) values (i.e. points lying on a common line).
- A threshold is used in the histogram H to find only large valued elements. These elements correspond to a line in the original image.

Lane Detection

The lane detection process uses the lines identified in the image using the MATLAB Hough Transform [17], and uses reference boundary line inputs. The Hough Transform object finds Cartesian coordinates of lines that are described by rho and theta values. The boundary lines are the left and right vertical boundaries and the top and bottom horizontal boundaries of the reference image. The object outputs the row and column positions of the intersections between the lines and two of the reference image boundary lines.

Figure 3 explains the lane detection algorithm used in this project. The video is captured from camera at the start of the operations. The input image is then passed to the module that detects the lines for identifying the boundaries of the lane, using the Hough transform. These detected lines are then converted from polar to Cartesian space to differentiate right and left lane. Once the lane is found in the current frame, lines are drawn to track the lane in the next step. A polygon function is drawn for the lane to track the visibility of the lane. The next step is to calculate the centroid of the lane, and determine an offset value from the centre of the lane. The offset indicates the distance between the centre of the vehicle and the centre of the lane markings.

Obstacle Avoidance

Autonomous vehicles operating in different environment conditions are required to be capable of detecting obstacles and planning an obstacle free path. In our vehicle we used LIDAR to detect obstacles. LIDAR is an acronym of Light Detection and Ranging. It is an automatic remote sensing device that calculates distance to targets by emitting a laser light and measuring the timing of the reflected light.

The advantage of using LIDAR is it has high scanning speed and accuracy for the distance measurements. We used a *2-D* SICK LIDAR LMS 291 which has a wide angle range of 180 degrees at an angular resolution of 0.5 degree and a distance range of 80 meters. The LIDAR can also be configured to output in 1.0 degree resolution. Every time the LIDAR scans, the array of the 180 degrees scan data can be mapped onto their respective angles in the polar coordinate system (r, ϕ). The LIDAR can output up to 12 to 15 scans per second.

Configuration of Sensors on the Test Vehicle

The camera and LIDAR are mounted on a test vehicle, as shown in Figure 4, for experimenting with the developed lane following and obstacle detection technique. The setup of the camera is most important, as position of the camera affects the way it detects lanes on the road. The setup of LIDAR is also equally important. Both should be properly placed in the centre of the vehicle, for correct viewing of the lanes and detection of obstacles in front of the vehicle

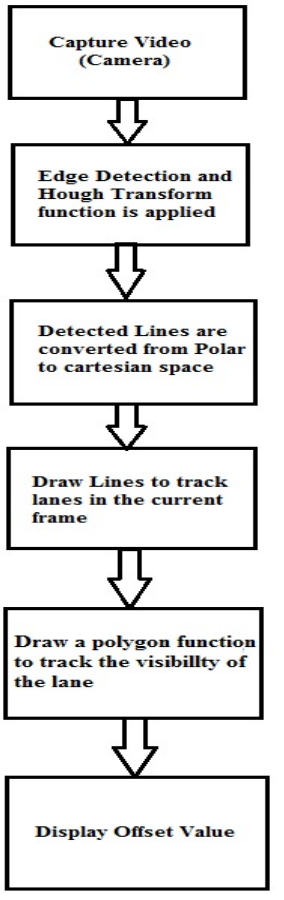


Figure 3: Lane Detection Algorithm Block Diagram

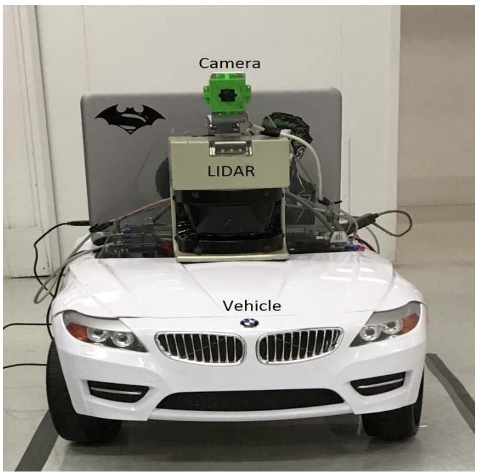


Figure 4: Setup of sensors on a test vehicle

The camera view is projecting downwards so that it is easy to obtain a good view of the lanes in the image frames. The camera centre and the LIDAR centre should be in line, so that we can easily overlay the objects detected by the LIDAR onto the camera lane view.

