Applications of Image Processing and Real-Time embedded Systems in Autonomous Cars: A Short Review

Vidya Viswanathan *vidyakv@uw.edu*
Department of Electrical Engineering
University of Washington
Bothell, 98011, USA

Rania Hussein *rhussein@uw.edu*
Department of Electrical Engineering
University of Washington
Bothell, 98011, USA

Abstract

As many of the latest technologists have predicted, Self-driving autonomous cars are going to be the future in the transportation sector. Many of the billion dollar companies including Google, Uber, Apple, NVIDIA, and Tesla are pioneering in this field to invent fully autonomous vehicles. This paper presents a literature review on some of the important segments in an autonomous vehicle development arena which touches real time embedded systems applications. This paper surveyed research papers on the technologies used in autonomous vehicles which includes lane detection, traffic signal identification, and speed bump detection. The paper focuses on the significance of image processing and real time embedded systems in driving the automotive industry towards autonomy and high security pathways.

Keywords: Image Processing, Hough Transform, Canny Edge, Open CV, Polygonal Approximation, Computer Vision.

1. INTRODUCTION

Image processing is one of the main drivers of automation, security and safety related application of the electronic industry. Most image-processing technologies involve several steps like treat the image as a two dimensional signal and apply standard signal processing techniques to it. Images are also handled as 3D signals where the third dimension is the time or the z-axis. Highly efficient, low memory and reliable solutions can be achieved by utilizing Embedded Systems and Image processing to bring out the benefits of both for applications.

Google is one of the billion dollar companies who has demonstrated its own driverless car, a design that does away with all conventional controls including the steering wheel, and other astonishing technologies. In their driverless car, Google has not only included Image Processing, but also many other amazing technologies and one of the most important among them is Lidar, which stands for "Light Detection and Ranging". It consists of a cone or puck-shaped device that projects lasers which bounce off objects to create a high-resolution map of the environment in real time. In addition to helping driverless cars to "see", Lidar is used to create fast, accurate 3D scans of landscapes, buildings, cultural heritage sites and foliage. Some of the other technologies include Bumper Mounted Radar for collision avoidance, Aerial that reads precise geo-location, Ultrasonic sensors on rear wheels which detects and avoids obstacles, software which is programmed to interpret common road signs etc. Apart from these, there are altimeters, gyroscopes, and tachymeters that determine the very precise position of the car and offers highly accurate data for the car to operate safely. The synergistic combining of sensors is one of the most important factors in this autonomous car which includes the data gathered altogether by

these sensors are collated and interpreted by the car's CPU or in built software system to create a safe driving experience.

Apart from Google, many other companies like Tesla, Audi, Uber have also developed their own driverless cars and have tested potentially.

This paper concentrates on how Image processing can be used in vehicles to drive the automotive industry to completely autonomous and high security pathways. A real time embedded system environment is inevitable in an automotive application. Also, the scale of the industry is very high so the solutions should be cost efficient, fast and reliable. This paper intends to highlight these key points.

2. AUTONOMOUS VEHICLE CONTROL USING IMAGE PROCESSING

In autonomous vehicles, the driving commands from a human driver are replaced by a controller or a microcomputer system that generates these commands from the information it gets as it's input. Since this paper deals with the applications of image processing in autonomous control of a vehicle, the input given to the microcomputer system is the visual information obtained from a camera mounted on the vehicle. This section explains in detail, some of the important factors in Autonomous vehicles such as Lane Detection, Traffic Sign Detection, Speed Bump Detection, Steer By Wire System etc., which uses the processing of received image inputs and the algorithms used in them. Lane Detection represents a robust and real time detection of road lane markers using the concept of Hough transform in which the edge detection is implemented using the Canny edge detection technique and Spiking neural network technique. Traffic Sign Detection includes the recognition of road traffic signs in which, the concept of Polynomial approximation of digital curves is used in the detection module. Speed Bump Detection represents the detection and recognition of speed bumps present in the road which alerts the vehicle to control the speed automatically and Steer-By-Wire system represents an autonomous steering system using an Electronic Power Steering (EPS) module.

2.1 Lane Detection

Lane detection is one of the main part in the self-driving car algorithm development. On board cameras are kept in and around the cars to capture images of road and surrounding of the car in real time [1]. When the vehicle appears to deviate from the lane or vehicle safety distance is too small, it can timely alert the driver to avoid dangerous situations.

The basic concept of lane detection is that, from the image of the road, the on-board controller should understand the limits of the lane and should warn the driver when the vehicle is moving closer to the lanes. In an autonomous car, lane detection is important to keep the vehicle in the middle of the lane, at all-time, other than while changing lanes. Lane departure warning systems have already crawled into most of the high-end passenger cars currently in market.

A typical lane detection algorithm can be split into simple steps:

1. Select the ROI (Region of Interest)
2. Image Preprocessing (gray range/image noise subtraction)
3. Get the edge information from the image (edge detection)
4. Hough Transform (or other algorithms) to decide the lane markings

Step 1: Select the ROI

The images collected by the on-board camera are color images. Each pixel in the image is made up of R, G, and B three color components, which contains large amount of information. Processing these images directly makes the algorithm consume a lot of time. A better idea for this problem is to select the region of interest (ROI) from the original image by concentrating on just the region of the image that interests us namely the region where the lane lines are generally

present. Processing on only the ROI can greatly reduce the time of algorithm and improve the running speed. The original image is shown in Figure 1 the ROI region is shown in Figure 2.

