Mohammed, MS, Abduljabar, AM, Faisal, MM, Mahmmod, BM, Abdulhussain, SH, Khan, W, Liatsis, P and Hussain, A

Low-cost autonomous car level 2: Design and implementation for conventional vehicles

Results in Engineering 17 (2023) 100969

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering

Low-cost autonomous car level 2: Design and implementation for conventional vehicles

Mohammad S. Mohammed[a], Ali M. Abduljabar[a], Mustafa M. Faisal[a], Basheera M. Mahmmod[a], Sadiq H. Abdulhussain[a], Wasiq Khan[b], Panos Liatsis[c,*], Abir Hussain[b,d,**]

[a] Department of Computer Engineering, University of Baghdad, Al-Jadriya, Baghdad, 10071, Iraq
[b] School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool, UK
[c] Department of Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates
[d] Department of Electrical Engineering Department, University of Sharjah, Sharjah, United Arab Emirates

ARTICLE INFO

Keywords:
Self-driving vehicle
Deep learning
Image processing

ABSTRACT

Modern cars are equipped with autonomous systems to assist the driver and improve driving experience. Driving assist system (DAS) is one of the most significant components of a self-driving vehicle (SDV), used to overcome non-autonomous driving challenges. However, most conventional cars are not equipped with DAS, and high-cost systems are required to equip these vehicles with DAS. Moreover, the design of DAS is very complex outside of the industry while it requires going through the Electronic Control Unit (ECU), which has a high level of security. Therefore, a basic system needs be installed in conventional cars which makes driving more efficient in terms of driver assistance. In this paper, an intelligent DAS is presented for real-time prediction of steering angle using deep learning (DL) and raw dataset collected from a real environment. Furthermore, an object detection model is deployed to assist and warn the driver of various types of objects along with corresponding distance measurement based on DL. Outputs from DL models are fed into the steering control system, which has Electronic Power Steering (EPS). The steering angle is measured in real time using an angle sensor and is posted back to the steering control system to make automated adjustments accordingly. Real-time tests are conducted on a 2009 Toyota Corolla equipped with a digital camera to capture live video stream, Controller Area Network (CAN-BUS) messages, and a steering angle sensor. The performance evaluation of the proposed system indicates intelligent steering control and driver assistance when evaluated in a real-time environment.

1. Introduction

Self-driving vehicle (SDV) is considered game changer in the intelligent transportation and smart city context due to the significant role of artificial intelligent (AI) during the recent years [1,2]. There are significant positive impacts of SDV such as reducing traffic jam, saving fuel, and fewer accidents due to human error factor [3]. Thus, SDV has rapidly emerged among the hot researched topics recently and has become a significant element of the industry, where experts predict that it will have a significant impact on our society [4]. Composite of technologies are required to build an autonomous vehicle that include laser sensors, radar sensors, GPS, camera devices, and computer vision algorithms for the real time processing of data captured through these devices. Camera devices, compared with other types of sensors such as LIDAR, are inexpensive and able to capture additional information while driving [5]. The goals of SDV are reducing the road accidents and deaths, time consumption and driving time, and traffic reduction [6]. The reduction in road accident and deaths are performed by implementing SDV on public road. SDVs are equipped with intelligent capabilities that reduce road accidents by up to 90% due to intelligent decision-making in dangerous conditions [7], which are usually caused by human errors [8]. In addition, an efficient route to the destination using built-in route optimization methods will reduce carbon-dioxide emissions by 60% [8–10]. The reduction of time consumption and driving time can be performed via intelligent systems using efficient trajectory planning [6]. Compared to human driver, autonomous vehicles can save up to 1 billion hours per day [7]. Furthermore, SDV will reduce the predicted future traffic resulted from increasing world population in the urban cities which is a substantial

* Corresponding author.
** Corresponding author. School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool, UK.
E-mail addresses: panos.liatsis@ku.ac.ae (P. Liatsis), abir.hussain@sharjah.ac.ae (A. Hussain).

https://doi.org/10.1016/j.rineng.2023.100969
Received 23 September 2022; Received in revised form 14 February 2023; Accepted 16 February 2023
Available online 1 March 2023

List of abbreviations

DAS	Driving assistant system
SDV	Self-Driving Vehicle
EPS	Electronic Power Steering
ECU	Electronic Control Unit
CAN-BUS	Controller Area Network Bus
AI	Artificial intelligent
GPS	Global Positioning System
LIDAR	Light Detection and Ranging
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
SCS	steering control system
PWM	Pulse Width Modulation
GND	Ground
GPIO	General-purpose input/output
USB	Universal Serial Bus

future issue [11,12].

Several pilot studies related to SDV have been introduced in recent years [4,13–15]. One aspect these pilot projects have in common is that algorithms based on deep learning are used to perform certain driving tasks, such as environmental awareness, lane planning, and even steering wheel control. With the successful demonstration of autonomous prototypes based on deep learning, the focus of the automotive industry is steadily shifting from vehicle prototype development to serial production. However, the main challenge is to integrate neural networks into serial manufacturing vehicles in a safe and reliable way.