Parsing LIDAR Data

The LMS LIDAR unit is supplied with 24 Volts DC power. For accessing the scan data, the LMS LIDAR is equipped with a serial RS 232 interface. The default settings of the LIDAR are 9600 baud for communication, with angular range of 0 to 180 degrees at a resolution of 0.5 degrees. We change the baud rate from 9600 default to 38400 baud, but whenever the power is reset, the LMS LIDAR goes back to its default settings. So we needed to do initialization of the baud rate and the angular resolution in the control program, which helps increase the scanning speed.

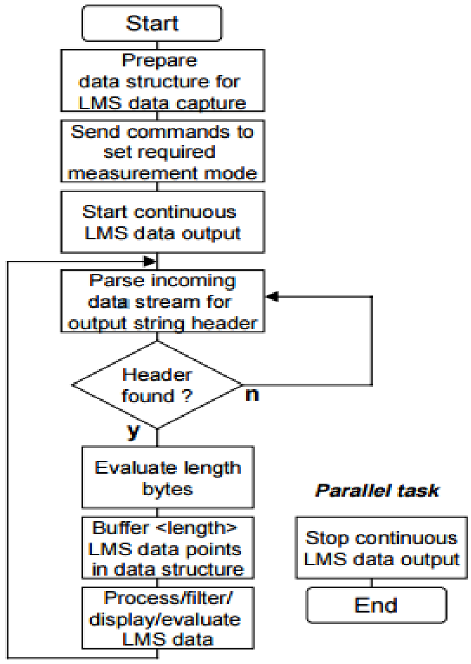


Figure 5: Interpreting/parsing the LIDAR scan data

The flowchart in Figure 5 explains the step by step procedure to parse the incoming data from the LIDAR and interpret it. The only way to change the settings in the SICK LIDAR [18] is by sending a set of command strings to it. We need to set the desired measurement mode of the SICK LIDAR to either 'cm' or 'mm' mode and change its baud rate, set the angular resolution, followed by starting a continuous data output instruction so that the device in return sends continuous data until a stop command string is sent to it. The next most important step would be to look for the packet header in the data which is received. The header is made up of 7 bytes of information and is received before the actual data. It contains information about the data being sent. By parsing the received bytes one can search for the header bytes and identify the beginning of the data.

In our case, this header is expected to be:

Hexadecimal Form:	02	80	6E	01	B0	B5	41
Decimal Form:	2	128	110	1	176	181	65

The incoming data from a scan is sent sequentially as the sensor scans through 180 degrees. In our case, since the sensor is set to scan 180° with resolution of 1 degree, the first data point sent will correspond to 0 degree, the next will correspond to 1 degree, the next to 2 degree, and so on. This means we should have a total of 181 data points. Each distance measurement is sent in the form of two bytes. The least significant byte is sent first, followed by the most significant byte. Since we're operating in mm mode, the unit for the measurements will be in mm.

Each distance measurement needs two bytes to store the data. Now after getting enough data bytes for the 180 degrees, we can take each pair of consecutive bytes and perform the following operation to calculate the distances in mm.

Distance (mm) = 1st Byte + (2nd Byte *256)

Since each measurement is represented by 2 bytes, and there a total of 181 data points, the incoming data stream should have a total of 362 bytes. This is important for choosing the buffer size and for parsing the data. When the application is done using the LIDAR sensor the program sends the Stop command string to the device, which then stops the LIDAR from sending scan data to the PC.

Obstacle Detection

Objects could be present anywhere in the environment, but it is important to ignore those points that would not disturb the path of the vehicle [19]. If these objects are directly in the path of the vehicle, they can be identified as an obstacle. The method of detecting the obstacle involves studying a certain degree-span of readings directly in front of the vehicle which are calculated based on a safe-distance and the minimum clearance required for the vehicle to pass safely.

In a scan we get 180 degrees but we limit the search to within 120 degrees which, we further divide into 6 parts with 20 degrees in each part. These regions are shown in Figure 6.

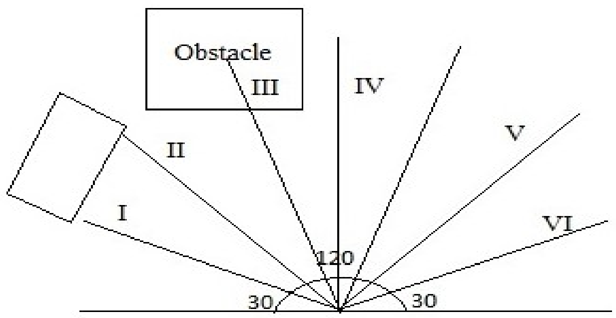


Figure 6: Defining zones for obstacle detection

When there is an obstacle in any part we look for the tail in the adjacent or the next part, through which we can estimate the size of the obstacle and can predict the turn needed to be made by the vehicle to avoid the obstacle. If no obstacle is detected we will simply continue following the lane.

