An Efficient and Scalable Simulation Model for Autonomous Vehicles with Economical Hardware

Muhammad Sajjad, Muhammad Irfan, Khan Muhammad, *Member, IEEE*, Javier Del Ser, *Senior Member, IEEE*, Javier Sanchez-Medina, *Member, IEEE*, Sergey Andreev, *Senior Member*, Weiping Ding, *Senior Member, IEEE*, Jong Weon Lee

***Abstract*—Autonomous vehicles rely on sophisticated hardware and software technologies for acquiring holistic awareness of their immediate surroundings. Deep learning methods have effectively equipped modern self-driving cars with high levels of such awareness. However, their application requires high-end computational hardware, which makes utilization infeasible for the legacy vehicles that constitute most of today's automotive industry. Hence, it becomes inherently challenging to achieve high performance while at the same time maintaining adequate computational complexity. In this paper, a monocular vision and scalar sensor-based model car is designed and implemented to accomplish autonomous driving on a specified track by employing a lightweight deep learning model. It can identify various traffic signs based on a vision sensor as well as avoid obstacles by using an ultrasonic sensor. The developed car utilizes a single Raspberry Pi as its computational unit. In addition, our work investigates the behavior of economical hardware used to deploy deep learning models. In particular, we herein propose a novel, computationally efficient, and cost-effective approach. The proposed system can serve as a platform to facilitate the development of economical technologies for autonomous vehicles that can be used as part of intelligent transportation or advanced driver assistance systems. The experimental results indicate that this model can achieve real-time response on a resource-constrained device without significant overheads, thus making it a suitable candidate for autonomous driving in current intelligent transportation systems.**

Index Terms—Autonomous Driving, Raspberry Pi, Scalar-Visual Sensor, Intelligent Transportation Systems

I. INTRODUCTION

A vehicle capable of perceiving its surrounding environment and driving by itself safely without human intervention is known as an autonomous vehicle (also referred to as self-driving, driverless, or robotic vehicle) [1, 2]. Autonomous cars are constantly making headline news over the last few years. Different manufacturing companies and startups are targeting to develop safer, more responsive, and reliable cars for consumers of the next generation [3]. There is a growing competition among the biggest car manufacturing companies, each making their own version of a self-driving car. Companies like Google, Apple, Honda, Porsche, and Tesla have also established labs for developing self-driving cars. Baidu [4], which is a Chinese web services corporation, has also focused their attention on improving different factors involved in self-driving cars. Other companies and research labs are also working on individual layers involved in autonomous vehicles, such as sensor, communication, operating system, infotainment system, and computational hardware to enhance their performance. As autonomous vehicles rely on several capabilities, these individual factors can be further fused in them to reliably resolve different challenges in the field of self-driving cars.

More than 1.25 million people die in car accidents around the globe each year. According to a report of the World Health Organization [5], over 50 million people suffer from non-fatal injuries, while many acquire a disability. Car crashes cause extensive financial losses to individuals, their relatives, and the nation. These losses include the cost of treatment, time taken off from jobs, and effort to care for the injured. The cause of these road crashes and accidents is distracted driving, which takes lives of innocent people. Considering these losses of precious human beings, a system is needed, which is totally free of human intervention or partially assist humans to minimize these fatalities, thus advancing autonomous driving industry.

Researchers from different parts of the globe are contributing to different aspects of autonomous vehicles [6-11]. To motivate research community toward autonomous driving technology, the DEFENSE Advanced Research Projects Agency (DARPA) arranged the Grand and Urban challenge competitions in the USA [12, 13]. The purpose of the challenge was the development of autonomous vehicles that can traverse off-road terrain by themselves [14]. Urban challenge competitors focused on improvement of autonomous vehicles with urban driving technology. As a result of these competitions, several IT companies and automakers including General Motors, Volkswagen, Google, and Toyota have invested into commercializing the concept of autonomous cars. Similarly, another competition in autonomous vehicles was held by the Hyundai Motor Group in the years of 2010 and 2012 in South Korea for establishing the foundation for autonomous driving technology.

Manuscript received August 15, 2018; Accepted: January 24, 2020, Published: XXXX. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No.2016R1A2B4011712), and sponsored by Qing Lan Project of Jiangsu Province, China as well as RADIANT project, Academy of Finland. (Corresponding author: Khan Muhammad)

Muhammad Sajjad is with Digital Image Processing Laboratory, Islamia College Peshawar, 25000, Pakistan (Email: Muhammad.sajjad@icp.edu.pk)

Muhammad Irfan, Khan Muhammad, and Jong Weon Lee are with Department of Software, Sejong University, Seoul 143-747, Republic of Korea (Email: irfantahir301@gmail.com, khan.muhammad@ieee.org, jwlee@sejong.ac.kr)

Javier Del Ser is with TECNALIA Research and Innovation and University of the Basque Country, Spain (Email: javier.delser@tecnalia.com)

J. Sánchez-Medina is with the Centro de Innovación para la Sociedad de la Información, University of Las Palmas de Gran Canaria, 35001 Las Palmas, Spain (e-mail: javier.sanchez.medina@ieee.org)

S. Andreev is with Tampere University, 33720 Tampere, Finland (e-mail: sergey.andreev@tuni.fi)

Weiping Ding is with the School of Information Science and Technology, Nantong University, Nantong 226019, China, (e-mail: dwp9988@163.com).

IntelliDrive, also known as Connected Vehicle (CV), enables [15] double-way wireless communication for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. Within the CV environment, vehicles with communication devices and roadside infrastructure share the previously exclusive traffic including the vehicle maneuvers, trajectories, and origins/destinations. Hence, the environment of CV will allow for better control of urban intersections cooperatively for other vehicles and infrastructure. This CV environment for collaboration among vehicles has attracted significant attention because of its potential benefits. A prominent development of cooperation among the CV vehicles is a Cooperative Adaptive Cruise Control system [16, 17] intended to optimally operate vehicle tactics in various situations. Furthermore, the vision of a Cooperative Vehicle Intersection Control (CVIC) system is that an intersection controller and a vehicle may work together to expand traffic maneuvers for fully automated cars.

In this paper, we outline an efficient and economical solution for autonomous driving in the form of a lightweight deep learning model running over a resource-constrained device in real time. The model car is equipped with scalar and vision sensors using Raspberry Pi as ground processing unit for autonomous driving on a pre-defined track. The proposed solution can be used as part of intelligent transportation or advance driver assistance system to improve traffic management. A Raspberry Pi camera sensor is utilized for capturing a video stream in real time. Preprocessing is applied to the data collected from the vision and scalar sensors. A computationally efficient deep neural network is then trained for making decisions to go straight, stop, and turn left or right autonomously. Ultrasonic sensors are needed for detecting and avoiding obstacles along the path of the car. The following are the major contributions of this work:

1. A Raspberry Pi based framework for a self-driving model car is proposed, which can handle four tasks:
    a. Self-driving car model moves on a pre-defined track autonomously, being capable of driving in three directions: straight, left, or right.
    b. The model car detects and recognizes various traffic signs and takes action accordingly.
    c. The distance to a traffic sign is calculated by using a vision sensor. Obstacles are detected by processing ultrasonic data and are then avoided by the self-driving model car.
    d. Different HAAR Cascade based classifiers for the traffic signs and a deep learning model are trained for the track; they are loaded by Raspberry Pi in real-time.
2. Raspberry Pi is used as an independent processing unit to handle visual and scalar data in real time, without reliance on a centralized server for model loading and processing.
3. Deep Neural Network (DNN) models require a high-end processing unit for execution in real time. Therefore, a lightweight deep model has been proposed for resource-constrained devices, which is executed in real time for autonomous maneuvering.

The rest of the paper is organized as follows: Section II conducts a literature review. In Section III, the proposed framework and the methodology offered for the development of the autonomous car are discussed. Experimental results and challenges faced during the development of this project are summarized in Section IV. Section V concludes the paper with the future directions of work.

II. RELATED WORK

In this section, we discuss some of the existing research already completed on autonomous driving generally or on the individual factor involved in autonomous cars, either as a safety tool for public or as a financial source for industries. This section further incorporates the traditional and deep learning approaches, limitations of the existing systems, and current challenges encountered in the domain of autonomous driving.

a) Conventional Approaches

Various car manufacturing and IT-related companies including General Motors, Waymo, Daimler-Bosch, Volkswagen Group, and Groupe Peugeot S.A. (Groupe PSA) are aiming to contribute in the field of autonomous cars. Human error can be minimized by making the concept commercial, which will provide a means of safe transport for public. As autonomous cars are equipped with many sensors, these produce lots of data. The latter is analyzed by various companies including Google and Facebook and researchers for many purposes and applications. For instance, Chen et al. [18] constructed high-quality 3D objects in images for autonomous cars. Objects from high-fidelity imagery are constructed in the form of 3D bounding boxes. The problem has been formulated using the Markov field encoding, ground plane, and various depth structures. Their technique performs well on the KITTI training set leading to 25% higher recall than other existing techniques. Similarly, Guidolini et al. [19] proposed an automatic obstacle avoidance mechanism for autonomous cars using the IARA dataset. The technology works effectively by avoiding obstacles that appear suddenly in the frame of view as well as when they appear normally as expected. The response time for obstacle avoidance is nearly 3 milliseconds. Choi et al. [20] studied obstacle detection using LIDARs. The algorithm designed for obstacle detection generates the obstacle position map from LIDARs data. With the help of six LIDARs fitted on a passenger car, it can successfully detect obstacles by reaching its targets.

