Autonomous Vehicle Control Using Lidar and Camera with CAN Network

4P. Kantharaju, S. A. Hussain[2], S. Bethanbotla, R. Kalva, B. Shahian and S. Cetin[1]

1. *Mechanical and Industrial Engineering, University of Illinois at Chicago, USA scetin@uic.edu*

2. *Electrical and computer Engineering Sciences, University of Illinois at chicago, USA*

Abstract: An autonomous vehicle operates in a dynamically changing environment, where multiple sensors must work in a cooperative mode. In these scenarios reliability of the communication protocol carries a lot of importance in real time data transfer. In this paper, CAN communication is used to demonstrate sensor integration using a LIDAR and a camera. Also demonstrated is a novel method for object detection, obstacle avoidance and navigation of an autonomous RC vehicle.

Key words: CAN Network; Embedded System; LIDAR; Vision; Obstacle Avoidance

1 Introduction

In the advent of 21st century, as intelligent control research is being driven towards autonomous machines, self driving cars is next cornerstone of this decade. Self-driving vehicles are being developed and tested by many research and development (R&D) teams, original equipment manufactures (OEM) on a regular basis on US roads to launch a fully autonomous vehicles by the turn of this decade[1]. It is predicted that fully autonomous vehicle will replace day to day transportation in two decades[2]. As the race goes on for the producing first fully autonomous car, many driving features are being automated which include stop-n-go, adaptive cruise control, pedestrian detection, sign detection, self-parking, and active accident prevention usually referred to as Advanced Driver Assistance Systems (ADAS)[3]. Autonomous vehicle fundamentally relies on multiple sensors to measure and to detect its environment. Generally sensors used are vision, LIDAR, radar, ultrasonic and GPS[4-5]. Multiple sensors on the car communicate with each other to accurately predict the presence of objects that have direct impact on the driving behavior of the vehicle[5-6]. The most common of these are vision based camera systems and LIDAR.

A typical vision system contains a camera which is a combination of focusing lens and set of photodetectors and image processing unit which forms the main processor of the system. This processor, processes the pixels received according to algorithms to detect objects. LIDAR (Light Detection And Ranging) sends a beam of light pulse from a rotating mirror and detects the light thatis reflected back to the sensor from any object. The time taken to reflect back is used to measure the target distance since speed of light is known. LIDAR sensor scans the field of view with a finite spatial resolution and frequency. Using this detection pattern, the sensor then constructs a 3D map of objects around its world. 1D and 2D LIDAR are also used. In this project, we used a 2D LIDAR These sensors use infrared (IR), visible or ultraviolet (UV) spectrum of light for different applications[7].

The sensors rely on the different communication protocols for sending their data to embedded " master" controller for various actuation decisions. Most common of the such protocols used in embedded communications are Controller Area Network (CAN), Local Interconnected Network (LIN), FlexRay[8]. CAN was established by Robert Bosch GmbH, in 1983 and widely used in automotive since for communication. It was developed for networking controllers, sensors and actuator devices in automotive control.

Digital data is transmitted serially, bit by bit over the two wire cable in CAN network. It can share all information between all the nodes or intelligent modules connected to the bus[9-10]. Using appropriate WiFi modules CAN Bus communication can also be done between two or more disjointed electronic systems. It supports interrupt driven distributed real time control applications. Each node transmits data on the same physical medium by sharing the bus either through hardwire connection or through WiFi.

CAN has no master or slave, each message has unique identifier and priority. A message is broadcasted by one node at a given time and can be received by any node connected to the bus. The other nodes refer to theidentifier and ignores if not required this is called message filtering. Access to the network physical medium by different nodes is controlled by a "message priority" protocol. If multiple nodes try to transmit messages on the bus at the same time, the node which has the message with the highest priority wins the access. This is called bus access by arbitration. The arbitration is achieved based on taxing ID set in the message that is being transmitted. Lower the taxing ID higher the priority of the message.

A CAN module is the interface between physical layer and microcontroller. It is a special integrated circuit designed to support CAN protocol. There are two different types of CAN modules, basic CAN which has one transmit and receive buffer and requires microcontroller involvement for transmitting each message whereas full CAN has 16 buffers for transmitting and receiving messages along with individual identifiers for each message and doesn't involve microcontroller for transmitting messages as required operations are handled by CAN module. On a given CAN bus, all the nodes must meet the same physical layer hardware interface requirements. Each node must have the same baud rate, recessive/dominant state logic levels, output driver levels of the physical layer transceiver. CAN chips are programmable to set up the physical layer for different configuration. The CAN bus driver software runs on the microcontroller and provides an interface between the application software running on the microcontroller and the CAN controller so that high level application software does not have to deal with the details of CAN bus communication. The communication between the microcontroller and the CAN module can be programmed in either polled or interrupt driven mode. In general, all message communication activity is between the microcontroller and CAN module is programmed in interrupt-driven mode or polling mode.