The minimum distance to detect the obstacle is set to 0.3 m that is based on the turning radius of the vehicle as it maneuvers to avoid from hitting the obstacle when it makes a turn.

The final step is to properly combine the information from the lane and obstacle detection tasks into a common program. This part is still a work in progress. We are currently developing an intelligent path planning algorithm that combines the two together.

IV. EXPERIMENTAL RESULTS

In this section the paper presents results of experiments of the lane following and obstacle detection developed in the project. For the lane following experiment the input to the system is a stream of video captured by the single fisheye camera mounted on the vehicle. The input image was divided horizontally into two sub-frames. The main intention behind dividing the frame was to reduce the image processing time and to set a field of view to the lower part of the frame. This is needed to limit the processing to the close range areas where the lane marks are expected to be found as solid line or dashed line.

Sometimes the system encountered difficulty detecting lane markers on poor lighting conditions and shadows introduce problems to the detection algorithm. This impact is reduced by calibrating the image filters employed. The filtered image is then applied processed using automatic thresholding to convert it from grayscale to binary image that detects lane markers under various lighting conditions. Then the left and right lane markers are determined by using Hough Transform. This system detects lane markers by matching the current lane markers with the markers from previous video frame. Based on the classified lane markers and the offset value, the relative position of the vehicle in the lanes is calculated. This is used to calculate the offset which in turn helps determine whether to drive the car straight or to apply steering to follow turns in the lanes.

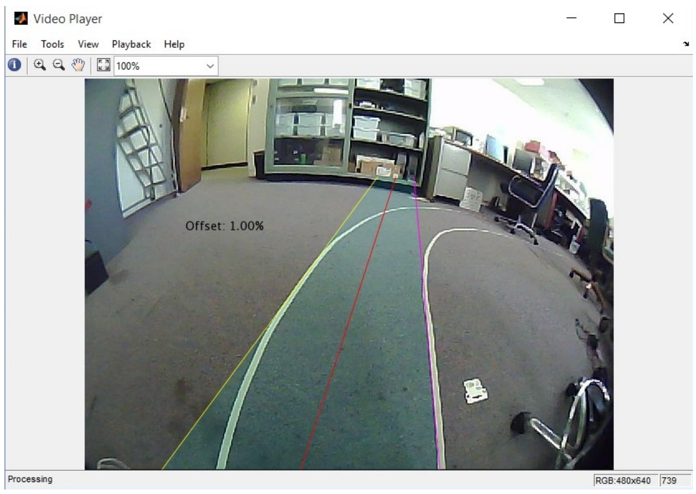
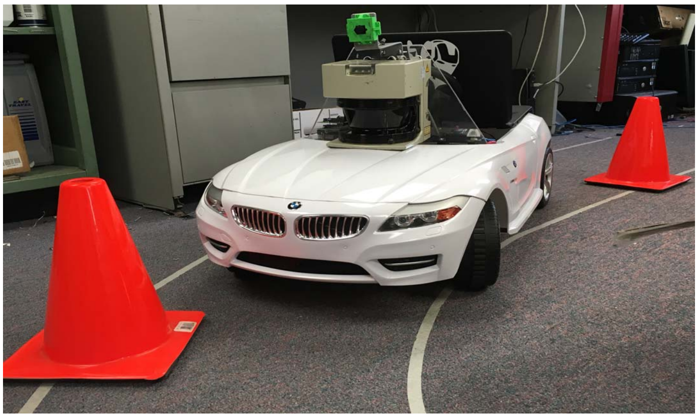
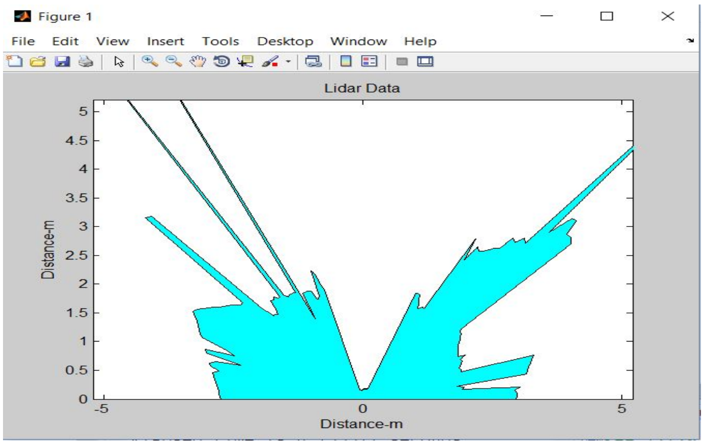


Figure 8: Experiment with the lane following algorithm

In obstacle detection, an incoming stream of LIDAR data is sent in a scan, where we get 180 degrees, although the optimum path is to be selected within 120 degrees which we further divide into 6 parts with 20 degrees in each part to define the position of the obstacle within the vehicles field of view. The actual control strategy for path planning and obstacle avoidance is currently being implemented.