In [35], the dataset was collected on a self-made track, and the convolutional neural network model was trained on that dataset. A toy car was crafted and used for testing the convolutional neural network model. This method can work but it is far from testing a model in a virtual environment in terms of difficulty and cost. In Ref. [16], three cameras were used to record and build the dataset, both in real-world and in the simulated environment. In Ref. [17], a deep learning-based approach is presented for lane keeping in autonomous vehicles. The framework helps the vehicle stay in its lane and avoid colliding with other vehicles or obstacles. An end-to-end learning framework is used to trains the deep neural network to predict the steering angle of the vehicle based on the input from a front-facing camera. In Ref. [18], a self-driving delivery car named Delicar is presented. The presented system is designed to deliver products in the context of Bangladesh. The car is equipped with deep learning algorithms that allow it to perform lane keeping, object detection, and collision avoidance. The paper presented Diclar to improve delivery efficiency and increase safety. The dataset is from Udacity simulator. In Ref. [19], an object detection system is presented by utilizing an improved version of the YOLO v5 deep learning architecture. The system detects vehicles and estimates their distances from the observer. This work was evaluated using real-world data captured from a driving scenario. However, it didn't present any steering control system and the dataset used is limited. In Ref. [16], a lane following based on deep-learning network is presented for vehicles. The deep neural network architecture can detect lanes and steering the vehicle accordingly. The network was trained and evaluated on a driving scenes dataset. A method for steering control of a vehicle is equipped with an automated lane centering system is presented in Ref. [20]. The objective is to maintain the vehicle in the center of the lane while driving on the road. The work described the implementation of the automated lane centering system and its impact on the steering control of the vehicle. To predict the steering angle, a CNN is used in Ref. [21]. The goal is to improve the accuracy and stability of the vehicle's steering control system, which is a crucial aspect of autonomous driving. A deep reinforcement learning approach is presented for controlling the actions of an autonomous vehicle in Ref. [22]. While, a method for controlling the steering of autonomous vehicles is presented in Ref. [23]. The method combines deep learning, a type of machine learning, with model predictive control, a control strategy for systems modeled by differential equations. The goal is to produce a control system that can make decisions based on both prediction and control. The paper outlines the process and experiments used to evaluate the method and demonstrates that it is effective for controlling the steering of autonomous vehicles. Moreover, CNN is used to predict the steering angle and acceleration of the vehicle based on input from various sensors, and then use reinforcement learning to optimize the network's decision-making based on the vehicle's performance. In Ref. [24], the use of deep reinforcement learning (RL) is described for controlling autonomous vehicles. The authors use a simulation environment, CARLA, to train and evaluate the RL-based control system for steering. The goal of the paper is to demonstrate the feasibility and effectiveness of using RL for steering control in autonomous vehicles. Most of the previous work are costly and the old cars are not transformed into autonomous cars since the cost of producing a SDV is high. The first Velodyne automotive lidar units (64 or 128 lasers) were available at an approximate cost of about $75,000 per unit [25]. In contrast, the proposed work aims to transform a conventional vehicle into an autonomous car with low cost by using single camera device. In addition, the proposed algorithm is implemented in controlled and real environments (on the road) to validate the proposed methodology. For this purpose, a primary dataset is collected and analyzed for solving the problem of autonomous driving. The contributions of the proposed work can be summarized as follows.

1. Develop a low-cost solution to convert a conventional car into an autonomous car using a single camera device.
2. Implementation of the proposed autonomous driving algorithm in realistic and controlled environments to verify its effectiveness which is performed by using Artificial Neural Network (ANN) for image processing data analysis.
3. Collecting and analyzing primary data to standardize it for neural network modeling. This is performed by the utilization of Snifter, an open-source program, to capture and display actuator data. Also, the use of an angle sensor to capture car's steering angle since it was not embedded from the manufacturer. In addition, we utilize NRF24L01 as a sender and receiver to wirelessly send steering angle information.
4. Using of Controller Area Network (CAN-BUS) protocol to connect the Electric Control Unit (ECU) and actuators of the car. The design and implementation of a hardware circuit to interface with the CAN network using a CAN-BUS shield.

The remainder of the manuscript is organized as follows. Section 2 provides the details of the proposed methodology with a description of the dataset. Section 3 presents the simulation results with a discussion, while Section 4 illustrates the conclusion and future works.

2. Materials and methods

The detail of the presented methodology and the materials used are given in the following subsections.

2.1. Dataset

Along with the development of SDV, data collection on public highways has been deemed a valuable activity to complete the training of CNN model and deploy it on-road. In this work, we attempted to gather and build primary dataset. This is mainly because of substantial differences between published datasets and our dataset, such as road marking, contain a numerous cracks and old bumps (which is not seen mostly in well-developed structures). While the car is being driven