This paper presents a setup which uses a Camera and a LIDAR over a CAN Bus to transmit sensor data to the controller to detect and avoid an obstacle in its path. The RC car has a electronically controlled steering and traction which can be controlled using the CAN messages. The microcontroller (Arduino UNO) communicates the logical decisions to RC Car over the CAN network using the CAN module.

2 Vehicle Hardware

2.1 RC Car

The test platform used in this project is a custom-made all wheel drive remote controlled (RC) car as shown in Fig 1. The traction power is transmitted from one DC motor to the wheel through axle and differential assembly. The steering mechanism is controlled by a servo motor. Power source for the DC motor and steering servo motor is a 12V, 3-Cell, Lipo battery. An ARM micro-processor establishes communication with controller using the CAN communication and converts the received CAN input signal to a PWM output signal.

2.1.1 Steering

A high torque 6V servo motor is used to facilitate the steering motion of the car, as shown in Fig 2. The PWM signal generated from the microprocessor controls the steering angle based the CAN message (control signal) provided. The message is send by the controller with a specific identifier and this microprocessor filter and reads the control input and controls the steering. This CAN message has the highest priority in the communication.

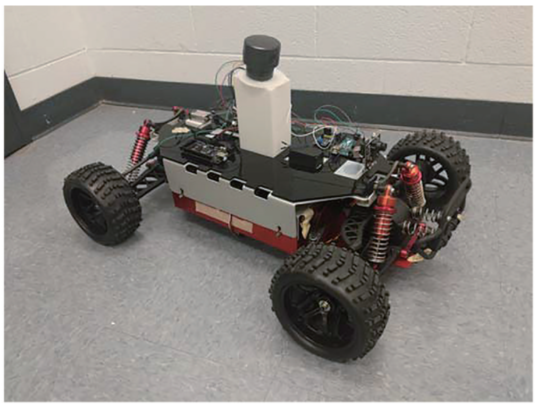

Fig. 1 RC Vehicle Test Platform

2.1.2 Traction

A 12V brush-less DC motor and drive (WP SC8 RTR 80A) is used to provide traction (drive power) which is connected to the axle and differential for transmitting power to all four wheels. The motor current is controlled using the WP-MAX8 RTR also known as Redcat racing motor drivers; uses a 12V 3 cell LiPo zippy compact battery. The Microcontroller controls the forward and backward motion as well as speed of the motor by sending messages with a specific CAN messageidentifier which is filtered by the microprocessor and convert it into a PWM signal.

2.2 Microcontroller

Arduino Uno with ATmega328P microcontroller with Flash memory of 32 KB, SRAM of 2 KB, EEPROM of 1 KB, Clock Speed of 16 MHz and Operating Voltage of 5V was used for receiving data from senor, Lidar and Vision and for controlling the motion of the Car, as shown in Fig 2. For providing CAN Bus communication capabilities following libraries were deployed: Arduino MCP2515 CAN library and CAN Bus Shield - MCP2515 interface library. For interrupt based message passing paradigm Timer1 library was deployed, which allowed for calling an interrupt service routine (ISR) at a specified sampling rate. Interrupts were used over polling method because Interrupts provide more flexibility regarding communication between all modules. Digital pin 10 was used as Chip select/ Slave select pin for Serial Peripheral Interface Bus (SPI) in microcontroller. This is used when there is necessity to connect several devices to the same set of input wires (e.g., CAN Bus), but retain the ability to send and receive data or commands to each device independently of the others on the bus.

2.3 Sensors

2.3.1 LIDAR

A two dimensional 360 degrees motorized laser scanner with a range of 40 meters, sampling rate of 1000 samples per second is used as the LIDAR sensor in this project. The distance is measured by a time of flight ranging method by transmitting a packet of micro pulses of light in a unique pattern. When this light bounce back from an obstacle to the receiving detector, a correlation algorithm is used to find the unique pattern from ambient noise. It uses an optical encoder to measure the angle of the rotating sensor head at which the obstacle range reading is recorded. It performs a calibration routine before it collects the data. It uses a serial protocol to communicate with the microcontroller and sends values of distance(range), angle and signal strength of sample. Software provided for LIDAR is the Sweep Visualizer. This tool is used to analyze the data from the device and parameters like sampling frequency, signal strength, motor speed can be adjusted. A map of obstacles detected is shown as a point cloud on the screen. The data can be analyzed using tools provided in the software. An Arduino library is provided to integrate LIDAR over the microprocessor.