FIGURE 1: Original Image [1].

FIGURE 2: ROI area [1].

Step 2: Image Preprocessing

Most of the road images have a lot of noise associated. So, before we do any image processing steps we need to remove those noises. This is typically done through Image Preprocessing. Image preprocessing includes grayscale conversion of color image, gray stretch, median filter to eliminate the image noise and other interference information. Gray stretch can increase the contrast between the lane and the road, which makes the lane lines more prominent. Equation (1) represents the function which is to be applied to an RGB image to convert it to Gray Scale.

$$L(x,y) = 0.21\ R(x,y) + 0.72G(x,y) + 0.07\ B(x,y) \quad (1)$$

Where
R - Red component of the image
G - Green component of the image
B - Blue component of the image
X,y - position of a pixel

Figure 3 shows the gray scaled conversion of the ROI region and Figure 4 shows the gray stretch image.

FIGURE 3: RGB to Gray conversion of Original Image [1].

FIGURE 4: Gray Stretch Image [1].

The method of image filtering includes the frequency domain filtering and the spatial domain filtering [2]. Spatial domain filtering is simpler and faster than the frequency domain filtering. Spatial domain filtering can remove the salt and pepper noise from the original image and preserve the edge details of the image. Its main principle is to use the middle value of every pixel in the neighborhood of one pixel instead of current pixel value. The image after median filtering is shown in Figure 5.

FIGURE 5: Median filtered image [1].

Step 3: Edge Detection

The next step is to perform edge detection on the output of the preprocessing. It is basically to detect the lines around the objects in the images. One of the common methods of edge detection is called Canny Edge Detection introduced by John F Canny, University of California, Berkeley in 1986. It basically uses multiple steps including Gaussian filters, intensity gradient changes to determine edges [3]. Figure 6 shows the Canny edge detected output of the grayscale converted image.

FIGURE 6: Canny Edge Detection Output image [1].

In recent researches, one of the main goals are to develop higher efficient edge detection algorithm for better detecting edges from varying image quality. For that purpose, an alternative approach to edge detection called Spiking Neural Network [1] is used, which is claimed to be a much efficient method than canny edge detector. Figure 7 shows the sparking neural network output of the gray scale converted image.

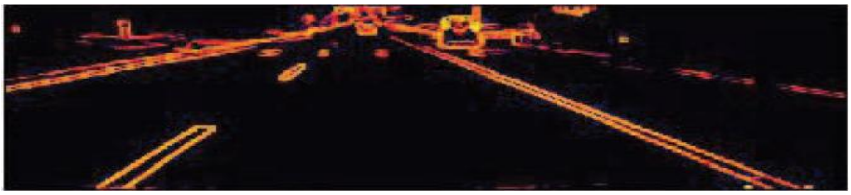

FIGURE 7: Spiking Neural Network Output Image [1].

Step 4: Hough Transform

Hough transform uses simple mathematical model of a straight line in a two-coordinate system, to predict straight lines or other simple shapes from an image. Equation (2) represents the basis for performing Hough transform.

$$r = x \cos \theta + y \sin \theta \qquad (2)$$

From the edge detection algorithm, we get a set of points which are considered part of the lane markings, Hough transform is used to generate a line through these points from the image. It connects points in the 2 dimensional polar coordinates of r and θ (theta), and generates multiple lines from each point. After generating all the different combination of lines of r and θ (theta) from each point, there will be one specific r and θ (theta) which intersects which is common to all the lines generated. This line will pass through all the points of interest. Thus, a line is generated. Figure 8 shows the lane detection output after performing Hough transform.

FIGURE 8: Lane detection output after Hough transform [1].

Hough transform is a widely-used method in the lane detection. It has very good suppression of noise performance and is not sensitive to the target which is partially occluded and covered in the image. But, because of the complexity of Hough transform, the computation speed is very slow, and the false detection rate is large. It cannot meet the real-time requirements accurately. The detection speed is improved by limiting the polar angle and the polar diameter in Hough transform space as described in Q. Wang [4]. Z.L.Jin [5] detected the lane lines by finding the best threshold of the image. Random sample consensus (RANSAC) algorithm is used to detect the lane lines in J.Guo [6], but it has high wrong rate. J.R.Cai [7] used the peak point inverse transform to extract line segments in Hough transforms space. All the methods mentioned above are modified in the process of Hough transform, but do not consider the appropriate processing methods in the image before applying Hough transform, to improve the detection accuracy and shorten the calculation time. X.Li [1] considered all these discrepancies and integrates a set of image processing techniques and spiking neural network to improve accuracy of the lane detection. Initially, the ROI is set on the image captured from the on-board camera. This can greatly reduce the running time of the algorithm and improving the detection speed; Secondly, a set of image preprocessing techniques is applied to the ROI region, such as transform from RGB image to grayscale, gray stretch and median filtering. Edge detection based on the spiking neural network with biological mechanism is used to extract efficient edge lines. At last, the Hough transform is used to detect the lane. The new ideas are that the ROI preprocessing is used to speed up the detection time and spiking neural network is used to obtain better edge lines to improve the detection accuracy.