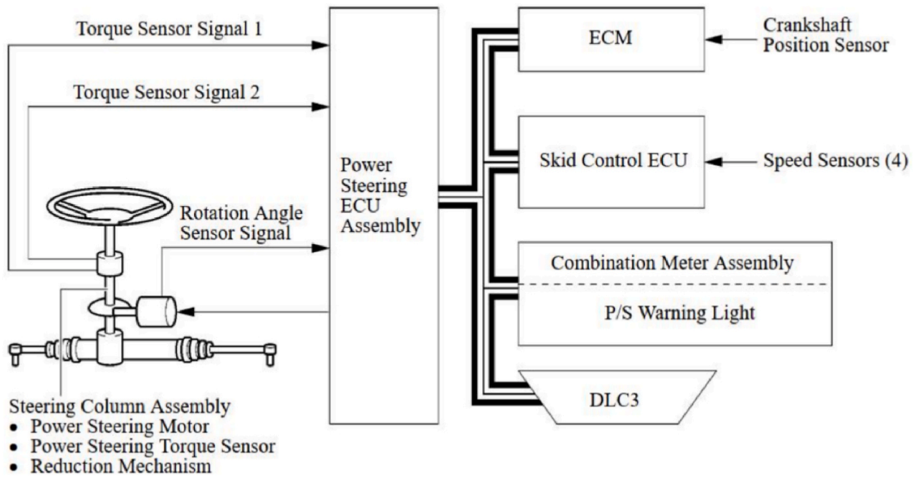


Fig. 1. System diagram of the main component used in the utilized car.

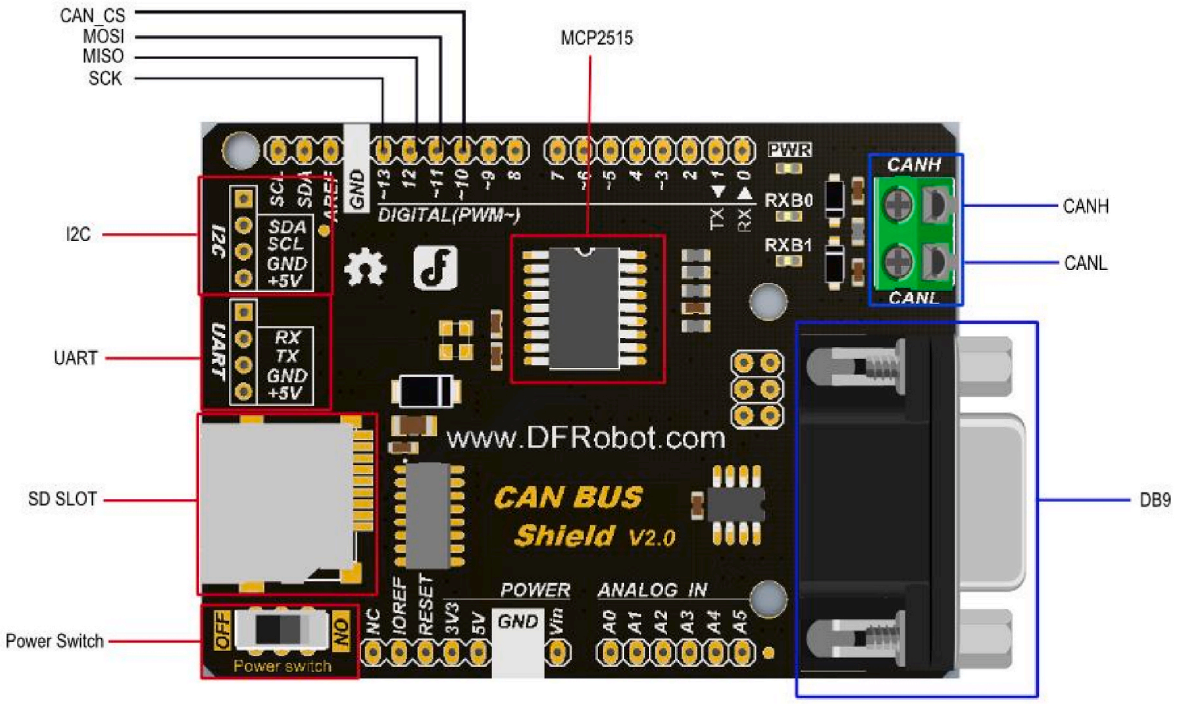


Fig. 2. CAN-BUS shield from DFRobot.

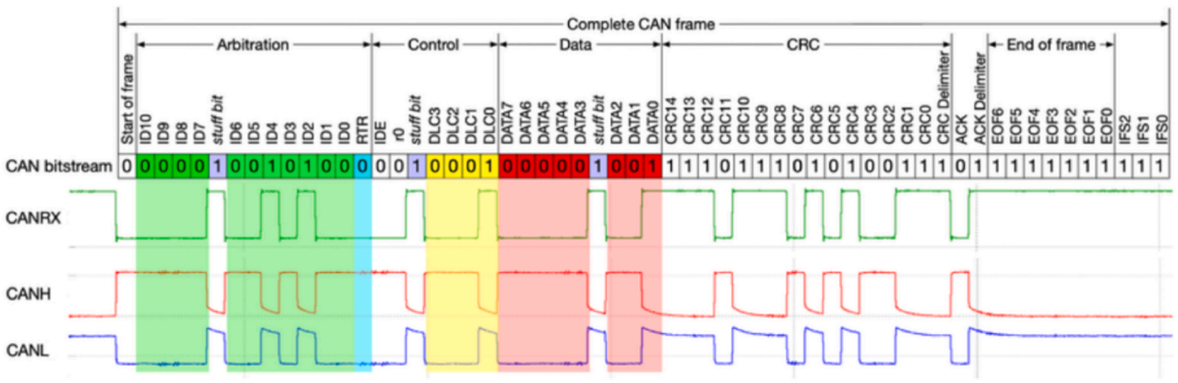


Fig. 3. Sample of CAN-BUS packet shape.