2.3.2 Camera

Pixy (CMUcam5) vision sensor with standard M12 lens with a 75 degrees horizontal and 47 degrees vertical field of view captures 640x400 image frame every 1/50 of a second that detects objects using a hue-based color filtering algorithm is used as a vision sensor in this project. It calculates the hue and saturation of each RGB pixel and uses them as the primary parameters for filtering. It has an omni vision OV9715 image sensor with an NXP LPC4330, 204 MHz, dual core processor having 264K bytes RAM and 1 Megabyte flash memory. The processor sends the useful data to the microcontroller through a SPI output port about the detected objects. For this project built-in open source software PixyMon is used to teach the car about the object (which had only one color code) and was made to stop as soon as it detects the object.

2.3.3 CAN Communication

This system setup hardware provided by Kvaser Inc. was used to establish communication system between all the intelligent modules used in the project which includes microcontrollers used for sensors, steering and traction. Error debugging was done through CAN Kingdom software through which we can see the data flowing through the CAN Bus. ISO 11898 is set of standards defined by International Organization for Standardization generally built on CAN which is a basic low level standard that supports distributed real-time control and multiplexing for use within road vehicles with speed nearly 250 kbit/s or 500 kbit/s, but for this project only 125 kbit/s was sufficient. One of the many key features of this protocol is that message is broadcasted through the bus i.e. data is transmitted on the network without a specific destination. This permits the usage of data sent without requiring additional message requests. When a message must be directed to a particular device, a specific destination address can be included within the message identifier. This also allows future software revisions to easily accommodate new devices (address assignments).

3 Software

To establish CAN communication, Arduino IDE with MCP CAN add-ons are used. PixyMon, Sweep Visualizer and Kvaser CAN were used to calibrate sensors and test the internal CAN communication.

CAN communication was carried out with Arduino IDE software between the microcontrollers. PixyMon is an open source software provides an interactive GUI to set the color signature and various color codes for object detection. Once the signature is set in the flash memory of the Pixy, there is no need for retraining the Pixy to detect the object. Sweep Visualizer is the tool used to communicate with the LIDAR. This software is used to figure out the modified range of the LIDAR due to the angle of tilt. It is also used to decide the range of the angles in LIDAR's plane that car must care about while navigating through the environment i.e. all the data lying beyond some range of angles can be avoided as car will not collide with those obstacles, hence control action is not necessary for them. Based on these thresholds the control algorithm and sensor data is collection is done. Kvaser CAN Kingdom and Kvaser CAN Leaf are used to check the CAN Bus communication and also to determine the throttle & steering response of the system.

Arduino IDE is the primary software used to set up the CAN communication and also develop the control algorithm. The programming language used is C and various libraries such as TimerOne and MCP CAN are used.

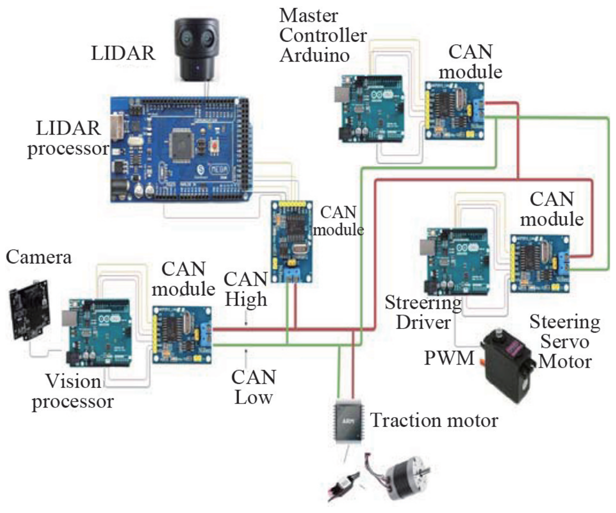

Fig. 2 Complete Schematic Representation of the System

4 Mathematical Formulation

Obstacle detection using LIDAR and Camera is the key aspect of the control problem. A grid of 10ft length and 4ft breath was constructed and camera data was collected at every 1ft length wise and breadth wise only five points were taken i.e in totality the grid size was 10x5. Then the collection of the width and height samples, and actual dimensions of the box was used to compute a scaling factor at every constant horizontal distance away from the car. Then a graph was plotted between scaling factor and horizontal distance between car and obstacle, which produced a nice empirical relation between distance and scaling factor. The empirical relation found out to be:

$$y = 7.6892 \cdot x \ 813\text{[illegible]}311 \qquad (1)$$

where y = Scaling Factor, x = Distance in feet between Car and Obstacle. Here Scaling Factor = (Pixy Width or Height)/(Actual Width or Height). This formula was extracted from the trend line of the graph shown in Fig 3, which depicts the variation of the Scaling factor versus distance from the obstacle to car. Above experiment also helps to pin points the relation between the lateral displacement of the object and Cartesian data from the camera. This information extraction helps the controller to decide which side the obstacle lies.

In the experiment setup used here, LIDAR was mounted with a tilt angle of 46.3 degrees and distance between the LIDAR and the camera (i.e. where pixy is mounted, which can assumed to be the front of the car) is 0.75ft. Due to the tilt angle present in the LIDAR mount it limits the range of distance sensing to 6.25ft from its position, this means from the car's front it will be only 5.5ft. This particular LIDAR cannot sense objects which are closer than 1ft from car's front end (again this means 1.75ft from LIDAR). If the LIDAR fails to detect the obstacle before 1.75ft, it indicates the systems has failed to detect obstacle and any control decision taken on the obstacle within this range will be a futile attempt and collision is inevitable. This problem also arises due to the fact that in the experiments the car moves at 1m/sec velocity. Also, the objects must be tall enough so that the light emitted from the LIDAR keeps on hitting the object.

To calculate the horizontal distance between car and obstacle, simple trigonometry is used to calculate it. Since the distance given by the LDIAR will always the hypotenuse of the right triangle formed by the distance between car and object, LIDAR mount height and LIDAR laser rays as sides, as shown Fig 4. So, using the tilt angle and hypotenuse of the right angle triangle, the distance between the car and obstacle will be the sine component of the hypotenuse. Formula showing the relation between the parameters is shown below:

$$D = H \cdot \sin(\theta) \qquad (2)$$

Here D = Distance of the obstacle from the car, H = Hypotenuse or LIDAR output and θ = Angle of tilt. From above, using equation 2, distance of the obstacle from the car is computed, and substituted in equation 1 to get the scaling factor, which in turn is used to compute the real life dimensions. These relations helps to fuse the data from sensors to give all the information about surroundings.

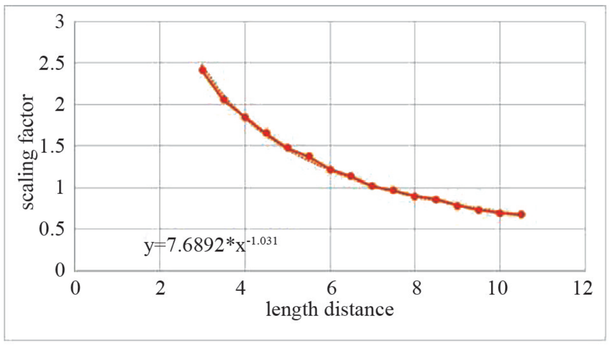

Fig. 3 In above graph X axis give distance object from the car and Y axis is the scaling factor. From the LIDAR the actual distance is obtained used to get the actual object dimensions.

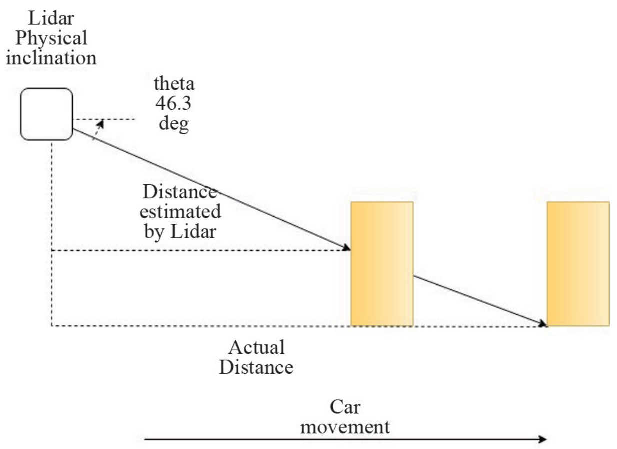

Fig. 4 Side View of the car approaching the obstacle & showing the right angle triangle formed between car and LIDAR. Also the LIDAR output and distance of object from the car.