Lenoard et al. [21] developed a software architecture for a perception-driven autonomous car. In the proposed work, they feed all the sensor data into a Kino-dynamic motion planning algorithm to accomplish the autonomous tactics. They have achieved autonomous driving of 55 miles over 6 hours without a mishap. In contrast with the current trends in autonomous car, Kalra et al. [22] carried out a study on how safe the journey of the autonomous car will be when driving up to hundreds of billions miles. Their study suggests that autonomous cars must be driven for billions of miles to check the reliability of autonomous driving. Their study claims that due to safety reasons it may not be possible to make it available for public use. Another study conducted by Petrovskaya et al. [23] developed a module to detect a moving vehicle and track the detected vehicle for their autonomous robot named “Junior”. To estimate the position of the tracked vehicle, they used geometric and dynamic properties of the detected vehicles. For position tracking, a Bayes filter is used for each vehicle and a Red-

Blackwellised Particle Filter (RBPF) is applied to eliminate separate data segmentation. "Junior" can find the position, shape, and velocity of the vehicles tracked.

b) Deep Learning Approaches

DNN, which is a part of artificial intelligence, is widely used in different fields including computer vision [24, 25], natural language processing [26], speech recognition [27], machine translation [28], social networks filtering [29], and bioinformatics [30]. In computer vision, DNN is utilized for image classification [31, 32] and object detection [33]. Deep Learning is also employed for object segmentation and several other applications [34]. For instance, Behrendt et al. [35] carried out a study on detection of traffic lights for autonomous cars in real time using DNNs. The information for the traffic lights in the early system was map-based. A neutral network is trained on a thousand images to achieve high accuracy. In experimental analysis, a video sequence of more than 8,000 frames was used. The contribution of the proposed system is traffic light detection, tracker, and classification of light (red, yellow, green, or off). The proposed approach achieved high accuracy in challenging environments. Using artificial intelligence for vehicle and lane detection in real time is another important task, to which several works are dedicated. For example, Huval et al. [36] studied the problem of vehicle and lane detection with the help of DNNs. This study mainly focused on vehicle detection in real-time scenarios. A large amount of data is usually required for training a neural network, which includes data on different road scenarios and highways. The proposed algorithm reported to produce high accuracy during practical scenarios in challenging environments. Instead of mediated perception and behavior reflex approaches, Chen et al. [37] proposed another paradigm – a direct perception approach for the estimation of afford driving. In the proposed method, an input image is resolved into small key points. This representation offers a description of the scene for autonomous driving. A deep Convolutional Neural Network was trained for this purpose, whose details are given in Section III.

c) Limitations and Major Challenges

One of the biggest problems for autonomous cars is finding the track and following it for the rest of the journey. Sun et al. [38] carried a brief study on lane detection for autonomous cars. In their work, images are converted into the binary form by adaptive thresholding, and then edges of the road are extracted. Lanes are extracted from the edges followed by their detection. They used different road images of various weather conditions, basic thresholds, and proportional coefficients. The same problem is also discussed by Saha et al. [39] suggesting a similar algorithm. An RGB image is taken from the autonomous car and converted into a grayscale image. A flood-fill algorithm labels the largest components that are connected in the grayscale image. After applying the flood-fill algorithm, extraction of the largest connected component is completed. Unwanted regions in the image are skipped and ROI is filtered for lane and road edge detection. Hong et al. [40] studied the problem of detecting solid and dashed lanes in the road. Existing techniques only detect the central lanes in the road and are unable to detect the solid and dashed lanes. The proposed algorithm overcomes these limitations and can differentiate between dashed and solid lanes.

The privacy of autonomous cars in vehicular networks is paramount in all aspects, same as privacy of other domains, such as industrial and surveillance environments [41]. For self-driving cars, a secure channel is needed for exchanging data. M. Ali Alheeti et al. [42] proposed a scheme for intrusion detection based on Integrated Circuit Metric (ICM). Authors claim that the features extracted by the ICM can be used for many purposes including security and identification with the efficient use of time, speed, and memory. The study focused on enhancing the authentication in autonomous driving and building an IDS as well as making them intelligent by using the features of autonomous vehicles. The main theme of the study was proposing a scheme mainly based on MEMS gyroscope and constructing a system for identifying the car as an ICM vehicle. The authors argue for satisfactory results while using FFNN-IDS and k-NN-IDS in blocking malevolent vehicles. M. Ali et al. [43] proposed a scheme for detecting a malicious car in an urban transport scenario. The detection system was mainly based on a Fuzzy Petri Net (FPN). The FPN was used for detecting dropped packets in vehicular ad-hoc networks. By finding the number of received and dropped packets, IDS showed satisfactory results in terms of vehicular network security.

Self-driving cars use V2V or V2I technology for exchanging information in a vehicular network. Gora et al. [44] built a microscopic module to arrange the traffic and exchange information with other autonomous car in vehicular networks. They developed simulation software to reduce the chances of collision in autonomous cars, while envisioning the flow of traffic. They included traffic lights, road lane junctions, and a specific route for the car to reach a random point. Speed and other properties of autonomous cars are adjusted according to the information received from other vehicles for safer driving. Path planning, vehicle sensing, and navigation of autonomous cars are challenging tasks. It is crucial for the autonomous car to safely maneuver in different environments. This problem is discussed by Huy et al. [45] who propose a path-planning algorithm. The latter follows a probabilistic method while filtering particles for dynamic obstacles. Obstacle points are detected by using a support vector machine to generate a smooth path via a Bezier curves algorithm. They tested the algorithm in simulation software and claimed that it obtained acceptable results in complicated scenarios with multiple moving objects. Road, vehicle, and obstacle recognition on the road using a vision sensor is another challenging task in the field of moving robotics. Birdal et al. [46] proposed algorithms, which help overcome the aforementioned problems while driving in vehicular networks. Images from the vision sensor are converted into gray values and then different techniques including texture analysis, geometric transformation, background modeling, and contour algorithms are applied for extracting different ROIs. The latter are tested using different road images, thus achieving significant results.

d) Advance of State-of-the-Art

The perspectives of autonomous vehicles cannot be easily predicted. However, such prediction enables planning possible

future conditions. Many practitioners and analysts are worried about the future of self-driving cars in terms of their effect on parking, traffic problems, and public transportation [47]. In this line of reasoning, the Society of Automobile Engineers (SAE) claims that by 2030 autonomous cars will be safe and reliable enough to replace human driving, thereby minimizing driver stress and tediousness [48]. Advancement in technologies for autonomous vehicles will also reduce accidents, congestion, and pollution problems. Some analysts also foresee that the total cost of shared passenger travel in autonomous vehicles will be lower than in human operated vehicles [48].

Despite many benefits of autonomous cars, they do impose extra cost by adopting different equipment and tools for processing and sensing the external environment. For instance, the introduction of a simple electronic device, such as an adaptive cruise, an active lane assist, a high beam assist, or a top-view camera, will cost thousands of dollars. Similarly, the processing unit of autonomous vehicles is even more expensive, thus increasing the overall price as well as requiring extra costs [48]. By analyzing the current trends in autonomous cars, we conclude that their focus is mainly on autonomous decision-making technologies. Even though there exist variations in such technologies, their common attribute is an autonomous driver system for the vehicle.

III PROPOSED METHODOLOGY

For a better understanding of the proposed system, this section is broadly divided into two sections: A) System Design and B) Methodology. Subsection A is structured as: i) input units, which relate to vision and ultrasonic sensors, ii) camera calibration and distance measurements, and iii) output units, which include DC motors, a buzzer, and a LED. Subsection B is further divided into: i) preprocessing module, where morphological operations are applied on the input frame, ii) classification module, which includes DNN units, traffic sign detection, and distance calculation using a vision sensor, and iii) decision module for driving the autonomous car.

A. *System Design*

This subsection provides a detailed description of how the input units (vision and ultrasonic sensors) are connected and the information inferring is elaborated. The system design for our proposed method consists of three main parts: inputs units, camera calibration for distance measurements, and output units.

i. *Input Units*

The input components are composed of a vision sensor with an ultrasonic sensor working together to collect data from the environment in real time. A stream of video frames is supplied to Raspberry Pi from camera, which are processed by the processing unit. Ultrasonic HC-SR04 unit is used for obstacle detection and measurements of distance to the obstacle. Table I shows the specification of hardware modules used in this prototype car. The specified ultrasonic unit has four pins, Trigger pulse unit (TRIG), Echo unit (ECHO), Ground unit (GND), and Power supply unit (VCC). The power was supplied to the unit from 5 volts General Purpose Input Output (GPIO) pins of Raspberry Pi. Fig. 1 shows the assembly of all hardware sensors in detail. The ultrasonic sensor works by the principle that a pulse is sent from the sensor using TRIG. This pulse is

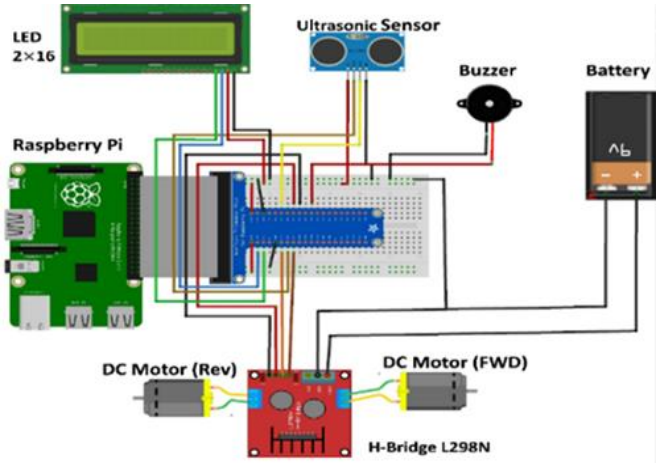

Fig. 1. Hardware assembly of our self-driving car. Raspberry Pi GPIO pins are extended through extension board, where to physical units are connected.

bounced by the nearby objects and received by the ECHO. Distance is calculated from the difference of the transmitted pulse and the received pulse using a speed formula. The speed of sound is 343 (34300 centimeter) meters per second in the air. Time is divided by 2 because the pulse travels to the object and back again:

$$d = speed \times time \quad , \tag{1}$$

$$d = \frac{34300 \times echo}{2} \quad , \tag{2}$$

$$d = 17150 \times echo \quad . \tag{3}$$

In equation (3), *echo* is the return pulse time and *d* is the distance to the object. A threshold value of 15 inches is set for the distance. The obtained distance is passed from the threshold for controlling the motor's Pulse Width Modulation (PWM). For instance, if the distance is greater than the threshold value, PWM is set to 100 Hertz, and if the distance is less than the threshold value, PWM is set to zero. The returned value from the ultrasonic sensor is also used for the buzzer as a horn. When the car reaches an obstacle, the buzzer is turned on after the car stops and warns on the obstacle next to it. A message *Obstacle Detected* is displayed on the LED. Fig. 2 captures images of the self-driving car on a pre-defined track.