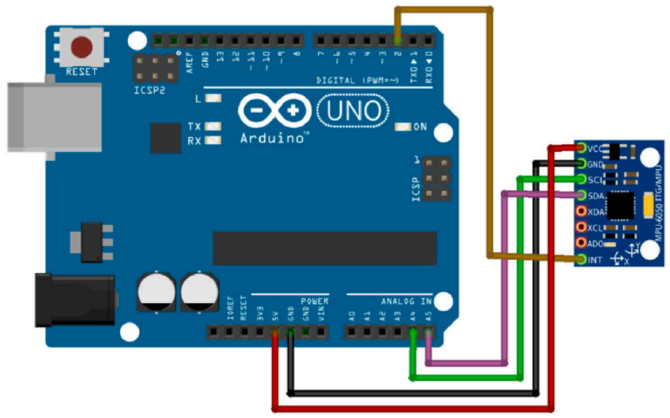


Fig. 4. Wiring diagram to Arduino UNO with MPU 6050.

manually, the raw data from all sensors is acquired and stored. The data obtained is used for training and testing purposes, such as vehicle/pedestrian identification, and motion estimates [26].

To obtain the training data, the car is driven on the campus road of University of Baghdad in a variety of lighting and weather conditions. The campus road has two lanes, while in some sections it contains three road lanes. It also has some cracked on it which is obtained within the collected data. In addition, the data was acquired in different weather conditions as the sun was low in the sky in some cases, causing glare to reflect off the road surface and scatter the camera.

A Toyota Corolla 2009 LE class with one camera was used to acquire the data. It is worth noting that a specific make or model of vehicle is not required for our proposed system to work. This car, like other cars, has ECU and several actuators are distributed at various car locations and the Engine (please refer to Fig. 1). To make a connection between the ECU and the actuators, some communication protocols are used to manage the information sharing between the devices and this protocol called Controller Area Network (CAN-BUS) protocol. CAN-BUS protocol, developed via Bosch company, is employed to replace the direct wiring connection between ECU and actuators. Moreover, a hardware circuit is designed to use the CAN network where CAN-BUS shield is required. This interface will provide reliable access to the network and with the correct wiring and configuration, we will be able to read and write messages to the network. The CAN-BUS Arduino shield v2.0 is employed for the Arduino Microcontroller. It is compatible with the Arduino standard interface and can be stacked on an Arduino UNO, Arduino Leonardo or Arduino MEGA board. The shield integrates the MCP2515 CAN-BUS chip on the shield and has a CAN-BUS transceiver function. With an on-board DB9 and CAN-BUS connector it is possible to select a suitable port according to the host device. There is also an integrated MicroSD socket for data storage which is a perfect solution for data logging applications (The CAB_BUS shield is shown in Fig. 2).

The CAN-BUS network has two lines, CAN-LOW (CANL) and CAN HIGH (CANH). Sample of CAN-BUS packet shape is shown in Fig. 3.

Snifter program, an open source provided by Adam Verga, is used to record data, and display them on the computer. The captured actuators are Throttle position, Signal left and right, and Vehicle speed. For steering angle, the Corolla LE class is used for economic purpose. This class comes with less sensors and navigation system with minimum features; therefore, the steering angel sensor are not embedded from manufacturer. An angle sensor is used to capture the steering angle. The interface must provide the controlling environment on the part that comes in the car, for this reason, controlling the steering system is one goal of this interface. MPU 6050 is used to capture the position of the shaft. Since it is a gyroscope, the data comes from it provides the position in three-dimensional space, using Cartesian axis X, Y and Z. Also, we can present the data with Aircraft principal axes Yaw, Pitch, and Roll.

The MPU 6050 uses I2C communication protocol to communicate with the controllers that support this protocol, in this case, the Arduino uno has only one interface for this protocol. In addition to I2C, the MPU 6050 has a First in First Out "FIFO" buffer with 1024-bytes, an interrupt is required to read the data from this FIFO buffer, so that we used the INT pin with I2C configuration to read the data from buffer as shown in Fig. 4.

The wiring from steering angle sensor is not recommended because the rotation of the steering will affect and lopped the wire. Thus, an NRF24L01 as sender and receiver is used to send the information without wiring as illustrated in Fig. 5.

All drivers are encouraged to maintain full attentiveness and drive normally when collecting the data in which about 400,000 frames were captured. To understand how the recording system works, refer to Fig. 6.

To summarize the workflow of the system presented, a single camera was installed on top of the vehicle, with a gyro-scope sensor attached to the steering wheel that reads the steering wheel angle (MPU-650). The car was driven one lap at a time to collect and record the data. Frames were captured using the camera. Using the MPU-6050 sensor and an Arduino NANO, the steering angle were recorded. The vehicle speed and throttle position were collected using the CAN-BUS network with a CAN-BUS shield and an Arduino UNO. To achieve the best results from these sensors, a program was designed having the following requirements.