2.2 Traffic Sign Detection

Traffic sign detection is key to semi and fully autonomous vehicles. The purpose of the technique is to identify the sign boards on the side of the road and identify what they imply and let the car computer know. Figure 9 shows the block diagram for the implementation criteria of the system. The system has two components, one is the preprocessing module, which is based on image processing techniques such as color segmentation, threshold technique, Gaussian filter, canny edge detection and contours detection [8]. The second is the Detection module based on Polynomial Approximation of digital curves technique applied on contours to correctly identify the signboard. It uses a similar technique of converting Image from RGB to Gray to Binary and then finding contours. Based on the shape of the sign board, the algorithm identifies what Traffic Signal it is.

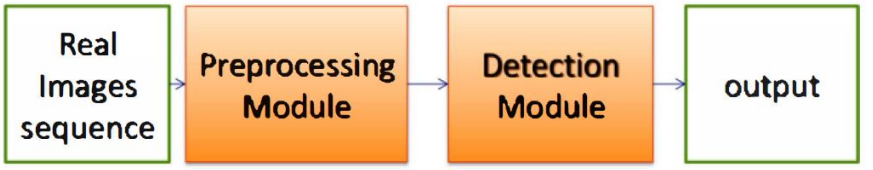

FIGURE 9: Traffic Sign Detection System [8].

The preprocessing module takes the RGB image and converts to HSV (Hue, saturation and Value) Color space. This can then be used for color segmentation. The images are checked for circular and triangular shapes of Blue and Red. The detection module then identifies the shape of sign board using the technique of Polygonal Approximation of digital curves, applied on contours, which is based on the algorithm of Ramer-Douglas-Peucker (RDP). RDP is an algorithm for reducing the number of points in a curve that is approximated by a series of points [9]. The purpose of the algorithm is, given a curve composed of line segments, to find a similar curve with fewer points. The algorithm defines 'dissimilar' based on the maximum distance between the original curve and the simplified curve. The simplified curve consists of a subset of the points that defined the original curve. An algorithm called Freeman Chain code, which is also behind Optical Character Recognition (OCR), is used to identify the letters, numbers and signs inside the sign board. Figure 10 shows the simplification of a piece-wise linear curve using RDP algorithm.

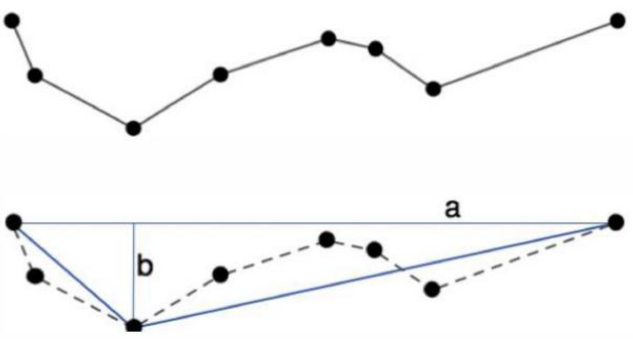

FIGURE 10: Simplifying a piece-wise linear curve using RDP algorithm [9].

The system was tested on a database containing more than 900 real images [10], with complex natural background, to test de validity of the system in different lighting condition (day, night, fog, rain, etc.) and in different location (city, town, nature, etc.). The results show the speed, robustness, and reliability of the system in these different situations. Tests show that the system implemented is very fast and good enough (detection rate 99.99%), even if it is dependent on size of the images [8]. The program may also detect some blobs which are not real traffic signs, but 99.98% of true traffic signs are detected. A Recognition System can be used later to distinguish these blobs separately. To more improve the system, the techniques of multithreading

can be used, which allows parallel execution of several programs at the same time. This will significantly improve the execution time which eventually leads to a more robust and efficient system for real-time implementation.

The new ideas proposed in this system are that the preprocessing module has processed the input images using image processing techniques, such as, threshold technique, gaussian filter, canny edge detection and contours, which leads to a more accurate and reliable implementation. The detection module to detect the traffic sign patterns, is based on Polygonal Approximation of digital curves, implemented by the algorithm of Ramer-Douglas-Peucker and the main reason to select this method is to reduce the computational cost in order to facilitate the real-time implementation.

OpenCV is an open source computer vision algorithm widely used in image processing applications [3]. Some of the widely used OpenCV function calls are defined below, which can be related to the applications described in this survey paper.

```
//Declarations
//Mat is 8 bit data type of image file in OpenCV can be jpg or //png
Mat input_image, output_image
Int intGradthesh_min, intGradthresh_max, aperture_size
// convert RGB image to gray
cvtColor(input_image, output_image, CV_BGR2GRAY);
//Canny Edge conversion - it needs a min and max threshold of
//intensity gradient to be provided to identify right edges and //discard wrong edges; aperture_size default is 3.
Canny(input_image, output_image, intGradthesh_min, intGradthresh_max, aperture_size)
```

2.3 Speed Bump Detection

Speed bump detection is one of the key action which the vehicle need to understand in-order to improve user comfort, as part of the Advanced driver assistance system (ADAS) in the vehicle [11].