5 Algorithm

The 2D LIDAR sensor is mounted at a 46.3 degrees tilt angle from the horizontal plane. It intersects the ground at 181 centimeters. The data must be filtered from entire 360 degrees to only data between +/- 35 degrees is taken into account as obstacle lying in too far left or right side need not be considered because the car can carry on with its motion without coming into contact. The obstacle is detected when the distance is recorded less than 181centimeters. The LIDAR is mounted at the center of the car at 14.5 inches from the base equidistant from the front wheels. The laser rays intersect the wheel center at an angle of 35 degrees from the zero-reference angle of the LIDAR (i.e. when laser from LIDAR coincides with car's mid section line) which is also happens to be the maximum steering angle of the car wheels as shown in Fig 6. An angle filter is implemented to filter the required angular data. It can only detect obstacles which are present in the range of 1ft to 5.5ft. Combining the above two constraints the sensing range of the LIDAR is reduced to a small part of a circle i.e. a annular sector, as shown in Fig 5. The edges of the obstacle are detected from the recorded data and sent to the controller.

According to the trained color code data, vision system will send the raw sensor data as soon as it detects that specific color code in the environment. Data sent contains the Cartesian coordinates, width and height of the bounding box of the object as seen from the camera's 2D frame. The data from vision sensor and LIDAR are processed in the microprocessor attached to it and the obstacle data is send to the controller by external interrupt driven data transmission while the data is received by the controller using timer interrupt.

Initial logic implemented in the car was based only upon the vision system. It was designed such a way that the as soon as the experiment vision sensor checks for any obstacle present in the surroundings. If it fails to find one then the car starts and moves until the car vision system finds a obstacle. Here as one can see there was no necessity of obstacle's real life dimensions, as car was requested to stop as soon as it sees one obstacle based on vision system.

Next, control logic was modified to do more complex maneuvers. Here the car relied on LIDAR data, so extraction of actual dimensions of the obstacle was possible. Also, from fused data of the both sensors the RC Car was able to decide the direction of the turn and intensity of the turn i.e. if the left or right turn is required and angle of the turn. The way sensor data was fused remained same for the different maneuvers. First LIDAR checks if there is a obstacle in car's critical range i.e. 181 centimeters away from the car's front wheel base and if the obstacle lies in the "danger" zone i.e. if there is a obstacle present in range of +/-35 degrees from zero reference of LIDAR. If not, the car starts moving in forward direction. As soon as the obstacle is detected when both conditions mentioned above are satisfied, thenfirst using the LIDAR output, linear distance from the car and object is calculated. Angle given by LIDAR tells the on which plane the object lies i.e. whether it lies on Right Hand Plane (RHP) or Left Hand Plane (LHP). Next using horizontal distance, scaling factor is computed to calculate the real life dimensions using the camera data. Also using the previously generated empirically data on camera, we can find the actual offset of the obstacle from the car' s center, this helps to figure out the angle of turn required to take. After deciding on the direction of turn, next to compute the magnitude of turn we use the obstacle' s real life dimensions. For these experiments we took sum of the half width of the obstacle, width of the car and 5 inches safety offset. This gives one side of the triangle formed using obstacle and car lying in car' s plane, another side is given by horizontal distance away from the car, then using arctan one can compute steering angle (refer the Fig 7).

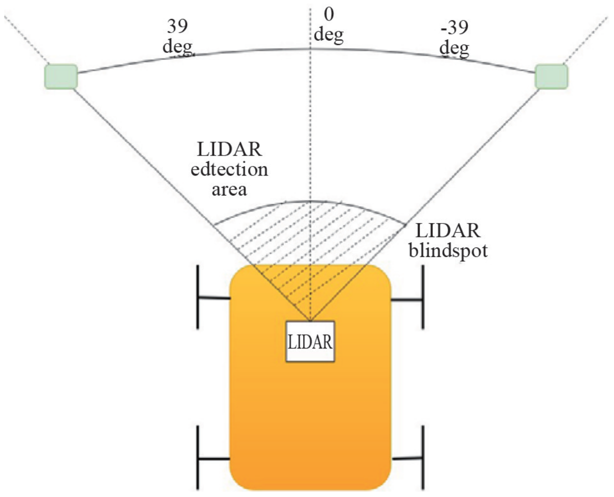

Fig. 5 Top View of the Car: showing the annular sector where LIDAR can detect obstacle.