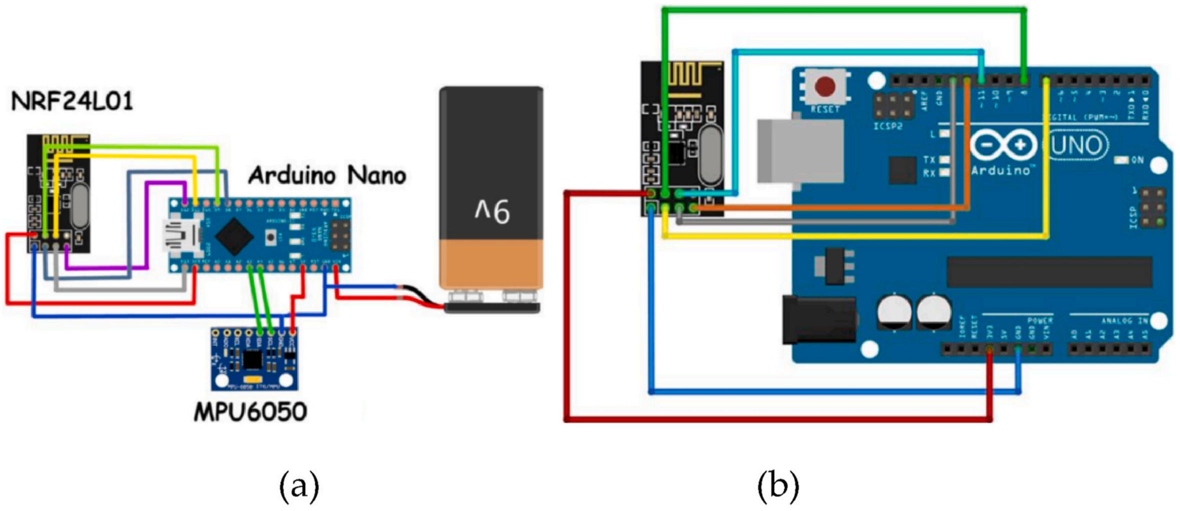


Fig. 5. Wiring diagram for (a) Arduino NANO with MPU 6050, NRF24L01 and 9v power source (sender), and (b) for NRF24L01 and Arduino UNO (receiver).

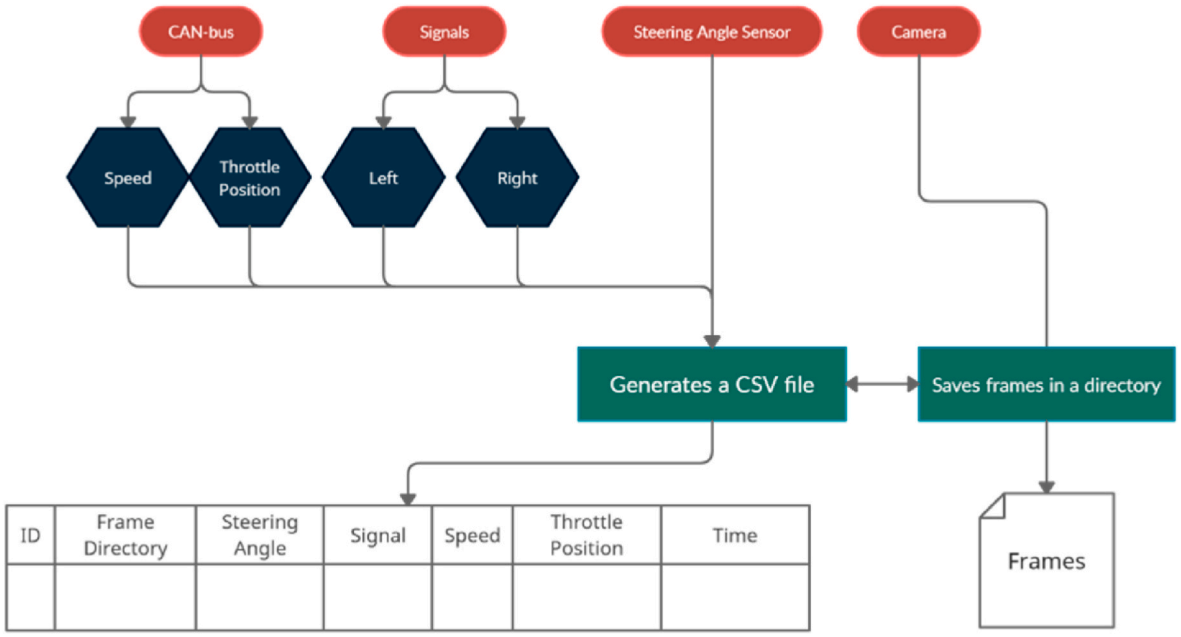


Fig. 6. Dataset recording system.