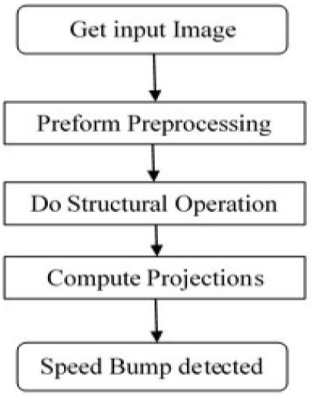

FIGURE 11: Flow Chart of Speed Bump Detection [11].

It is also a safety feature for the vehicle and the driver. A similar approach as the previous system is taken here as well. Figure 11 shows the flow chart used for the implementation criteria in this system. First the image is converted from RGB to gray using the Gray scale function. Then the Grayscale is converted to binary image. A morphological image processing is performed on this binary image [11]. A morphological processing is a set of operations that transform images per the size, shape, connectivity using the concept of set theory and other function. The process includes three steps, Opening, Area opening and Filling. After which a spread over Horizontal projection is

performed. Projection is a one-dimensional representation of an image computed along horizontal and vertical axis. It's the summing up of pixel value in rows and columns. The results show that this type of speed bump detection is ideal for all the bumps with zebra crossing pattern. Figure 12 shows the analysis and results of the implementation used in this system.

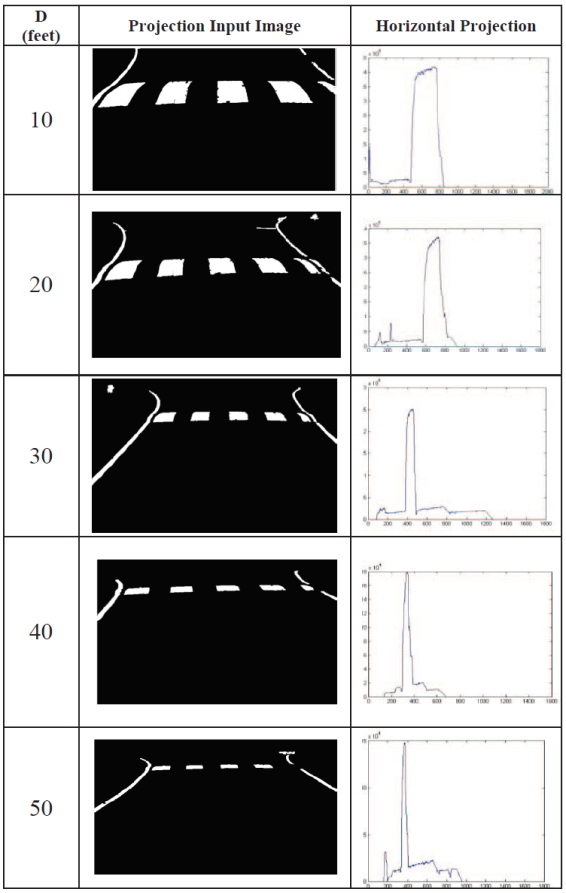

FIGURE 12: Horizontal Projection width increases as the vehicle approaches a speed bump [11].

M.Jain [12] and B.R [13] gives an overview for detection of speed bumps using accelerometer. M.P[14] focus on the sensing component, accelerometer, microphone, GSM radio, and/or GPS sensors. A.Danti [15] detected bump based on texture variation, Hough transform and local neighborhood information. B.R [13] proposed Wolverine - a non-intrusive method that uses sensors present on smart phones. M.A [16] describes a mobile sensing system for road irregularity detection using Android OS based smart-phones. K.Ganesan [17] detects pothole and speed bump using image processing, but they mainly focused on shadow cancelling using OpenCV. VP.Tonde [18] describes about a mobile sensing system for road irregularity detection using Android OS based smart phone sensors.

Most of the work mentioned above on speed bump detection are done with sensors, accelerometer and smart phone applications. The information about speed bump are collected and stored in the database initially and these offline data are used for testing the system. Thus it cannot be suitable for real-time scenarios whereas, the proposed system in W.Devapriya [11] is very much compatible with the real-time implementation. This methodology is applicable for the standalone cars without disturbing the network and can be embedded in higher end vehicles, especially self-driving cars. The average performance of the system considering only speed bump with proper marking is 85%. This methodology does not need any external hardware, sensors and smart phones. Without including the GPS network and disturbing the mobile battery,

detection of speed bumps can be made possible easily. Improvements to the existing system can be done by concentrating on detecting speed bump during night time, detecting speed bumps which do not follow any pattern or marking, training the speed bump detection system using neural network concept and distinguishing zebra crossing with the speed bumps.

2.4 Steer by Wire System

Automotive Steering has evolved a lot from its mechanical gear driven system to completely electronic system. The difference between these two are mentioned in Figure 13 below.