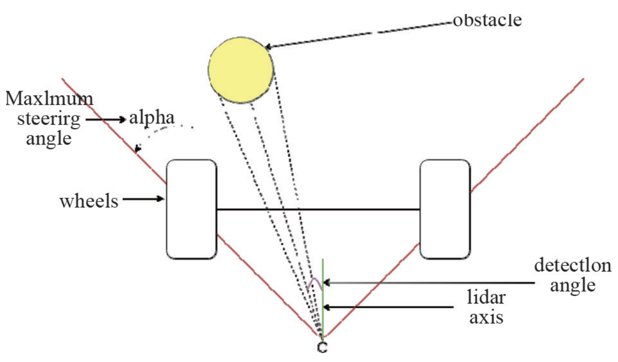

Fig. 6 Top View of the Car: showing the maximum steering angle of the car. If the obstacle was detected in this range by LIDAR then collision avoidance must be performed.

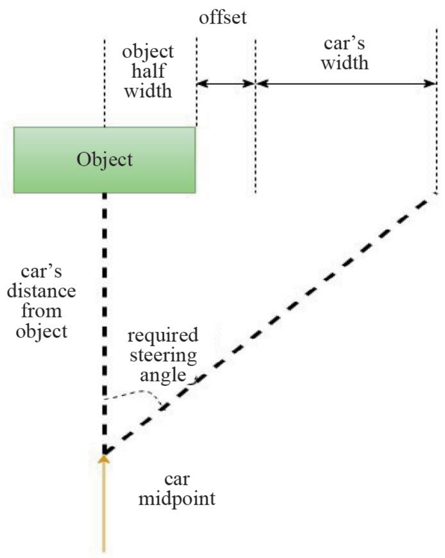

Fig. 7 Top View of the Car: showing the way to derive the steering angle

Above mentioned steps and known information were same for all experiments. Only thing that changed for different experiments was the control decision which in turn affected the car' s maneuver. Different maneuver' s performed on the car were:

· When obstacle is detected change car' s current lane and shift to new lane by taking appropriate turn.

· When obstacle is detected do a 90 degree turn in opposite direction to the obstacle' s current existing plane.

· When obstacle is detected change car' s current lane and shift to new lane by taking appropriate turn and then after traveling for few seconds in new lane come back to the old lane.

Using above control decisions, a final maneuver was attempted, which was to do a zig-zag avoidance of obstacle i.e. obstacles are kept in different lanes in regular intervals and car avoids them by going in zig-zag motion.

6 Results and Discussion

Pictures from a video are used to demonstrate the various actions done by the car autonomously.

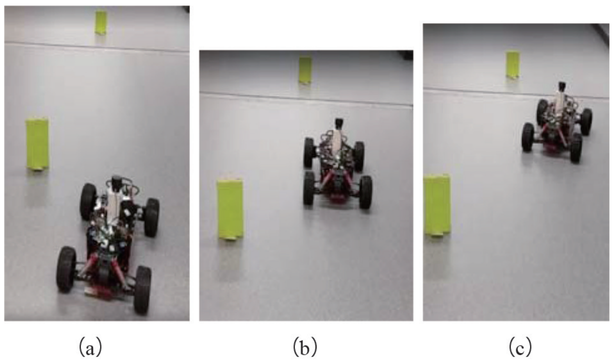

Fig. 8 Car recognizes the object in the lane and make right or left lane change.

In the Fig 8, the car recognized the object on the lane, calculated the angular position of the object also real dimensions of the object and adjusted the steering angle to avoid this obstacle maneuver.

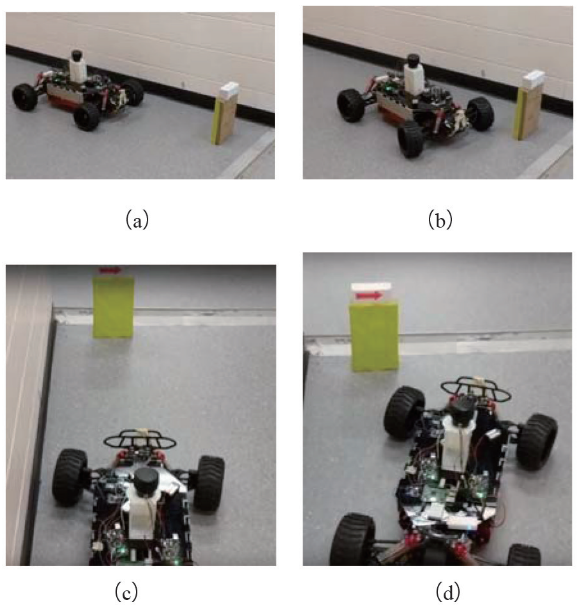

Fig. 9 Right Turn of the Car

The Fig 9 shows a hard-right steering motion achieved by the car. Here the algorithm was changed in such a way that as soon as big obstacle on the left side of the car was seen, it takes a hard right.