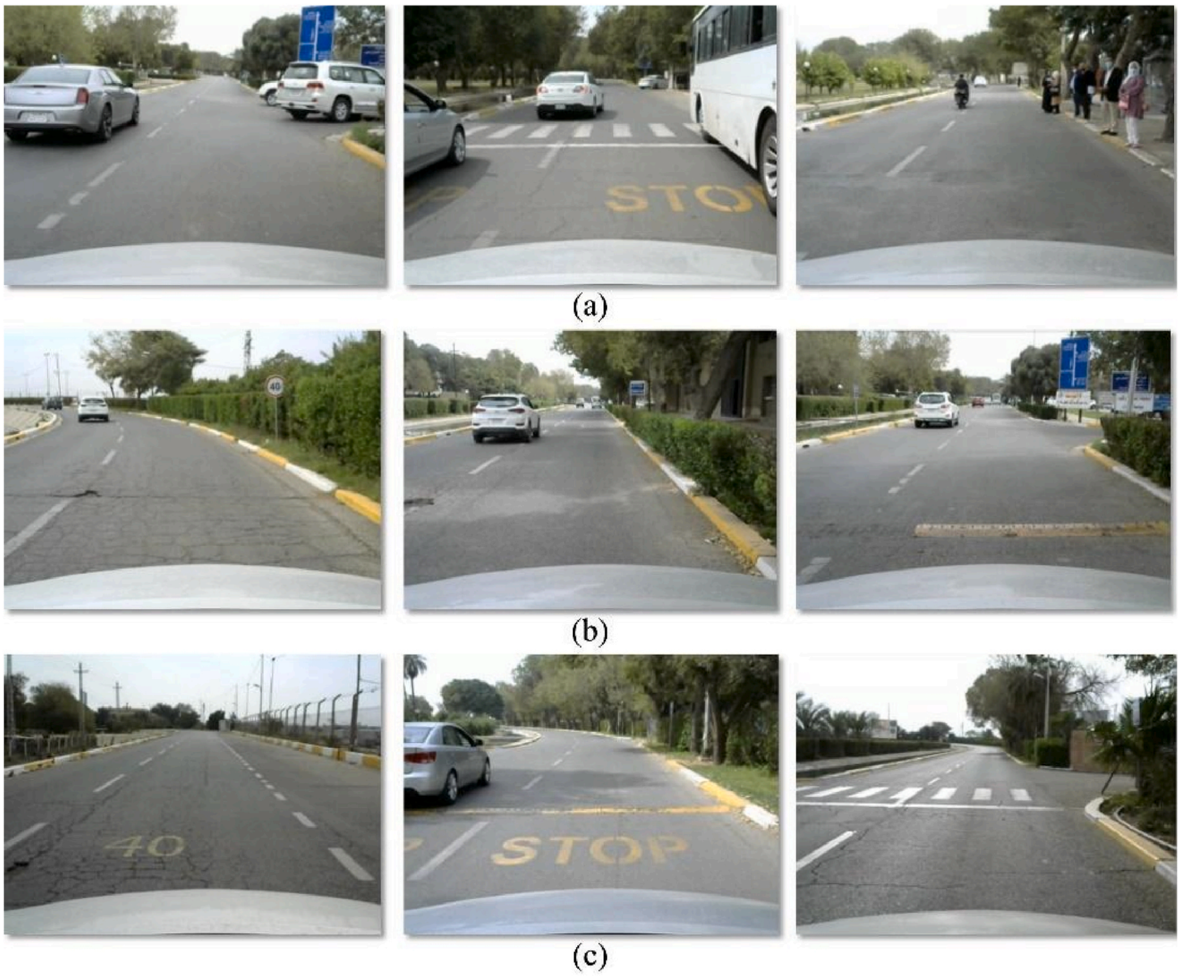
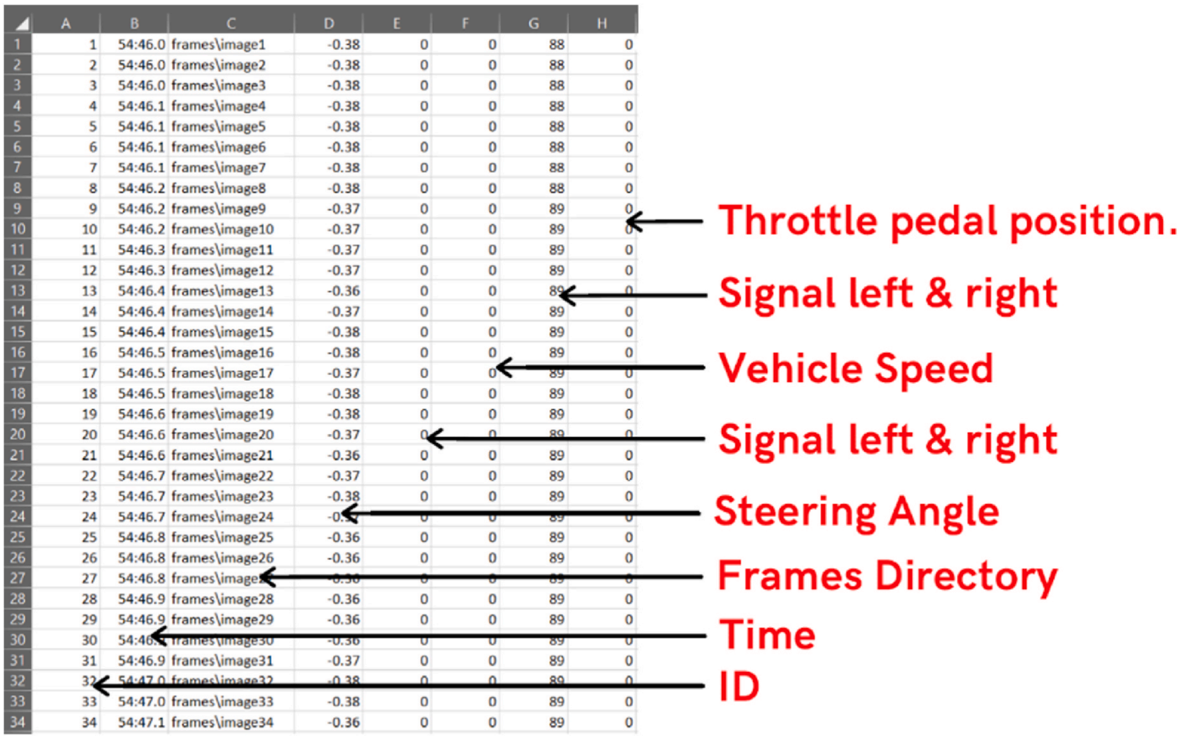