TABLE I
SPECIFICATION OF HARDWARE MODULES

Module	Specification
Camera	8 megapixels,1080p30,780p60
Ultrasonic-SR04	5v, 15mA, Ranging distance 6inches-156inches
DC-motor	5v, 20mA, 2000RPM
2x16 LED	5v, 20mA
Buzzer	5v, 12mA

ii. *Camera Calibration and Distance Measurements*

Radial and tangential types of distortion are commonly introduced by a camera to the images, due to which some of the meaningful information is lost. Straight lines in the images will appear curved in radial distortion. This effect can be easily observed while moving away from the center of the image. In Fig. 3, a checkered board is marked with red lines at the edges.

It can be seen that the border is not a straight line and does not match with the red line.

Fig. 2. Self-driving car on pre-designed track with traffic signs. To assess the efficiency of the system, several experiments are performed for each traffic sign.

The method proposed by [49] is used for the calibration of the Raspberry Pi camera. The distortion is resolved using equations (4) and (5).

$$U_{dist} = U(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)\,, \quad (4)$$

$$V_{dist} = V(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)\,, \quad (5)$$

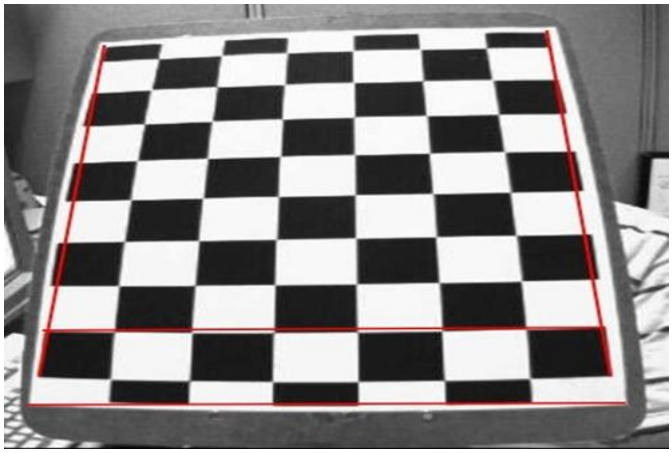

Fig. 3. Checkered-board is used for camera calibration using OpenCV

where U_{dist} and V_{dist} are the undistorted pixel location and U and V are the normalized image coordinates. Tangential distortion occurs because the lenses that capture images are not aligned perfectly parallel to the plane of the image. Due to this, some regions in the image appear closer than expected. This can be resolved with the help of equations (6) and (7). OpenCV library has built-in tools to calibrate the camera and reduce the distortions based on (6) and (7).

$$U_{dist} = U + \left(2P_1 UV + P_2(R^2 + 2U^2)\right), \quad (6)$$

$$V_{dist} = V + (P_1(R^2 + 2V^2) + 2P_2 UV). \quad (7)$$

There are five parameters, which are known as distortion coefficients and are given below:

$$Distortion\ Coefficient = (k1\ k2\ P1\ P2\ k3)\,, \quad (8)$$

where $k1$, $k2$, and $k3$ are the radial distortion, while $P1$ and $P2$ are the tangential distortion coefficient of lens. Along with these parameters, the information like intrinsic and extrinsic parameters of the camera is also required. Intrinsic information includes focal length (*fx*, *fy*) and optical centers (*cx*, *cy*). All these parameters are camera-specific and only depend on the model and focal length of the camera. Expression (9) is used for the camera calibration.

$$\text{Camera Matrix} = \begin{vmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{vmatrix}. \quad (9)$$

Extrinsic parameters are related to the translation and rotation vectors, which translate into the coordinates of a three-dimensional point. All these distortions must be corrected to obtain optimized results. Sample images, in which the patterns are well defined, are provided for correction. Checker board images are suitable in this case.

OpenCV provides convenient functions to further facilitate the calibration process. The OpenCV method findChessboardCorner() returns corners when a 7×6 grid image is passed to it as shown in Fig. 4. The accuracy of the points is improved by using the cornerSubPix() method. The drawChessboardCorner() function is used for drawing patterns.

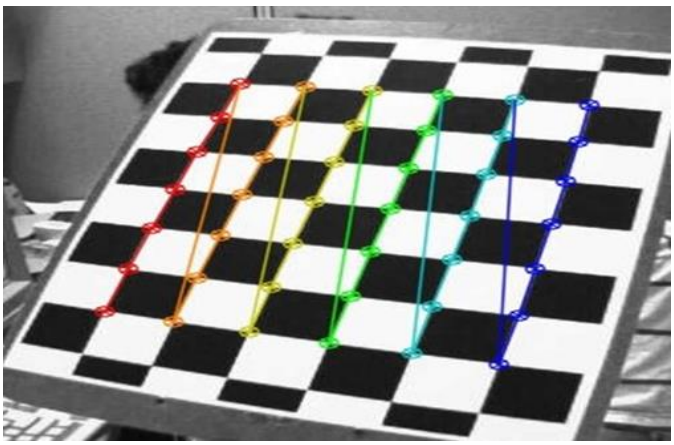

Fig. 4. Calibrated output image of the Raspberry Pi Camera using OpenCV

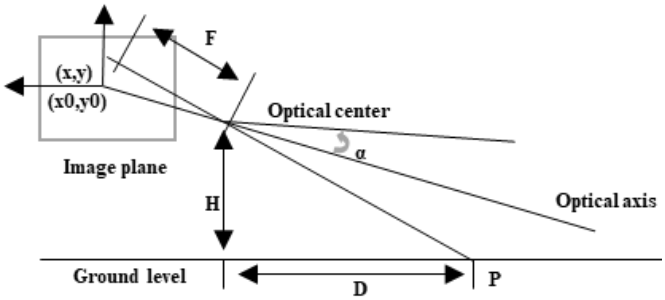

Fig. 5. Monocular-vision based distance measurement

The calibration of the camera is now a one-step process calibrateCamera() by obtaining the image and the object points. This will return the camera matrix, the distortion coefficients, as well as the rotation and translation vectors. The measurements of distance for detecting a sign using a

monocular vision sensor is one of the most challenging tasks in this system. The method proposed by [50] to calculate the distance from traffic signs in real time is employed.

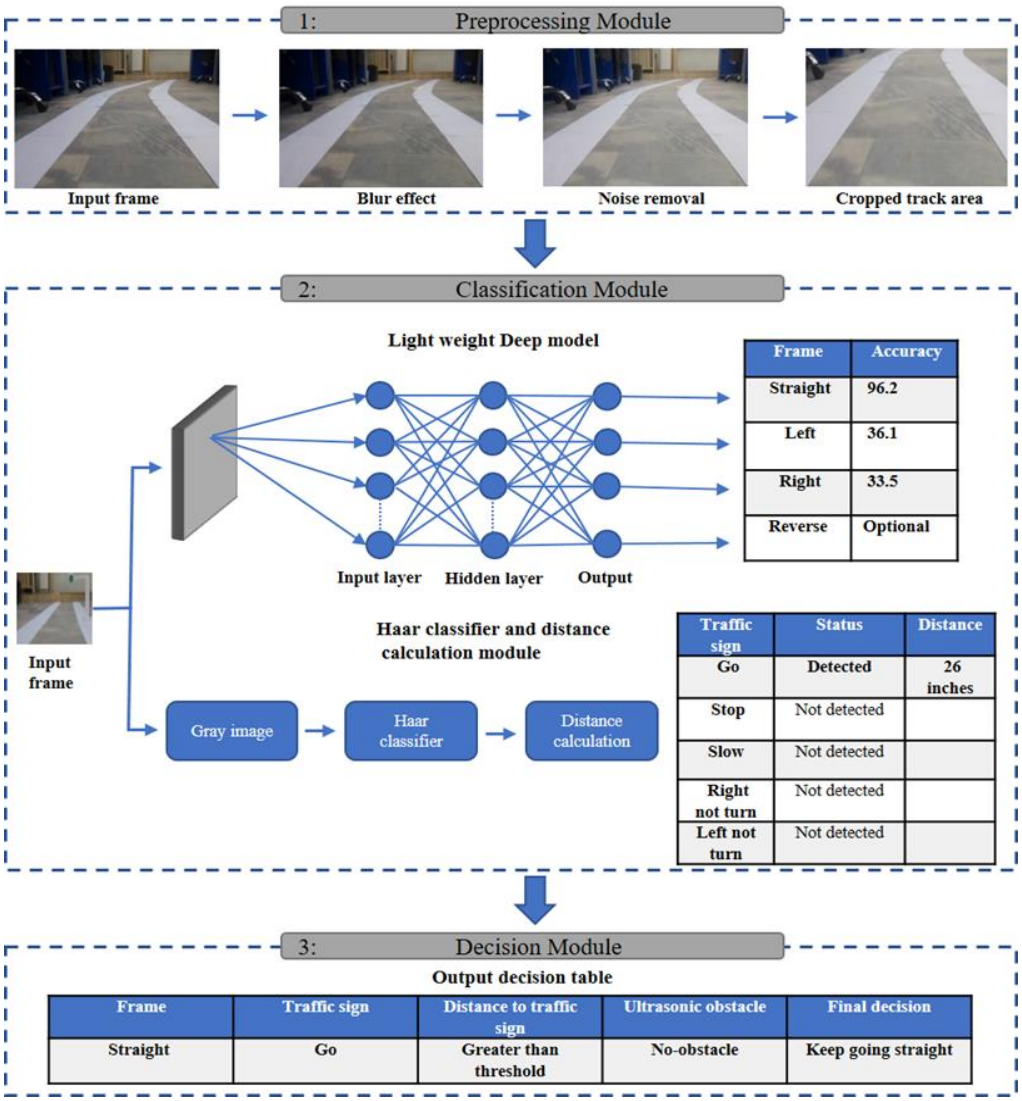

Fig. 6. Detailed overview of the proposed framework. In Preprocessing module, input frame is passed from morphological operation to enhance the frame. In Classification module, input frame is further processed for traffic sign detection i.e., Go, Stop, Slow, "Right not turn" and "Left not turn", and classification of frame i.e. straight, left, or right. Decision module stores information received from the second step into an array. Decision is taken on the array from left to right, in which the last column is the final decision for the input given conditions.