(a)



(b)

Figure 9: (a) An obstacle encountered in front of the vehicle. (b) Shows laser scan graph plotted from LIDAR data.

V. CONCLUSIONS

This paper presented successful design and implementation a lane following and obstacle detection system for autonomous vehicle applications. The tools developed are implemented to processes the video frames from the camera to calculate the position of the lanes on the road. The vehicle control system then determines the motor speed and steering commands based on the lane positions or relative offset of the vehicle from the center of the lanes. The LIDAR scan data is used to determine if there is an obstacle is on the vehicle's path within its safety range. If it finds an obstacle it also determines its position by calculating the angular range the obstacle is located in.

This system was able to perform at an average speed of 10 to13 frames per second in MATLAB on Windows laptop with Core i5 processor and 8 GB of RAM, allowing a near real-time performance.

For future work we plan on improving the lane following algorithm's performance, and its reliability under varying lighting conditions and noise levels. We plan also to fully integrate the path planning and obstacle avoidance strategies using the LIDAR data.

REFERENCES

[1] W. Chen, L. Jian, and S. Kuo, "Video-based on-road driving safety system with lane detection and vehicle detection,n" 12th International Conference on ITS Telecommunications, Taiwan, 2012, pp. 537-541.

[2] S. Lee, H. Son, and K. Min, "Implementation of lane detection system using optimized Hough transform circuit," 2010 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp. 406-409.

[3] J. He, H. Rong, J. Gong, and W. Huang, "A Lane Detection Method for Lane Departure Warning System," International Conference on Optoelectronics and Image Processing (ICOIP), 2010, pp. 28-31.

[4] V. Gaikwad and S. Lokhande, "An Improved Lane Departure Method for Advanced Driver Assistance System," International Conference on Computing, Communication and Applications (ICCCA), 2012, pp. 1-5.

[5] "*GPS Basics,*" Web: <https://learn.sparkfun.com/tutorials/gps-basics?_ga=1.125849101.1285293773.1460411913>

[6] M. Haloi, "A novel pLSA based Traffic Signs Classification System", accepted in the Asia Pacific Conference on Multimedia and Broadcasting (APMediaCast), 2015.

[7] N. Srinivasa, Y. Chen, and C. Daniell, "A fusion system for real-time forward collision warning in automobiles," Proceedings IEEE Intelligent Transportation Systems, 2003, vol. 1, pp. 457-462.

[8] M. Aly, "Real time detection of lane markers in urban streets," IEEE Intelligent Vehicles Symposium, 2008, pp. 7-12.

[9] B. Benligiray, C. Topal, and C. Akinlar, "Video-based lane detection using a fast vanishing point estimation method," 2012 IEEE International Symposium on Multimedia, pp. 384-351.

[10] N. Arshad, K. Moon, S. Park, and J. Kim, "Lane detection with moving vehicle using color information." Proceedings of the World Congress on Engineering and Computer Science (WCECS), 2011.

[11] D. Hanwel and M. Mirmehdi, "Detection of lane departure on high-speed roads," International Conference on Pattern Recognition Applications and Methods (ICPRAM), 2012, pp. 529-536.

[12] Matlab, Web: <http://www.mathworks.com/help/releases/R2012b/vision/ref/2dfirfilter.html>

[13] Rafael C. Gonzalez and Richard E. Woods, Digital image processing, 3rd edition, Pearson, 2008.

[14] Matlab, Web: <http://www.mathworks.com/help/images/ref/edge.html?refresh=true >

[15] J. B. McDonald, "Application of the hough transform to lane detection and following on high speed roads", Proceedings of Irish Signals and Systems Conference, 2001.

[16] G. Hamarneh , K. Althoff, and R. Abu-Gharbieh, "Automatic line detection", Project Report for the Computer Vision Course, Chalmers University of Technology, September 1999.

[17] Matlab, "Computer Vision System Toolbox," Web: <http://www.mathworks.com/help/releases/R2012b/vision/ref/vision.houghlinesclass.html>

[18] "Interfacing with the Sick LMS-200," Drexel University, Web: <http://www.pages.drexel.edu/~kws23/tutorials/sick/sick.html>

[19] I. Ulrich and J. Borenstein, "VFH*: Local Obstacle Avoidance with Look-Ahead Verification," IEEE International Conference on Robotics and Automation (ICRA), 2000, vol.3, pp. 2505-2511.