Fig. 7. (a) Different types of road traffic and pedestrians, (b) Different types of road cracks and bumps, and (c) Different types of road signs and markings.



	A	B	C	D	E	F	G	H
1	1	54:46.0	frames\image1	-0.38	0	0	88	0
2	2	54:46.0	frames\image2	-0.38	0	0	88	0
3	3	54:46.0	frames\image3	-0.38	0	0	88	0
4	4	54:46.1	frames\image4	-0.38	0	0	88	0
5	5	54:46.1	frames\image5	-0.38	0	0	88	0
6	6	54:46.1	frames\image6	-0.38	0	0	88	0
7	7	54:46.1	frames\image7	-0.38	0	0	88	0
8	8	54:46.2	frames\image8	-0.38	0	0	88	0
9	9	54:46.2	frames\image9	-0.37	0	0	89	0
10	10	54:46.2	frames\image10	-0.37	0	0	89	0
11	11	54:46.3	frames\image11	-0.37	0	0	89	0
12	12	54:46.3	frames\image12	-0.37	0	0	89	0
13	13	54:46.4	frames\image13	-0.36	0	0	89	0
14	14	54:46.4	frames\image14	-0.37	0	0	89	0
15	15	54:46.4	frames\image15	-0.38	0	0	89	0
16	16	54:46.5	frames\image16	-0.38	0	0	89	0
17	17	54:46.5	frames\image17	-0.37	0	0	89	0
18	18	54:46.5	frames\image18	-0.38	0	0	89	0
19	19	54:46.6	frames\image19	-0.38	0	0	89	0
20	20	54:46.6	frames\image20	-0.37	0	0	89	0
21	21	54:46.6	frames\image21	-0.36	0	0	89	0
22	22	54:46.7	frames\image22	-0.37	0	0	89	0
23	23	54:46.7	frames\image23	-0.38	0	0	89	0
24	24	54:46.7	frames\image24	-0.[illegible]	0	0	89	0
25	25	54:46.8	frames\image25	-0.36	0	0	89	0
26	26	54:46.8	frames\image26	-0.36	0	0	89	0
27	27	54:46.8	frames\image2[illegible]	-0.36	0	0	89	0
28	28	54:46.9	frames\image28	-0.36	0	0	89	0
29	29	54:46.9	frames\image29	-0.36	0	0	89	0
30	30	54:46.[illegible]	frames\image30	-0.36	0	0	89	0
31	31	54:46.9	frames\image31	-0.37	0	0	89	0
32	32	54:47.0	frames\image32	-0.38	0	0	89	0
33	33	54:47.0	frames\image33	-0.38	0	0	89	0
34	34	54:47.1	frames\image34	-0.36	0	0	89	0

Fig. 8. CSV file of the recorded dataset on-road.

Table 1
The details of the CNN Architecture.

Layer (type)	Output shape	Number of Parameters
conv2d (Conv2D)	(None, 31, 98, 24)	1824
conv2d_1 (Conv2D)	(None, 14, 47, 36)	21,636
conv2d_2 (Conv2D)	(None, 5, 22, 48)	43,248
conv2d_3 (Conv2D)	(None, 3, 20, 64)	27,712
conv2d_4 (Conv2D)	(None, 1, 18, 64)	36,928
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 100)	115,300
dense_1 (Dense)	(None, 50)	5050
dense_2 (Dense)	(None, 10)	510
dense_3 (Dense)	(None, 1)	11
Total params: 252,219		
Trainable params: 252,219		
Non-trainable params: 0		

- Multiprocessing is used which allows each sensor to work separately without any throttling issues. This resulted in a valid and accurate data set.
- The recording process was made much easier due to the program's pause, resume, quit, and save features.
- Data was stored in a CSV file.

Fig. 7 shows some samples of the recorded frames in the dataset.

A sample from the recorded CSV file is shown in Fig. 8.