From Fig. 5, we suppose that an object *P* is at the distance of *D* from the optical center. *H* is the height of the optical center, *F* is the focal length of the camera, (*x*0, *y*0) are the point of intersection for the image and the optical axis; while the projection of point *P* is given by (*x*, *y*). Consider further parameters, such as (*u*0, *v*0) for the camera coordination; then, the physical dimension of the pixel corresponding to x-axis and y-axis on the image plane are D_x and D_y.

$$D = \frac{H}{tan(\alpha + arctan(y_1 - y_0/f))}, \tag{10}$$

$$u = \frac{x}{D_x} + u_0, \quad v = \frac{y}{D_y} + v_0, \tag{11}$$

$$D = \frac{H}{tan(\alpha + arctan(v_1 - v_0/a_y))}, a_y = \frac{f}{D_y}. \tag{12}$$

iii. Output Units

The output components used in the development of the system include DC motors (wheels), a 2×16 led for showing the necessary information (i.e., IP address of the Raspberry Pi, "Ready", "Obstacle detected", "Left turn", "Right turn", "Going straight", and "Stop"), and a buzzer as a horn. In the first stage, an ultrasonic sensor is measuring the distance to the obstacle (if any). If there is no obstacle in the track ahead, the car travels smoothly on it. If there is an obstacle on the track, the ultrasonic sensor measures the distance to this obstacle. When the distance reaches a minimum of 15 inches,

voltage supply to the wheels is cut off and the car stops at a distance of 15 inches from the obstacle. In the second stage, the wheels are controlled from the vision sensor. Each frame is scanned for the detection of traffic signs. After that, the ROI is passed through the respective Haar cascade classifier for further action. Four Haar cascade classifiers are used, and in the identification of each traffic sign, the wheels are controlled according to the identified sign.

B. Methodology

This subsection offers a closer look at the proposed system where each module combines different units. The first module is composed of various morphological operations carried on the input frames for enhancement. Classification module consists of two units: DNN and traffic sign detection with distance calculation. The last module is an array where different predictions from the second module are stored. Operation on this array is carried from left to right where the last column is the final decision for given input conditions. The overall view of our methodology is shown in Fig. 6.

i. Preprocessing Module

Input frames from the vision sensor are passed from several modules in order to remove noise and crop the track region from the frame. In the first step of the preprocessing, an image is passed from Gaussian blur effect step to smoothen it. Due to different lighting conditions and motion of the car, input frames contain noise, which is removed by using a noise removal module. This is based on a morphological operation, such as opening followed by closing. In the last step of the preprocessing, extra regions from the frame are cropped to only the track region and resized to 640×480 resolution. This frame is supplied to the classification module for further operations.

ii. Classification Module

This module is composed of two parts: 1) neural network and 2) traffic sign detection with distance calculation units. Input frame from the preprocessing module is passed from each unit of the classification module for further predictions.

- *Neural Network for Prototype Model Car*

The performance of the DNN architecture is directly depended upon the total number of hidden layers inside the model. However, these hidden layers and the number of nodes do not interact with the outside environment directly, but they affect the output result. The DNN model with very few neurons in the hidden layers will cause underfitting. In this case, the number of neurons is insufficient to detect the signal accurately and transfer the output to the next hidden layer. On the other hand, using a larger number of neurons in the hidden layers will cause overfitting, which requires more information and training data. To overcome these problems, there should be a tradeoff among the numbers of hidden layers, neurons, and training data. For this reason, we started with a simple DNN containing 128 nodes in the first hidden layer and 16 nodes in the second hidden layer. The complete details of these nodes are given in Table II, where the first two columns represent the numbers of nodes in the first and the second hidden layer with their corresponding accuracy in the third column. The performance of the model is observed carefully, and the numbers of nodes are adjusted accordingly.

After a node adjustment in the hidden layers, Hidden layers 1 and 2 contain 4800 and 64 nodes, respectively, while the output layer contains four nodes that are responsible for controlling the wheels for driving. The output of the model includes straight, left, right, and an optional layer "reverse" that is not used in the current system. The number of images used for training is ten thousand, where 80% of the images are used for training and the rest of 20% serve validation. The images used for training are cropped, and only the track portions of these images are supplied to the network for classification. The size of these images is fixed to 80 × 60 (4800 nodes) for a performance optimization of the network. During the training process, the model is trained for more than six hundred epochs with a starting learning rate of 0.01, which is adjusted by the gradient descent optimizer according to the performance of the model. Fig. 7 shows a brief description of the neural network for the self-driving car.

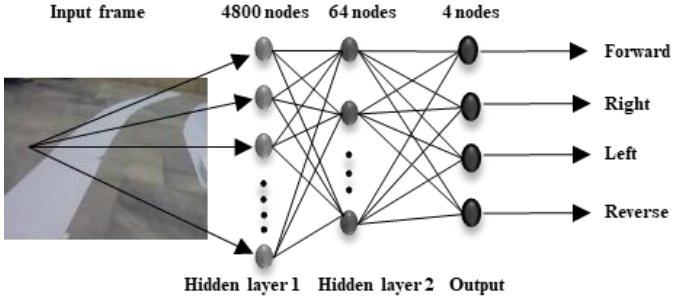

Fig. 7. Neural network for self-driving car maneuvering, where Hidden layer 1, Hidden layer 2, and final Output contain 4800, 64, and 4 nodes, respectively.

TABLE II
NUMBER OF NODES AND CORRESPONDING ACCURACY

No. of nodes in Hidden layer 1	No. of nodes in Hidden layer 2	Accuracy (%)
128	16	53
512	32	62
512	64	68
512	128	62
1024	64	72
2048	64	86
3072	64	92
4800	64	97

The processing of each frame is presented in Fig. 8. In the first step, an image is cropped and converted into a Numpy array. In the second step, labels for the training images are supplied in a separate labels file. After the DNN training, the trained model is saved to the local directory for future use.

- *Traffic Sign Detection and Distance Calculation*

An extraction of multiple ROIs from the supplied frames includes traffic sign detection and distance measurements to the detected signs. For an intelligent transportation system, the detection and recognition of traffic signs is an essential capability. Taking advantage of the "Haar based classifier" method by P. Viola [51], we conduct traffic sign detection and recognition. This algorithm requires a large number of positive and negatives images to train the cascade function.

A separate Haar Cascade classifier is trained for each traffic sign. OpenCV provides libraries for both training and

detecting the Haar cascades. 2,000 negative images (other than the traffic sign images) and 50 positive images (Traffic sign images) were used for training the Haar cascade. Only regions of interest were supplied in the positive case. Five Haar cascades including "Go", "Stop", "No Left Turn", "No Right Turn", and "Slow" signs are used in this system. Fig. 9 shows samples of positive and negative images, and Table III demonstrates traffic signs, numbers of positive and negative images, and size of the images. Each frame is given to a function for detecting the ROI, while only the contour region is extracted and passed from the distance calculation module for distance calculation. Decisions are taken after recognizing a specific sign.

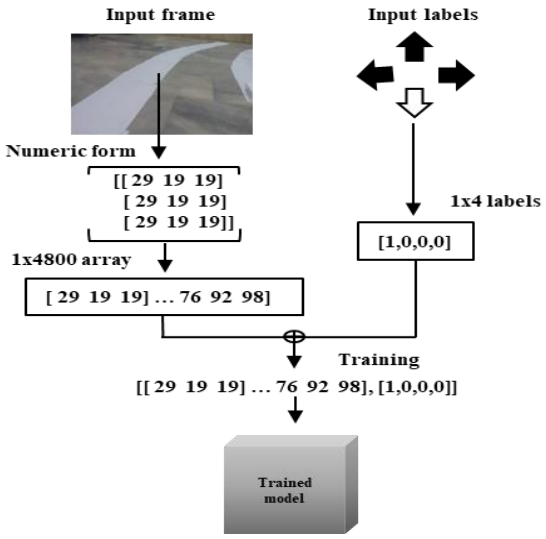

Fig. 8. Training a DNN-based classifier for a self-driving car. In the first step, a multi-dimension frame is converted into the 1×4800 array. In the next step, the associated labels are concatenated at the end of each frame array. After training the DNN, the trained model is saved for future use.

TABLE III
DIMENSIONS AND NUMBERS OF POSITIVE/NEGATIVE IMAGES FOR HAAR CASCADE TRAINING

Sign	Positive	Negative	Size (pixel)
Go	50	2000	30×30
Stop	50	2000	30×30
Slow	50	2000	30×30
No Left	50	2000	30×30
No Right	50	2000	30×30

Fig. 9. Training of Haar cascades with Positive (left) and Negative (right) images for a traffic signs detection.

iii. Output Decision Module

Processed data from the processing module are passed to the output decision module for maneuvering the model car on the track. Return data from the processing module are put into an array. The returned array contains information about the frame classification (straight, right, left), traffic sign detection (go, stop, left not turn, right not turn, slow), distance to the detected traffic sign, distance to the obstacle detected by an ultrasonic sensor, and decision i.e., stop, go, turn left or right. For instance, if the frame is classified as 'right', the detected sign is *Go*, the distance to the traffic sign is greater than the threshold, and there is no obstacle on the track, then the final decision for this type of information is "turn right". These decisions are forwarded to the output units in system design where voltage to certain motors is controlled depending upon the decision.