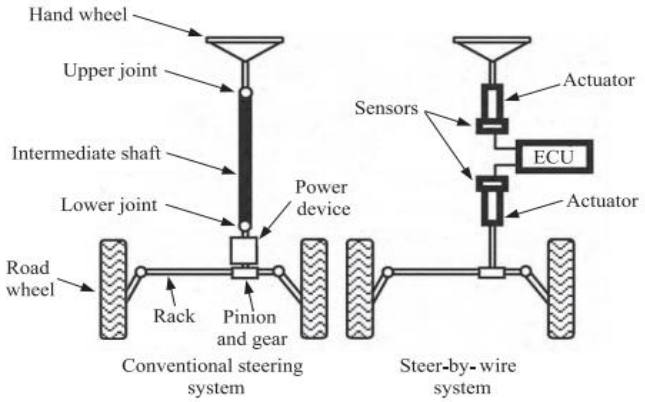

FIGURE 13: Difference between Conventional and Steer-by-wire steering system [19].

The steering wheel is connected to an angle sensor which detects the angle of rotation of the wheel with respect to its default position. This data is then sent to the ECU (Electronic Control Unit) as CAN (*Controller Area Network*) Signals. A description of CAN Communication protocol is mentioned in the following section. The electronic control unit uses this steering angle and then actuates a motor connected to the EPS module or the Electronic Power Steering module. The angle of rotation of the EPS motor is correlated to the angle of the rotation of the steering wheel, though they are not connected to each other through a steering column, but electronically, through the ECU [19]. This mechanism is called Steer-by-wire or Drive-by-wire system.

Security for automotive embedded systems has been studied initially. S.Checkoway [20] analyzed the external attack surface of a modern automobile. The authors discovered that remote exploitation is possible via a broad range of attack vectors such as mechanics tools, CD players, bluetooth, and cellular radio. M.L.Chavez [21] incorporated confidentiality service in CAN based on RC4. However, the authors did not consider fault-tolerance (FT) aspects or CAN message priorities while analyzing the security over CAN. Several earlier works explored dependability for automotive embedded systems. M.Baleani [22] discussed various FT architectures for automotive applications including lock-step dual processor architecture, loosely-synchronized dual processor architecture, and triple modular redundant architecture. Although, the authors compared various FT architectures' costs based on the area estimates, the study did not quantify the architectures' FT capabilities subject to real-time constraints. H.Schweppe [23] studied an active braking system assisted by vehicle-2-X communication. C.Wilwert [24] presented an approach for evaluating the temporal performance and behavioral reliability of an SBW system considering the delay variation introduced by network transmission errors. The authors evaluated the maximum tolerable system response time and the impact of this response time on QoS for a time division multiple access communication protocol. However, the work considered only communication transmission errors and not the computational errors in ECUs. Furthermore, the authors did not consider the system vulnerability to security attacks and did not analyze the scalability aspects. In A.Munir [19], the system considers security overhead while analyzing the system response time and analyzes the SBW system's response time under different load conditions and message priority assignments.

The advantage of this system is a closed loop control as well as fault detection and analysis, as steering is a highly safety critical mechanism in a car. Steer by wire also come handy in developing autonomous cars as the ECU can now control the Steering by tapping in to the CAN Bus and sending and receiving relevant angle info and motor turn. An SBW system eliminates the risk of steering column entering the cockpit in the event of a frontal crash and thus enhances safety. Since steering column is one of the heaviest components in the vehicle, removing the steering column reduces the weight of the vehicle and therefore lessens fuel consumption. Steer by wire systems enhance driver comfort by providing a variable steering ratio.

2.5 Controller Area Network

CAN is a serial communication standard with 3 layers: (1) Physical layer, (2) Data-Link layer, and (3) Application layer. These layers are mentioned in Figure 14 below. Each node is a ECU, for eg: Electronic Power Steering ECU, Body Control ECU etc.

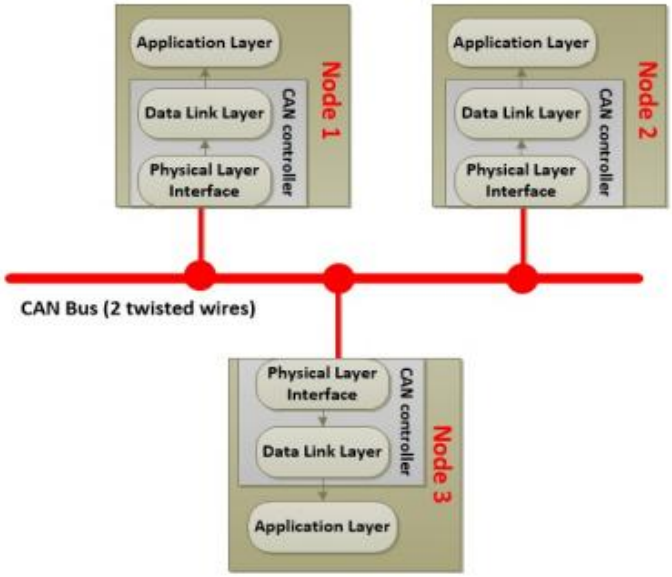

FIGURE 14: A Typical CAN Bus Nodes in a Car [25].