Similar to the Fig 9, the control algorithm was change to achieve hard left turn as shown in the Fig 10.

In the above Fig 11 the maneuver was achieved by using only the camera.

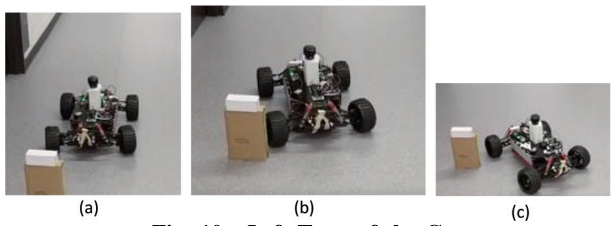

Fig. 10 Left Turn of the Car

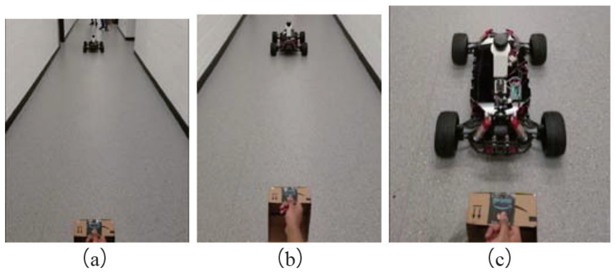

Fig. 11 All the above figures represent Car approaching the target and avoiding it by breaking at optimal distance

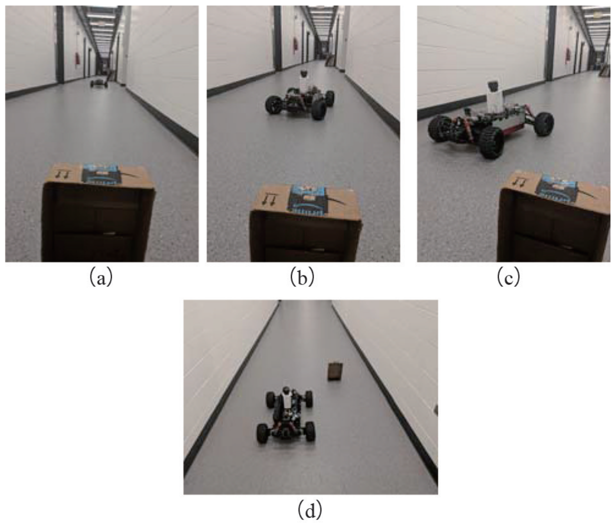

Fig. 12 Single Lane Change

Here Fig 12 shows the real life scenario of lane change as soon an object detection in the lane.

The above Fig 13 demonstrated a complex lane change motion by the car, by placing obstacle in two lanes alternatively forcing it to follow a zig-zag trajectory. Due to action of the LIDAR and Camera this path was obtained.