IV. EXPERIMENTAL RESULTS

All the experiments are carried out on optimized OpenCV version 3.3.0 compiled on Raspberry Pi 3 Model B+, 4x ARM Cortex-A53 1.2GH processor using Python version 3.6 and Tensorflow version 1.4.0. Other dependencies include Numpy, Scipy, and Matplotlib for visualizing and processing of output data. The Raspberry Pi has limited resources in terms of the computational capacity and memory [52]. ARM processor comes with ARM NEON optimization architecture and VFPV3 extension for the purposes of faster image, video, and speech processing, machine learning techniques, and floating-point optimization. ARM NEON supports the use of Single Instruction Multiple Data (SIMD), where multiple processing elements in the pipeline perform on multiple data points, all executed with a single instruction. VFPV3 comes with configurable rounding modes and customizable default Not a Number (NaN) mode. Enabling all these special modes of Raspberry Pi while compiling OpenCV results in running our neural network faster, while the compiled OpenCV can be referred as Optimized OpenCV. Taking advantage of these features (ARM NEON, SIMD, VFPV3, and NaN) in the Raspberry Pi, OpenCV is a built-in optimized mode. Further, tensorflow provides a possibility to use a number of processor cores for a task. Leveraging this feature, deep experiments are set on multiple cores and different versions of OpenCV. Fig. 10 shows the average execution time of the normal OpenCV, optimized OpenCV, and Optimized OpenCV with the support of Movidius Intel Computing Stick. Fig. 11 demonstrates the average temperature of the core during frame processing. A high increase in temperature on cores 3 and 4 is because the load shifting probability is decreased among the CPU cores.

TABLE IV
DESCRIPTION OF PARAMETERS

Parameter	Description
↑	Straight
→	Right
←	Left
↔	Not straight
↖	Not right
↗	Not left

In order to achieve the maximum possible accuracy and to reduce the computational cost, the number of frames per second were decreased, thus generating a prediction 5x faster i.e., 5 frames/s. The parameters used in this work are described in Table IV. For evaluating the model car, different types of tests were conducted using various track scenarios. A total number of 158 frames (31.6 seconds video) were evaluated under different track scenarios.

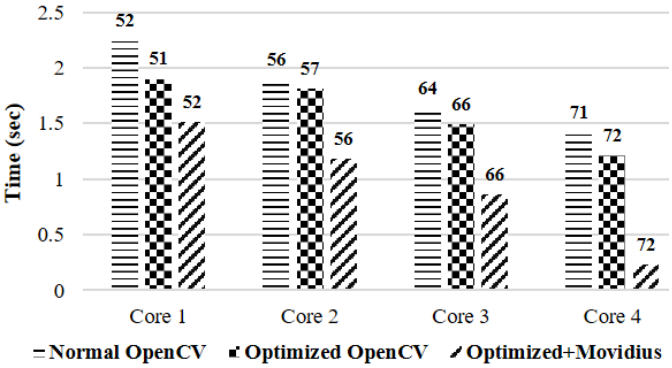

Fig. 10. Time complexity. Number of CPU Cores vs. different versions of OpenCV, i.e., Normal, Optimized, and Optimized + Movidius confirming that the number of available resources and the use of Movidius improve performance of the system.

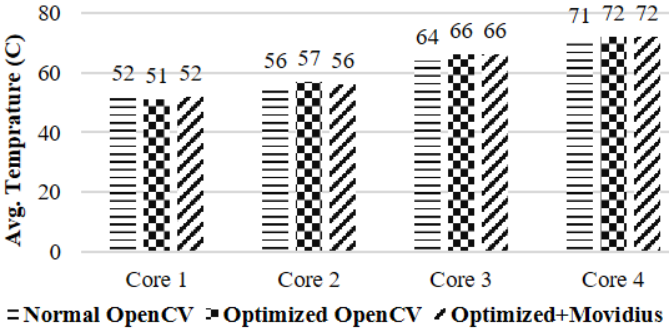

Fig. 11. Average temperature of different CPU cores on various versions of OpenCV, i.e., Normal, Optimized, and Optimized + Movidius. Average temperature of Raspberry Pi increases as DNN process is distributed among several cores.

Two versions of videos from the model car were obtained: the first version was converted into frames and categorized into three groups including straight, left, and right. Each group is further divided into two classes: (straight and non-straight), (right and non-right), and (left and non-left). Each frame obtained from the first version is placed in their respective class as a ground truth for prediction and training. The second version video was predicted by the model car and was compared with the ground truth. The overall accuracy of the model is presented in Table V. The results are also color coded for the ease of interpretation. Green columns represent tests for the class "straight"; the ground truth for this class includes 60 "straight" frames and 98 "non-straight" frames. During the validation, the model car has predicted all the 60 frames as "straight" and 98 as "non-straight" frames, thus reaching its destination without any error. Yellow columns show the ground truth for the class "right". In this class, 55 are "right" frames and 103 are "non-right" frames. The model car has identified 52 as "right" frames and 106 frames as "non-right" frames. The fifth and sixth columns of Table 5 are colored blue; they are the ground truth for the class "left". There are 103 frames for "left" and 55 frames for "non-left". The model car identified 107 frames as "left" and 51 frames as "non-left", thus attaining an overall accuracy of 98.5%. Distance calculations with monocular vision sensors became a challenging task. A shorter distance to the sign gives nearly the actual distance, but when the distance from the sign was increased, error in the distance calculations also increased as shown in Fig. 12.

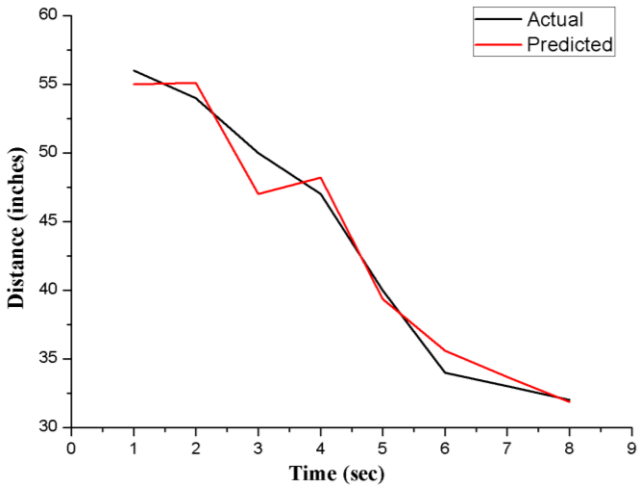

Fig. 12. Distance calculated by a vision sensor. As model car moves toward the detected traffic sign, the difference between the actual and the predicted values decreases.

Fig. 13 shows a sample image of the distance calculated by a vision sensor. The difference between the actual distance and the distance calculated by the vision sensor may be due to the following reasons:

1) Error in the measurements of the actual values.
2) Error in the camera calibration.
3) Variation in the object bounding box while detecting the signs.
4) Non-linear relationships between distance and camera coordinates. With greater distance, the coordinates of the camera change rapidly, thus resulting in higher error.
5) Raspberry Pi camera is general-purpose camera and has average image quality.

Ultrasonic sensors have only been used for detecting objects and distances from the car. An ultrasonic sensor uses sound waves to calculate the distance. Due to this, some errors were experienced in calculating the distance during our demonstrations. Fig. 14 shows the actual distance and the distance calculated by the ultrasonic sensor. The difference between the actual and the measured values may be due to the following reasons:

1 Sound waves easily strike larger objects as compared to smaller objects. The farther the distance is, the greater the error is since few pulses are returned from the object.

TABLE V
OVERALL ACCURACY OF MODEL CAR

No. of Tests	Ground Truth						Proposed System						Overall Accuracy
	↑	↔	→	↖	←	↗	↑	↔	→	↖	←	↗	
Test 1	*60*	*98*	*55*	*103*	*103*	*55*	*60*	*98*	*52*	*106*	*107*	*51*	**98.5 %**
Test 2	*79*	*79*	*71*	*87*	*98*	*60*	*79*	*79*	*73*	*85*	*99*	*59*	**99.1 %**
Test 3	*87*	*71*	*118*	*40*	*40*	*118*	*88*	*70*	*116*	*42*	*39*	*119*	**98.9 %**
Test 4	50	108	90	68	100	108	53	105	87	71	99	59	**98.8 %**

Fig. 13. Sample images showing distance to the traffic sign by using a vision sensor. Each sign is detected (green rectangle), recognized, and the distance is calculated between the traffic sign and the model car.

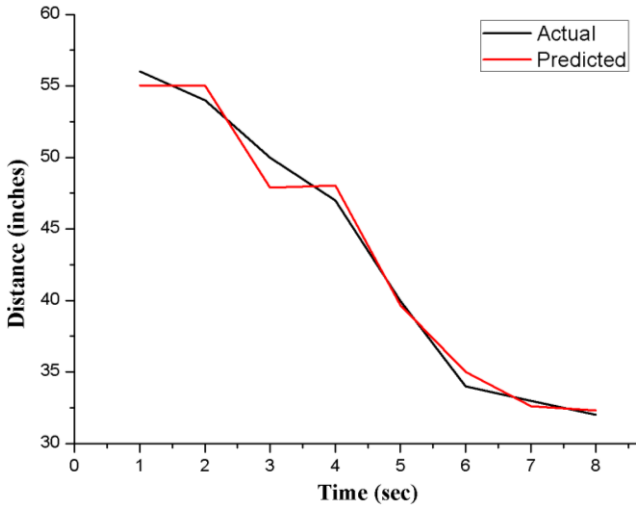

Fig. 14. Distance measurements: actual vs. predicted by an ultrasonic sensor. The actual and predicted distances are nearly equal as the model car moves towards the obstacle.

2 Ultrasonic waves are greatly influenced by the air temperature. The sensor calculates the distance to the object using the speed of sound. The speed of ultrasonic waves alters as the air temperature changes [53].