All the ECUs talk to each other through the CAN Communication protocol in the vehicle. A typical CAN message will have 8 bytes of data. Each byte can be allotted to specific info as per the developers wish. Each message has an ID assigned to it. All nodes can send and receive data. Also, CAN is a broadcasting protocol. So, it is not point to point, thus multiple ECUs can receive info from a single sender. In the steer by wire e.g., Node 1 is the Power Steering ECU, node two can be Vehicle Control ECU. The Power steering ECU sends Steering angle as 2 bits with message ID 0x180. The receiving ECU listens for his message ID, as soon as it receives it, it decodes the 2-bit info into steering angle and then activate the power steering motor [25]. The same information can be received from Instrument cluster ECU and can display angle info in the cluster and similarly the Body control unit can activate the turn signal automatically. Thus, it is a highly scalable network, used widely in automotive field for decades.

The important parameters like Bus Load, Quality of Service of the network etc. are important in analyzing the robustness of the network. As threats of remote hacking of cars have been made recently, there is great emphasize of increasing the efficiency of CAN network.

2.6 NVIDIA Real Time Hardware Solutions for Autonomous Cars

NVIDIA DRIVE™ PX 2 is the open AI car computing platform that enables automakers and their tier 1 suppliers to accelerate production of automated and autonomous vehicles. It scales from a palm-sized, energy efficient module for Auto-Cruise capabilities, to a powerful AI supercomputer

capable of autonomous driving. The platform uses two Parker processors and two Pascal architecture-based GPUs to power deep learning applications. Parker delivers up to 1.5 teraflops (a unit of computing speed equal to one million-million (10^12) floating-point operations per second) of performance for deep learning-based self-driving AI cockpit systems. It also has dual CAN (Controller Area Network) channels, an automotive communication protocol standard. Its main brain is four 64-bit ARM Cortex A57 CPUs.

The researchers trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands and this end-to-end approach proved surprisingly powerful. With minimum training data from humans the system learns to drive in traffic on local roads with or without lane markings and on highways. It also operates in areas with unclear visual guidance such as in parking lots and on unpaved roads. The system automatically learns internal representations of the necessary processing steps such as detecting useful road features with only the human steering angle as the training signal without providing any explicit training [26]. Compared to explicit decomposition of the problem, such as lane marking detection, path planning, and control, this end-to-end system optimizes all processing steps simultaneously and will eventually lead to better performance and smaller systems. Better performance will result because the internal components self-optimize to maximize overall system performance, instead of optimizing human-selected intermediate criteria.

This hardware setup can run a machine learning algorithm, which can take in real time of camera images and steering angle from a manual run car and process those data to create an algorithm, and then use the algorithm later to drive by itself. The three simple steps are mentioned with block diagrams below [26]. Most of the algorithms mentioned above comes in Step 2 below.

Step 1: Data collection

Figure 15 shows a simplified block diagram of the collection system for training data. Three cameras are mounted behind the windshield of the data-acquisition car. Time-stamped video from the cameras is captured simultaneously with the steering angle applied by the human driver. Training with data from only the human driver is not sufficient. The network must learn how to recover from mistakes [26]. Otherwise the car will slowly drift off the road. The training data is therefore augmented with additional images that show the car in different shifts from the center of the lane and rotations from the direction of the road.

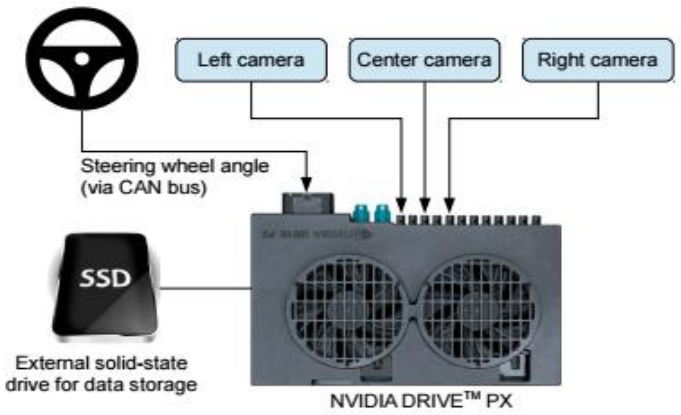

FIGURE 15: Data Collection using NVIDIA Drive PX in a manual car [26].

Step 2: Training the Neural Network

A block diagram of the training system is shown in Figure 16. Images are fed into a CNN which then computes a proposed steering command. The proposed command is compared to the

desired command for that image and the weights of the CNN are adjusted to bring the CNN output closer to the desired output. The weight adjustment is accomplished using back propagation.