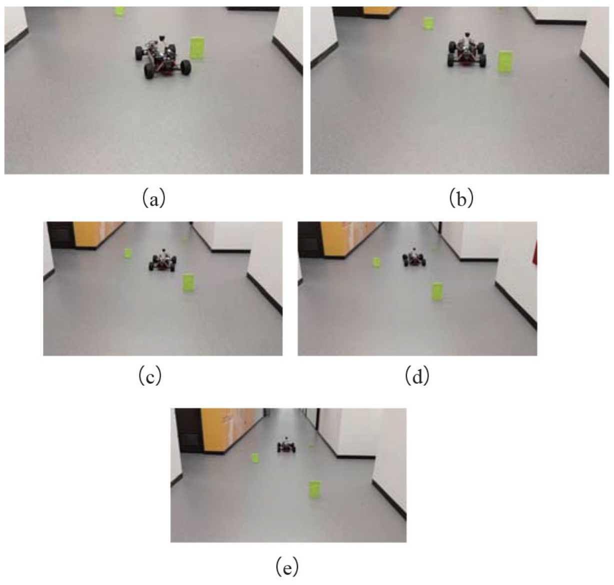

Fig. 13 Lane Change on Object Detection

7 Conclusion

Perception of the environment for detecting obstacles plays an important role in autonomous driving systems, especially when the environment varies dynamically. The sensors measuring and communicating data to the controller where the data from different sensors are combined and a control decisions are made. The data from each sensor is filtered in the embedded microprocessor attached makes it an intelligent unit and the processed data is send to the master controller. The communication of data between different sensors and actuators is the backbone in achieving the dynamic perception of environment. CAN communication has many advantages in this scenario. Using the CAN system actions such as lane change, object avoidance, right and left turn can be implemented. The modular nature of the CAN communication system can be used to add more sensors such as IMU and GPS which will help with the localization.

ACKNOWLEDGMENT

This project was supported by Kvaser Inc and Servotech Inc.

References

[1] Bansal, Prateek and Kara MKockelman (2017). "Forecasting Americans' long-term adoption of connected and autonomous vehicle technologies". In: *Transportation Research Part A: Policy and Practice* 95, pp. 49 – 63.

[2] Beiker, S (2014). "History and Status of Automated Driving in the United States". In: *Road Vehicle Automation*.

[3] Farsi, Mohammad, Karl Ratcliff and Manuel Barbosa (1999). "An overview of controller area network". In: *Computing & Control Engineering Journal* 10.3, pp. 113 – 120.

[4] Litman, Todd (2017). *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute

[5] Navet, Nicolas et al. (2005). "Trends in automotive communication systems". In: *Proceedings of the IEEE* 93.6, pp. 1204 – 1223.

[6] Schwarz, Brent (2010). "LIDAR: Mapping the world in 3D". In: *Nature Photonics* 4.7, p. 429.

[7] Stiller, Christoph, Fernando Puente León, and Marco Kruse (2011). "Information fusion for automotive applications -An overview". In: *Information Fusion* 12.4, pp. 244 – 252. ISSN: 15662535. DOI: 10.1016/j.inffus.2011. 03.005.URL: http://dx.doi.org/10.1016/j. inffus.2011.03.005.

[8] Wu, Fei et al. (2017). *Virtual Controller Area Network*.

Author Biographies

P Kantharaju, Currently pursuing Master's in mechanical engineering at University of Illinois at Chicago. His main research interest is Autonomous vehicle, Controls systems and Realtime control development. He previously completed his bachelors of Engineering in mechanical engineering from, Visvesvaraya technological University, Belgaum, India.
Email: pkanth3@uic.edu.

Syed A. Hussain, completed his Ph.D. in Electrical and Computer Engineering at University of Illinois at Chicago (UIC), Chicago, Illinois on August 2018. His research interest is in applied mechatronics engineering with special interest in autonomous vehicles engineering and embedded control. He also worked as a mechatronics intern for a summer at Servotech working on developing hardware-in-loop testing platform. Earlier to his current studies, he was Lecturer in King Fahd University of Petroleum and Mineral teaching controls and instrumentation from 2004. He was a consultant and co-investigator for various embedded control projects involving air-conditioner (a/c) load management, regulating in-rush a/c currents, detecting irregular heart beats like arrhythmia and some involving application of artificial intelligent techniques for crude oil well analysis. He obtained Master's degree from Systems Engineering, King Fahd University of Petroleum and Minerals (K.F.U.P.M.) in Controls and Instrumentation and has a Bachelor of Mechanical Engineering at Muffakham Jah College of Engineering (M.J.C.E.T.), Osmania University, Hyderabad, India.

Email: shussa50@uic.edu

Sabri Cetin, is a full Professor at University of Illinois at Chicago (UIC). He got his Ph.D. in Mechanical Engineering from Georgia Institute of Technology (1987) and worked at UIC since. He is the Founder and President of Servotech Inc, a small high-tech company specializing in embedded control software development. Previously, he obtained his Master of Science from Georgia Institute of Technology (1984) and bachelor's in aerospace engineering from Technical University of Istanbul (1982). He has received many honors during his tenure at UIC including the UIC faculty research award. He has filed five patents between 2001 and 2007. His major research interest includes mechatronics and control systems.

He is currently the director of the Mechatronics Laboratory and focuses on various aspects of motion control of mechanical systems including, DC servo motor control, ultra-precision motion control, electro-hydraulic servo control with earth moving equipment applications, adaptive self-learning real time control algorithms

Email: scetin@uic.edu