3 The ultrasonic waves are also influenced by air pressure.

A. Energy Consumption

The total expenditures used by a system during completing a specific task are known as energy consumption. To evaluate the total energy consumption by our system with deep learning model, 5 frames have been processed in one second. The parameters have been calculated using a Keweisi device while estimating the total power consumption of our system.

Fig. 15 shows sample images of power consumption. The unit power and the energy consumption of the Raspberry Pi can be observed in Table VI when the system is idle and no task is in progress. The total ampere, voltage, time, power, and energy drain by Raspberry Pi during processing a single frame with deep learning can be seen in Table VII.

B. Comparison with Other Computing Platforms

To compare the performance of Raspberry Pi with other computing platforms in terms of the energy, power, and average processing time of a frame, we used Intel CPU and GPU with NVIDIA graphics card. The specification of each system is

presented in Table VIII. In order to evaluate the performance of the deep model on CPU and GPU, a multi-thread TCP server is used for receiving the video stream and the ultrasonic data from Raspberry Pi on the computer. Data from the Raspberry Pi is processed on the computer and only the decisions reached by the deep model are sent to the Raspberry Pi to take the necessary actions.

TABLE VI
POWER AND ENERGY CONSUMPTION OF RASPBEERY PI IN IDLE STATE

No. of tests	Current (A)	Voltage (V)	Power (W)	Energy (J)
1	0.12	5.25	0.63	0.63
2	0.19	5.24	0.99	0.99
3	0.16	5.24	0.83	0.83
4	0.21	5.25	1.10	1.10
5	0.17	5.25	0.89	0.89
6	0.22	5.23	1.15	1.15
7	0.18	5.25	0.94	0.94

Fig. 15. Power consumption for on-board execution of deep learning model on Raspberry Pi. The upper value shows current while the second value shows voltage consumed by the Raspberry Pi during execution of the DNN model.

TABLE VII
ENERGY AND POWER CONSUMPTION DURING NEURAL NETWORK EXECUTION

No. of tests	Current (A)	Voltage (V)	Time (T)	Power (W)	Energy (J)
1	0.82	5.25	0.23	4.30	0.98
2	0.99	5.23	0.25	5.17	1.29
3	0.99	5.26	0.30	5.20	1.56
4	1.0	5.23	0.26	5.25	1.36
5	1.2	5.22	0.23	6.30	1.44
6	0.99	5.25	0.28	5.19	1.45
7	0.89	5.22	0.29	4.64	1.34
8	0.87	5.26	0.25	4.57	1.14

TABLE VIII
SPECIFICATION OF COMPUTING SYSTEM

Hardware platform	Raspberry Pi	Intel CPU	GPU
Processor	4x ARM Cortex-A53 1.2GH	Intel Core i3-40 1.70GHz	Intel Core i5-50 3.32GHz
RAM	1GB-LPDDR2	8GB-DDR3	8GB-DDR4
GPU-use	No	No	8GB-NVIDIA GeForce-GTX-1070
Cost	$35	$280	$3300

In Fig. 16, the red bar shows the average power consumption of the GPU during processing the deep model. The total average power consumed by the GPU on the frame processing is 440 Watts, as the normal current increases from 1.5 to 2amps on the execution of the neural network. The power consumption of the GPU is 330 Watts in the idle state. Yellow bar shows the average power consumption of the CPU on the execution of the deep model. The total average power consumed by the CPU on executing the deep model is 224.4 Watts. A small amount of power of 6 Watts is consumed by the Raspberry Pi, as shown in green color in Fig. 16, while attaining the same accuracy as obtained by the GPU and CPU.

The energy consumption of different computing platforms is presented in Fig. 17. The average energy consumption of the GPU is 4.4 Joule on a single frame execution. The CPU consumes higher energy than the GPU since the former is not Graphics card enabled. Raspberry Pi, in contrast to GPU and CPU, consumes less energy, which is on average 1.38 Joule/frame.

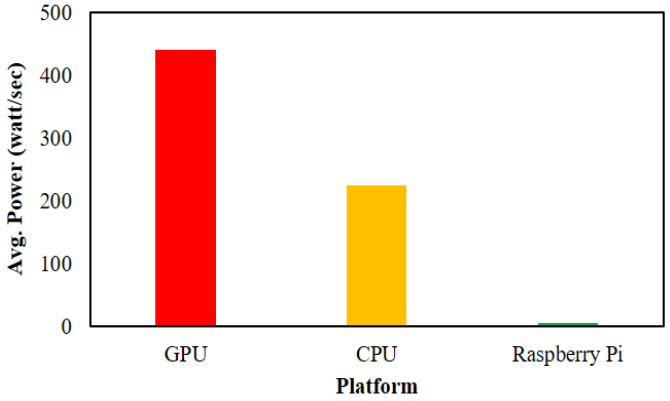

Fig. 16. Average power consumption of different multicores computing platforms.

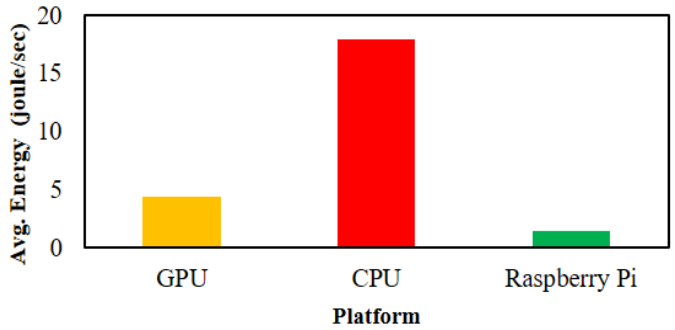

Fig. 17. Average energy consumption on different platforms

Fig. 18 shows the time complexity of a single frame on different hardware platforms. A total of 1,000 frames are executed on each platform. The time complexity of the GPU on a single frame is 0.01 seconds. The average time consumed by the CPU to execute a single frame is 0.06 seconds. Raspberry Pi did not perform well as it had limited resources and took an average of 0.23 seconds to execute a single frame. Summarizing all the conducted experiments, we conclude that Raspberry Pi, an economical computing platform, is powerful enough and capable of running a DNN for real-time applications with low power and energy consumption. GPU and CPU perform well in terms of the execution time, but as evident from our experimental results, these platforms consume excessive energy and power while attaining a similar accuracy with Raspberry Pi.

C. *Speed Test*

Speed of an object can be referred to as the total distance covered by a body over a unit time. To analyze the speed of our model car on the track, we have conducted a total of seven tests to assess the average speed of the car. Details of each test are offered in Table IX. The length of the track was kept to 132 inches (11 feet). The speed of the model car is directly related to the voltage; i.e., greater voltage increases the speed of the vehicle. During each test, the voltage and current supplied to the car are kept constant. Due to the energy consumption of the different units in the model car, the voltage is dropped after each test thus resulting in a decrease in the speed of the model car. Further, with increased voltage supply to the motors of the model car, it covers the distance in less time, while decreasing supplied voltage increases the time. For 3000mah battery, this model car works for an average of 45 minutes, in which 85% of the battery is drained by the motors of the model car, and 15% is consumed by Raspberry Pi.

TABLE IX
VOLTAGE DROP OF MODEL CAR AFTER EACH TEST

No.	Voltage	Speed (inches/sec)
01	5.16	13.1
02	5.03	12.5
03	4.96	12
04	4.92	11.13
05	4.86	11.14
06	4.79	10.86
07	4.71	9.67

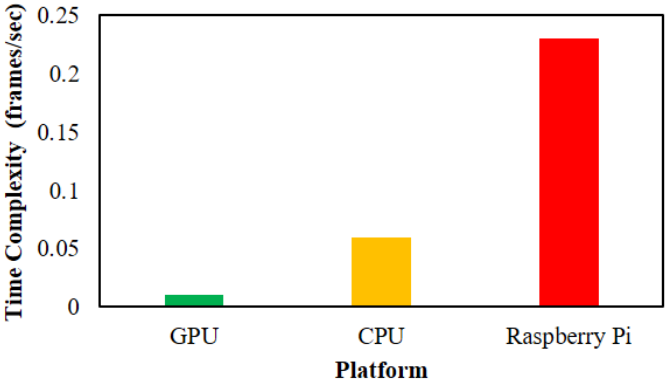

Fig. 18. Average time complexity of frame/sec execution on different platforms.

D. *Comparison with Other State-of-the-Art Methods*

With advancements in technology, the performance of resource-constrained devices, such as Raspberry Pi, have been left behind. Today's autonomous industry mainly focuses on the development of autonomous decision-making capabilities by deferring as the second design driver the optimization of efficiency in terms of the hardware costs. This subsection elaborates on the performance and accuracy of our prototype with respect to the related more expensive technologies.

For comparison with other methods, several experiments were carried out on traffic sign detection using Raspberry Pi. A total of 100 images of different traffic signs are used for testing. Table X presents the traffic sign, the number of test images correctly classified, and the accuracy of our system. In the first test, a total of 25 images of the *stop* sign are passed from the trained Haar based classifier. The system recognized all the 25 images correctly. In the second test, a total of 25 images of the *go* sign are tested on the trained Haar classifier, which detects 24 signs correctly. Results for other traffic signs can be seen in Table X.

TABLE X
PERFORMANCE OF HAAR-BASED CLASSIFIER.

Sign	No. of images	Correctly Classified	Accuracy
Stop	25	25	100%
Go	25	24	99.9%
Left not turn	25	25	100%
Right not turn	25	25	100%
Crosswalk	25	25	100%

These results are compared with the existing techniques to assess the performance of the trained Haar-based classifier. Table XI compares the proposed system with other two existing methods in terms of the accuracy and the execution time for frame processing. The state-of-the-art methods contain several RC-car based self-driving car test-beds. For instance, a study conducted by MIT [54] is based on NVIDIA jetson computing platform and Shim [55]. Both works are based on LIDAR and many other sensors. However, such systems consume excessive energy and power. Also, the employed sensors cost more than $4,000, thus requiring much investments. Compared to these solutions, we propose a CNN-based workload in real time on a resource-constrained and low-cost computing platform, thus providing a cheaper solution for real-time applications.