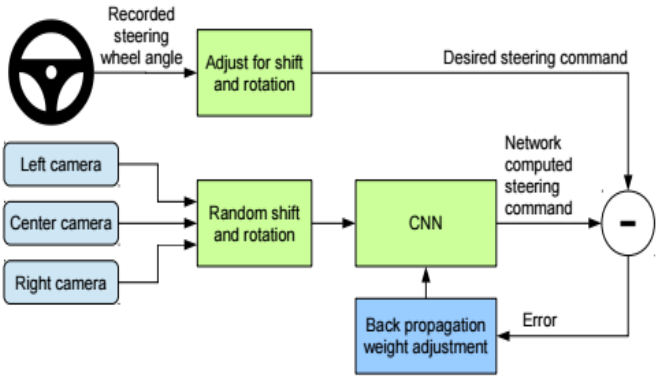

FIGURE 16: Training the NVIDIA Drive PX Artificial Neural Network Algorithm using the collected data [26].

Step 3: Autonomous Drive

Once trained, the network can generate steering from the video images of a single center camera. This configuration is shown in Figure 17.

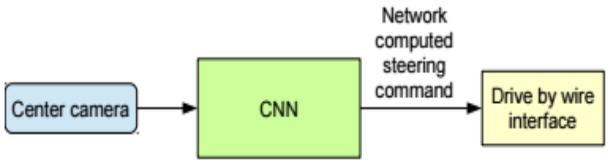

FIGURE 17: DRIVE PX driving autonomously by analyzing real time data using the trained Algorithm [26].

Training data was collected by driving on a wide variety of roads and in a diverse set of lighting and weather conditions. Most road data were collected in central New Jersey, although highway data was also collected from Illinois, Michigan, Pennsylvania, and New York. Other road types include two-lane roads (with and without lane markings), residential roads with parked cars, tunnels, and unpaved roads. Data was collected in clear, cloudy, foggy, snowy, and rainy weather, both day and night. In some instances, the sun was low in the sky, resulting in glare reflecting from the road surface and scattering from the windshield. The system has no dependencies on any vehicle make or model [26]. Drivers were encouraged to maintain full attentiveness, but otherwise drive as they usually do.

Evaluation of networks is done in two steps, first in simulation, and then in on-road tests. In simulation, the networks provide steering commands in their simulator to an ensemble of prerecorded test routes that correspond to about a total of three hours and 100 miles of driving in Monmouth County, NJ. The test data was taken in diverse lighting and weather conditions and includes highways, local roads, and residential streets. In simulation, what percentage of the time the network could drive the car (autonomy) is estimated. The metric is determined by counting simulated human interventions [26]. These interventions occur when the simulated vehicle departs from the center line by more than one meter, under the assumption that, in real life, an actual intervention would require a total of six seconds: this is the time required for a human to retake control of the vehicle, re-center it, and then restart the self-steering mode. Equation (3) represents formula for calculating percentage autonomy.

$$\text{autonomy} = \left(1 - \frac{(\text{number of interventions}).6 \text{ seconds}}{\text{elapsed time [in seconds]}}\right).100 \qquad (3)$$

After a trained network, has demonstrated good performance in the simulator, the network is loaded on the DRIVETM PX in the test car and taken out for a road test. For these tests, performance is measured as the fraction of time during which the car performs autonomous steering. This time excludes lane changes and turns from one road to another [26]. For a typical drive in Monmouth County NJ from Holmdel to Atlantic Highlands, the system was autonomous approximately 98% of the time.

3. FUTURE SCOPE

X.Li et al [1] experimentally showed that Spiking Neural network based edge detection is more efficient than Canny edge detection method in the case of lane marking identification on roads. The future work in this field would be identifying much more robust edge detection algorithms with less computational time. A.Salhi [2] discusses the method of traffic sign identification using Polygonal Approximation of digital curves technic applied on contours. The results are also presented where real time traffic signs are classified and identified using this method. The future work in this field would be analyzing other algorithms in traffic sign detection and figure out the most optimum and efficient methods. Also, the samples traffic signs are limited in this paper. The system should be able to scale the samples to include all the diverse varieties of traffic signs and identify it precisely. W.Devapriya [4] discusses Speed bump detection using Horizontal projection. The test results show that it could identify speed bumps with zebra crossing using this method. The future work in this field would be identify all varieties of speed bumps with different markings and no-markings. A.Munir [5] talks about Steer by wire systems and its security features. It also exhaustively talks about Controller Area Network and how it is impacted when its network parameters like Quality of Service Bus Load varies. The paper also defines fault tolerance, multithreading and error detection of the CAN Bus network in a modern car. M.Bojarski [7] shows the industrial case study done by NVIDIA to develop a self-driving car based on real time embedded systems. It shows the systematic approach to be taken while using cameras as sensors and Artificial Neural Network to teach the ECU to learn from the camera images and the steering angle information from the controller. It is an exciting field of conglomeration of all the latest technologies in automotive, image processing, computer vision, neural networks and embedded systems.

4. CONCLUSION

Various image processing techniques have been presented through these papers which have high end real time applications in the day to day life. The important take away terms are Computer vision techniques, Noise reduction and Content extraction. Various algorithms and filters are being used to achieve high efficiency data extraction from images. After evaluating the end results of the papers analyzed, it can be concluded that, out of the several algorithms reviewed in the literature survey, 75% were found success in real time in embedded systems. Real time environment functionalities push the realm of embedded controllers to be used to execute these features. Software development environment OpenCV has also been discussed in these papers. Also, there are small abstract of how steer by wire system and CAN communication protocol is involved in an automotive application. In the end a case study done by NVIDIA to develop an autonomous car using these mentioned technologies is discussed.

5. REFERENCES

[1] X.Li, Q.Wu, Y.Kou, L.Hou, H.Yang, ‘‘Lane Detection Based on Spiking Neural Network and Hough Transform’’ 2015 8th International Congress on Image and Signal Processing (CISP 2015)

[2] Y. Yuan, G. L. Zhang, “Improved Median Filtering Method,” Journal of Chengdu University of Technology,Vol. 31, pp.126-129,2013

[3] Canny J, “A Computational Approach To Edge Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986

[4] Q. Wang, “An Improved Lane Detection Algorithm Based on Hough Transform,” Computer and Digital Engineering,Vol. 42, pp.2164- 2167,2014.

[5] Z. L. Jin, G. M. Hu, H. J. Zhang, G. H. Man, “Lane Detection Method Based on Optimal Threshold and Hough Transform,” Instrumentation and Measurement,Vol. 28, pp.88-91,2009.

[6] J. Guo, Z. H. Wei, D. Q. Miao, “Lane Detection Method Based on Improved RANSAC Algorithm,” 2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems,pp.285-288,2015

[7] J. R. Cai, J. W. Zhao, “Camera Calibration of Binocular Stereo Vision System,” Journal of Jiangsu University,Vol. 27, pp.6-9,2006.

[8] A. Salhi, B. Minaoui and M. Fakir, "Robust Automatic Traffic Signs Detection using fast polygonal approximation of digital curves”, Multimedia Computing and Systems (ICMCS), 2014 International Conference on, Marrakech, 2014, pp. 433-437.

[9] Urs Ramer, "An iterative procedure for the polygonal approximation of plane curves", Computer Graphics and Image Processing, 1(3), 244-256 (1972).

[10] Dataset "The German Traffic Sign Detection Benchmark", http://benchmark.ini.rub.del?section=gtsdb&subsection=dataset 2013.

[11] W. Devapriya, C. N. K. Babu and T. Srihari, "Advance Driver Assistance System (ADAS) - Speed bump detection," 2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Madurai, 2015, pp. 1-6.

[12] M.Jain, A.P.Singh, S.Bali, S.Kaul, “Speed-Breaker EarlyWarning System”.

[13] B.R, N.Vankadhara, B.Raman and P.Kulkarni, “ Traffic and road condition estimation using smartphone sensors” in Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on (jan. 2012), pp. 1 –6.

[14] M.P, P.V. and R.R.; Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: 6th ACM conference on Embedded network sensor systems, SenSys ’08, pp 323– 336. ACM, New York, NY, USA (2008).

[15] Dr. A.Danti, J.Y. Kulkarni, Dr. P.S.Hiremath , “A Technique for Bump Detection in Indian Road Images Using Color Segmentation and Knowledge Base Object Detection”, International Journal of Scientific & Engineering Research, Volume 4, Issue 8, August 2013 ISSN 2229-5518.

[16] M.A, S.G, Z.R, K.G, And S.L, “ Real time pothole detection using android smartphones with accelerometers” Distributed Computing in Sensor Systems and Workshops, International Conference on (2011), 1–6.

[17] K.Ganesan, R.Natesan, A.Raj, N.Sekar , “An Image Processing Approach to Detect Obstacles on Road,” SAE Technical Paper 2015.

[18] V. P. Tonde, A.Jadhav, S.Shinde, A.Dhoka, S.Bablade “ Road Quality and Ghats Complexity analysis using Android sensors” International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 3, March 2015.

[19] A. Munir and F. Koushanfar, "Design and performance analysis of secure and dependable cybercars: A steer-by-wire case study," 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2016, pp. 1066-1073.

[20] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, “Comprehensive Experimental Analyses of Automotive Attack Surfaces,” in Proc. of the 20th USENIX conference on Security (SEC), August 2011.

[21] M. L. Chavez, C. H. Rosete, and F. R. Henr´ıquez, “Achieving Confidentiality Security Service for CAN,” in Proc. of IEEE International Conference on Electronics, Communications, and Computers (CONIELECOMP), March 2005.

[22] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, “Fault-Tolerant Platforms for Automotive Safety-Critical Applications,” in Proc. of ACM CASES, October-November 2003.

[23] H. Schweppe, T. Gendrullis, M. S. Idress, Y. Roudier, B. Weyl, and M. Wolf, “Securing Car2X Applications with Effective Hardware- Software Co-Design for Vehicular On-Board Networks,” in Proc. Of Joint VDI/VW Automotive Security Conference, Berlin, Germany, October 2011.

[24] C. Wilwert, Y.-Q. Song, F. Simonot-Lion, Loria-Trio, and T. Cl´ement, “Evaluating Quality of Service and Behavioral Reliability of Steer-by-Wire Systems,” in Proc. of IEEE ETFA,2003.

[25] A Rezaei, “An introduction to CAN (HEV Enterprise)”, CAN Tutorials.

[26] M Bojarski “End to End Learning for Self-Driving Cars” https://arxiv.org › cs, 2016.