TABLE XI
COMPARISON OF TRAINED HAAR CASCADE CLASSIFIER WITH EXISTING TECHNIQUES

Method	[56]	[57]	Our Trained Haar-based Classifier
Accuracy	97.75%	97.20%	99.9%
Time	0.003	-	0.02

V. CONCLUSION AND FUTURE WORK

This paper presents a cost-effective and computationally efficient solution for autonomous maneuvering based on resource-constrained devices and a lightweight deep learning model that can be used to facilitate vehicular perception and autonomous guidance in intelligent transportation systems. The proposed system achieves attractive performance scores in terms of the detection and avoidance of obstacles, traffic sign recognition, and intelligently following a smooth trajectory. The assembly of different hardware components, scalar, and vision sensors have their role in the overall output of the system. When compared to other more expensive solutions, our economical and computationally efficient prototype car is capable of autonomously driving on a specified track by avoiding obstacles as well as detecting and recognizing five different traffic signs based on Artificial Intelligence methods. An ultrasonic sensor is used for obstacle detection, which helps avoid collisions, thereby preventing the car from accidents. A Haar cascade classifier is used for traffic sign detection. The car can identify a traffic sign and adjust its speed of wheel motors by using these cascades. An algorithm capable of calculating the distance by using only a monocular vision sensor is used to detect and measure the distance to the traffic signs. Rich

experimental results show that our prototype model car achieves autonomous driving with an overall accuracy of 95.5%

Future research will be devoted to enhancing our algorithm further in order to increase the admissible number of frames per second and accommodate higher car speeds. Specifically, studies in the short term will elaborate on the distance calculation, the consideration of the vehicle sensing capability and overtaking other vehicles, the detection and recognition of traffic lights, and ultimately the optimization of the overall perception results. Moreover, instead of a single Raspberry Pi node, we will scale up the number and the heterogeneity of sensing devices for handling more realistic scenarios of inherently higher complexity. To this end, other input devices like LIDAR sensors will be under consideration in order to scan the surrounding environment for other obstacles. Similarly, an addition of vision sensors on the back of the car model may equip the vehicle with the capability of reverse function and, eventually, make it turn around to avoid possible detected obstacles by harnessing artificial intelligence and computer vision capabilities similar to the ones presented in this work.

REFERENCES

[1] S. K. Gehrig and F. J. Stein, "Dead reckoning and cartography using stereo vision for an autonomous car," in *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, 1999, pp. 1507-1512.

[2] S. Thrun, "Toward robotic cars," *Communications of the ACM*, vol. 53, pp. 99-106, 2010.

[3] B. Schoettle and M. Sivak, "A survey of public opinion about autonomous and self-driving vehicles in the US, the UK, and Australia," 2014.

[4] C. W. Company, "Baidu just made its 100th autonomous bus ahead of commercial launch in China".

[5] H. Woo, Y. Ji, H. Kono, Y. Tamura, Y. Kuroda, T. Sugano, *et al.*, "Lane-change detection based on vehicle-trajectory prediction," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1109-1116, 2017.

[6] K. Jo and M. Sunwoo, "Generation of a precise roadway map for autonomous cars," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, pp. 925-937, 2014.

[7] M. Althoff and A. Mergel, "Comparison of Markov chain abstraction and Monte Carlo simulation for the safety assessment of autonomous cars," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, pp. 1237-1247, 2011.

[8] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Transactions on Vehicular Technology*, vol. 63, pp. 540-555, 2014.

[9] A. Borkar, M. Hayes, and M. T. Smith, "A novel lane detection system with efficient ground truth generation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 365-374, 2012.

[10] H. Yoo, U. Yang, and K. Sohn, "Gradient-enhancing conversion for illumination-robust lane detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 1083-1094, 2013.

[11] S. Sivaraman and M. M. Trivedi, "Integrated lane and vehicle detection, localization, and tracking: A synergistic approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, pp. 906-917, 2013.

[12] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic* vol. 56: springer, 2009.

[13] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, 2011, pp. 163-168.

[14] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of field Robotics*, vol. 23, pp. 661-692, 2006.

[15] C.-Y. Lee, C.-T. Lin, C.-T. Hong, and M.-T. Su, "Smoke detection using spatial and temporal analyses," *International Journal of Innovative Computing, Information and Control*, vol. 8, pp. 4749-4770, 2012.

[16] W. J. Schakel, B. Van Arem, and B. D. Netten, "Effects of cooperative adaptive cruise control on traffic flow stability," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 2010, pp. 759-764.

[17] B. v. Arem, C. J. G. v. Driel, and R. Visser, "The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, pp. 429-436, 2006.

[18] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, *et al.*, "3D object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, 2015, pp. 424-432.

[19] R. Guidolini, C. Badue, M. Berger, L. de Paula Veronese, and A. F. De Souza, "A simple yet effective obstacle avoider for the IARA autonomous car," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, 2016, pp. 1914-1919.

[20] J. Choi, J. Lee, D. Kim, G. Soprani, P. Cerri, A. Broggi, *et al.*, "Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 974-982, 2012.

[21] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, pp. 727-774, 2008.

[22] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182-193, 2016.

[23] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, pp. 123-139, 2009.

[24] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806-813.

[25] S. Khan, K. Muhammad, S. Mumtaz, S. W. Baik, and V. H. C. de Albuquerque, "Energy-efficient deep CNN for smoke detection in foggy IoT environment," *IEEE Internet of Things Journal*, 2019.

[26] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[27] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, pp. 1533-1545, 2014.

[28] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[29] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844-3852.

[30] Y. S. Wong, N. K. Lee, and N. Omar, "GMFR-CNN: an integration of gapped motif feature representation and deep learning approach for enhancer prediction," in *Proceedings of the 7th International Conference on Computational Systems-Biology and Bioinformatics*, 2016, pp. 41-47.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[32] M. Sajjad, S. Khan, K. Muhammad, W. Wu, A. Ullah, and S. W. Baik, "Multi-grade brain tumor classification using deep CNN with extensive data augmentation," *Journal of computational science*, vol. 30, pp. 174-182, 2019.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.

[34] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, pp. 640-651, 2017.

[35] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017, pp. 1370-1377.

[36] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[37] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722-2730.

[38] T. Sun, S. Tang, J. Wang, and W. Zhang, "A robust lane detection method for autonomous car-like robot," in *Intelligent Control and Information Processing (ICICIP), 2013 Fourth International Conference on*, 2013, pp. 373-378.

[39] A. Saha, D. D. Roy, T. Alam, and K. Deb, "Automated road lane detection for intelligent vehicles," *Global Journal of Computer Science and Technology*, 2012.

[40] T. M. Hoang, H. G. Hong, H. Vokhidov, and K. R. Park, "Road lane detection by discriminating dashed and solid road lanes using a visible light camera sensor," *Sensors*, vol. 16, p. 1313, 2016.

[41] M. Sajjad, I. U. Haq, J. Lloret, W. Ding, and K. Muhammad, "Robust Image Hashing based Efficient Authentication for Smart Industrial Environment," *IEEE Transactions on Industrial Informatics*, pp. 1-1, 2019.

[42] K. M. A. Alheeti, R. Al-Zaidi, J. Woods, and K. McDonald-Maier, "An intrusion detection scheme for driverless vehicles based gyroscope sensor profiling," in *Consumer Electronics (ICCE), 2017 IEEE International Conference on*, 2017, pp. 448-449.

[43] K. M. A. Alheeti, A. Gruebler, K. D. McDonald-Maier, and A. Fernando, "Prediction of DoS attacks in external communication for self-driving vehicles using a fuzzy petri net model," in *Consumer Electronics (ICCE), 2016 IEEE International Conference on*, 2016, pp. 502-503.

[44] P. Gora and I. Rüb, "Traffic models for self-driving connected cars," *Transportation Research Procedia*, vol. 14, pp. 2207-2216, 2016.

[45] Q. Huy, S. Mita, H. T. N. Nejad, and L. Han, "Dynamic and safe path planning based on support vector machine among multi moving obstacles for autonomous vehicles," *IEICE TRANSACTIONS on Information and Systems*, vol. 96, pp. 314-328, 2013.

[46] T. Birdal and A. Erçil, "Real-time automated road, lane and car detection for autonomous driving," 2007.

[47] B. Grush and J. Niles, *The end of driving: transportation systems and public policy planning for autonomous vehicles*: Elsevier, 2018.

[48] T. Litman, *Autonomous vehicle implementation predictions*: Victoria Transport Policy Institute Victoria, Canada, 2017.

[49] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.

[50] J. Chu, L. Ji, L. Guo, Libibing, and R. Wang, "Study on method of detecting preceding vehicle based on monocular camera," in *IEEE Intelligent Vehicles Symposium, 2004*, 2004, pp. 750-755.

[51] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, pp. I-I.

[52] M. Sajjad, M. Nasir, K. Muhammad, S. Khan, Z. Jan, A. K. Sangaiah, *et al.*, "Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities," *Future Generation Computer Systems*, 2017.

[53] A. Vladišauskas and L. Jakevičius, "Absorption of ultrasonic waves in air," *Ultragarsas*, vol. 50, pp. 46-49, 2004.

[54] S. Karaman, A. Anders, M. Boulet, J. Connor, K. Gregson, W. Guerra, *et al.*, "Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT," in *Integrated STEM Education Conference (ISEC), 2017 IEEE*, 2017, pp. 195-203.

[55] I. Shim, J. Choi, S. Shin, T.-H. Oh, U. Lee, B. Ahn, *et al.*, "An autonomous driving system for unknown environments using a unified map," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 1999-2013, 2015.

[56] Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 2022-2031, 2016.

[57] F. Zaklouta, B. Stanciulescu, and O. Hamdoun, "Traffic sign classification using kd trees and random forests," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011, pp. 2151-2155.

Muhammad Sajjad received his Master degree from Department of Computer Science, College of Signals, National University of Sciences and Technology, Rawalpindi, Pakistan. He received his PhD degree in Digital Contents from Sejong University, Seoul, Republic of Korea. He is now working as an associate professor at Department of Computer Science, Islamia College Peshawar, Pakistan. He is also head of "Digital Image Processing Laboratory (DIP Lab)" at the same university. His research interests include digital image super-resolution and reconstruction, medical image analysis, video summarization and prioritization, image/video quality assessment, computer vision, and image/video retrieval.

Muhammad Irfan received his BS degree in Computer Science from Islamia College, Peshawar, Pakistan. He is currently a Master student at Department of Software, Sejong University, Seoul, Republic of Korea. His research interest includes image and video processing, intelligent transportation systems, computer vision, machine learning, and deep learning.

Khan Muhammad (S'16-M'18) received the Ph.D degree in Digital Contents from Sejong University, South Korea. He is currently working as an Assistant Professor at Department of Software and Lead Researcher of Intelligent Media Laboratory, Sejong University, Seoul. His research interests include intelligent video surveillance (fire/smoke scene analysis, transportation systems, and disaster management), medical image analysis (brain MRI, diagnostic hysteroscopy, and wireless capsule endoscopy), information security (steganography, encryption, watermarking, and image hashing), video summarization (single-view and multi-view), multimedia, computer vision, IoT, and smart cities. He has registered over 7 patents and published over 100 papers in peer-reviewed international journals and conferences in these research areas with target venues as IEEE COMMAG, NETWORK, TII, TIE, TSMC-Systems, IoTJ, Access, TSC, Elsevier INS, Neurocomputing, PRL, FGCS, ASOC, IJIM, SWEVO, COMCOM, COMIND, JPDC, PMC, BSPC, CAEE, Springer NCAA, MTAP, JOMS, and RTIP, etc. He is also serving as a professional reviewer for over 70 well-reputed journals and conferences. He is currently involved in editing of several special issues as GE/LGE. He is a member of the IEEE and ACM.

Javier (Javi) Del Ser (M'07–SM'12) received his first PhD in Telecommunication Engineering (Cum Laude) from the University of Navarra, Spain, in 2006, and a second PhD in Computational Intelligence (Summa Cum Laude, Extraordinary Prize) from the University of Alcala, Spain, in 2013. He has held several positions as a Professor and a Researcher at different institutions of the Basque Research Network (including the University of Mondragon, CEIT and Robotiker). Currently he is a Research Professor in Data Analytics and Optimization at TECNALIA (Spain), a visiting fellow at the Basque Centre for Applied Mathematics (BCAM) and an adjunct professor at the University of the Basque Country (UPV/EHU). He is also a Senior AI advisor at the technological startup Sherpa. His research interests gravitate on the use of descriptive, predictive and prescriptive algorithms for data mining and optimization in a diverse range of application fields such as Energy, Transport, Telecommunications, Health and Industry, among others. In these fields he has published more than 300 scientific articles, co-supervised 10 Ph.D. thesis, edited 7 books, co-authored 9 patents and participated/led more than 43 research projects. He serves as an associate editor in a number of indexed journals, including Information Fusion, Swarm and Evolutionary Computation, Cognitive Computation and IEEE Transactions on Intelligent Transportation Systems.

Dr. Javier Sanchez-Medina is currently an Associate Professor in the Computer Science department at the University of Las Palmas de Gran Canaria (ULPGC), Spain. Dr. Sanchez-Medina earned his Engineering Master Degree at the Telecommunications Faculty on 2002, and his PhD at the Computer Science Department on 2008. His PhD dissertation versed on the use of Genetic Algorithms, Parallel Computing and Cellular Automata based Traffic Microsimulation to optimize the Traffic Lights Programming within an Urban Traffic Network. His research interests include mainly the application of Data Mining, Evolutionary Computation and Parallel Computing to Intelligent Transportation Systems, in particular to Traffic Modeling and Prediction. Javier Sanchez-Medina has been volunteering for several years at many international conferences related to Intelligent Transportation, Computer Science, Evolutionary Computation, etc. He is reviewer for some Transportation related journals like the IEEE ITSS Transactions, or the IEEE Transactions on Vehicular Technology. Since 2010, we has served for the IEEE ITS Society organizing the TBMO 2010 Workshop at ITSC2010, co-organizing the "Travel Behavior Research: Bounded Rationality and Behavioral Response" Special Session at ITSC2011, being Publications Chair at the IEEE FISTS2011, Registration Chair at the IEEE ITSC2012 and Workshops and Tutorials Chair for IEEE ITSC 2013, Panels Chair for IEEE VTC-Fall 2013, Program co-chair for IEEE ITSC2014 and IEEE ITSC2016, Publicity Chair for IEEE IV2017, Program Chair for IEEE ICVES 2017 and Program co-Chair for IEEE ITSC2018. He served as general chair for the IEEE ITSC2015, a record beating edition of IEEE ITSCs, hosted at Las Palmas de Gran Canaria (Spain) on September 2015. He has also contributed to the IEEE ITS Society as Founding Editor in Chief of the IEEE ITS Podcast from May 2013 to December 2016, and Editor in Chief of the IEEE ITS Newsletter during 2015 and 2016. He has recently been appointed as Vice President for Technical Activities for the IEEE ITS Society and President of its Spanish Chapter for the term 2017-2018. Before of that he served as Vice President of that Spanish Chapter during 2015 and 2016. He has widely published his research with more than 30 international conference articles and more than 20 international journal articles and 3 research chapters, being the

main author of more than half of them. He has also been keynote speaker at two conferences (ANT2017, SOLI2017) and distinguished lecturer at the IEEE ITSS Tunisian Chapter in November, 2016.

Sergey Andreev [SM'17] (sergey.andreev@tuni.fi) is an assistant professor of communications engineering and Academy Research Fellow at Tampere University, Finland. Since 2018, he has also been a Visiting Senior Research Fellow with the Centre for Telecommunications Research, King's College London, UK. He received his Ph.D. (2012) from TUT as well as his Specialist (2006) and Cand.Sc. (2009) degrees from SUAI. He serves as editor for IEEE Wireless Communications Letters (2016-) and as lead series editor of the IoT Series (2018-) for IEEE Communications Magazine. He (co-)authored more than 200 published research works on intelligent IoT, mobile communications, and heterogeneous networking.

Weiping Ding (M'16-SM'19) received the Ph.D. degree in Computation Application, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2013, and was awarded the Outstanding Doctoral Dissertation by NUAA. He was a Visiting Scholar at University of Lethbridge(UL), Alberta, Canada, in 2011. From 2014 to 2015, He is a Postdoctoral Researcher at the Brain Research Center, National Chiao Tung University (NCTU), Hsinchu, Taiwan. In 2016, He was a Visiting Scholar at National University of Singapore (NUS), Singapore. From 2017 to 2018, he was a Visiting Professor at University of Technology Sydney (UTS), Ultimo, NSW, Australia. He is a member of Senior IEEE, IEEE-CIS, ACM, IAENG and Senior CCF. Now, Dr. Ding is the Chair of IEEE CIS Task Force on Granular Data Mining for Big Data. He is a member of Technical Committee on Soft Computing of IEEE SMCS, a member of Technical Committee on Granular Computing of IEEE SMCS, a member of Technical Committee on Data Mining and Big Data Analytics Technical Committee of IEEE CIS. He also is a member of IEEE CIS Task Force on Adaptive and Evolving Fuzzy Systems.

Dr. Ding's main research directions involve deep learning, data mining, evolutionary computing, granular computing, machine learning and big data analytics. He has published over 50 papers in flagship journals and conference proceedings as the first author, including IEEE Transactions on Fuzzy Systems, IEEE Transactions on Neural Network and Learning System, IEEE Transactions on Cybernetics, IEEE Transactions on Systems, Man, and Cybernetics: Systems, IEEE Transactions on Emerging Topics in Computational Intelligence. To data, he has held 12 approved invention patents in total over 20 issued patents. Dr. Ding was a recipient of Computer Education Excellent Paper Award (First-Prize) from the National Computer Education Committee of China, in 2009. He was an Excellent-Young Teacher (Qing Lan Project) of Jiangsu Province in 2014, and a High-Level Talent (Six Talent Peak) of Jiangsu Province in 2016. He was awarded the Best Paper of ICDMA'15, and awarded an Outstanding Teacher of Software Design and Entrepreneurship Competition by the Ministry of Industry and Information Technology, China, in 2017. Dr. Ding was a recipient of the Medical Science and Technology Award (Second Prize) of Jiangsu Province, China, in 2017, and the Education Teaching and Research Achievement Award (Third Prize) of Jiangsu Province, China, in 2018. He was a recipient of Natural Science Outstanding Academic Paper Award (First Prize), Nantong, China, in 2017, Science and Technology Progress Award (Second Prize), Nantong, China, in 2018, Outstanding Associate Editor of 2018 for IEEE Access Journal. Dr. Ding was awarded two Chinese Government Scholarships for Overseas Studies in 2011 and 2016.

Dr. Ding currently serves on the Editorial Advisory Board of Knowledge-based Systems and Editorial Board of Information Fusion. He serves/served as an Associate Editor of several prestigious journals, including IEEE Transactions on Fuzzy Systems, Information Sciences, Swarm and Evolutionary Computation, IEEE Access and Journal of Intelligent & Fuzzy Systems, Co-Editor-in-Chief of Journal of Artificial Intelligence and Systems, as well as the lead guest editor in several international journals. He serves/served as a program committee member for several international conferences and workshops.

Jong Weon Lee received M.S. degree in Electrical and Computer Engineering from University of Wisconsin at Madison in 1991, and Ph.D. degree from University of Southern California in 2002. He is presently Professor of Department of Software at Sejong University. His research interests include augmented reality, computer vision, machine learning, human-computer interaction and serious game.