The dataset, including the images frames and CSV file, can be obtained from: https://drive.google.com/drive/folders/1ckcWEJOOKJuyKMzdI0VEvbGdZ2Nft7uk?usp=share_link.

2.2. Network architecture

The architecture of CNN is developed from Ref. [27]. To reduce the costs and time involved in the training, the design took account of minimizing the trained parameters. Regulators and Batch-normalization are performed to significantly speed up the learning [28]. Detailed information regarding the CNN architecture is provided in Table 1.

A normalization layer, five convolutional layers, and three fully connected layers construct the network's nine layers. The network receives the input image, after the separation process into YUV planes. Image normalization is performed by the network's first layer. The normalizer is pre-programmed and cannot be changed throughout the learning process. The normalization technique can be changed to fit the network architecture and is expedited using GPU processing. Convolutional layers were determined empirically after a series of tests with varied layer configurations [27] to perform feature extraction.

Stride convolutions with a 22 stride and a 55 kernel were used in the first three convolutional layers, while non-stride convolutions with a 33-kernel size were employed in the last two convolutional layers. Three completely connected layers follow the five convolutional layers, leading to an output control value, which is the projected steering angle value. Exponential Linear Unit (ELU) is employed as an activation function for all layers.

2.3. Controlling the steering system

One of the main objectives of this proposed work is to control the steering system. Steering control requires two major components.

- The motor that can rotate the steering.
- The feedback that can be used to correct the steering angle.

Since the LE class does not have an angle sensor implemented within the car system. After adding the MPU 6050 as angle sensor to the car, we were able to develop a steering control system (SCS) using the aforementioned components. We developed a cost-effective method to control the EPS motor. This is done by utilizing a sensor as part of EPS system and overwriting them. This is performed without any physical connection between the sensors and the controlling system. In this manner, we achieved some advantage that makes SCS more reliable and requiring no modification on the EPS system from hardware to software.

The proposed method uses a sensor called Torque sensor, this sensor measures the torque that applied from the driver on the steering wheel

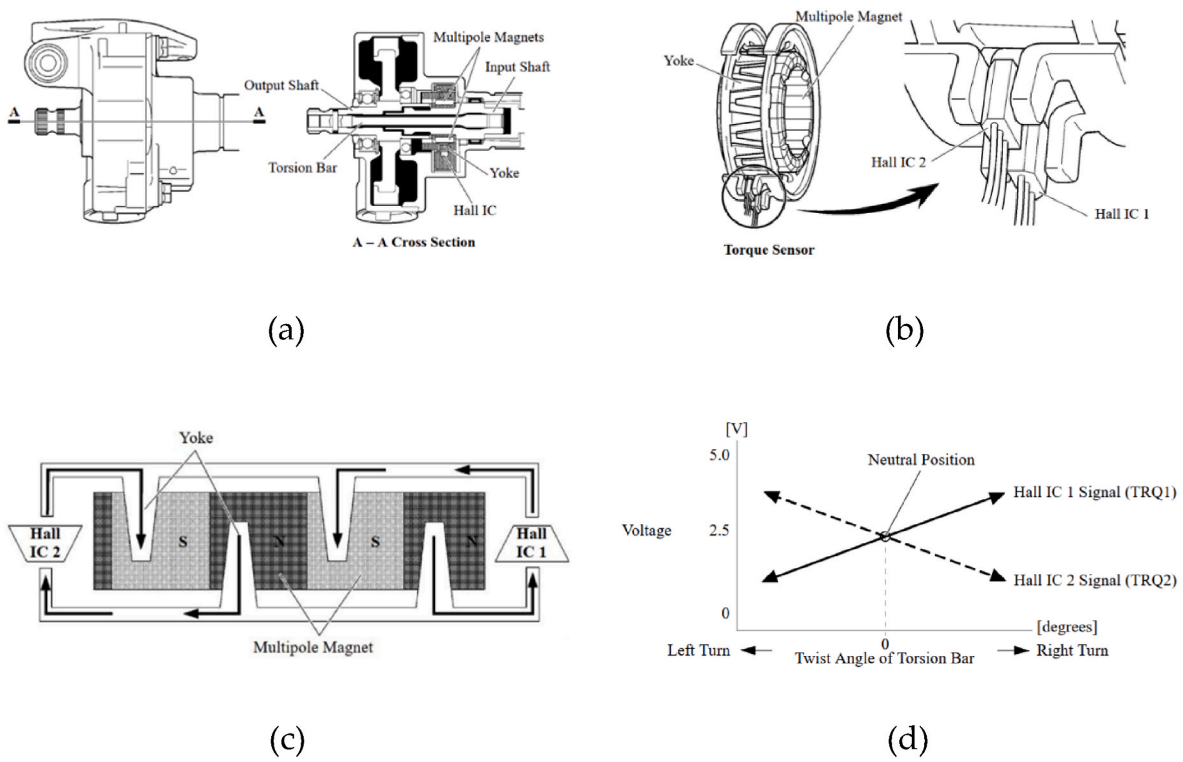


Fig. 9. (a) Torque sensor of EPS system, (b) Torque sensor of EPS another view, (c) Torque sensor (magnetic fields when route to left side), and (d) Torque sensor reading reference plot.

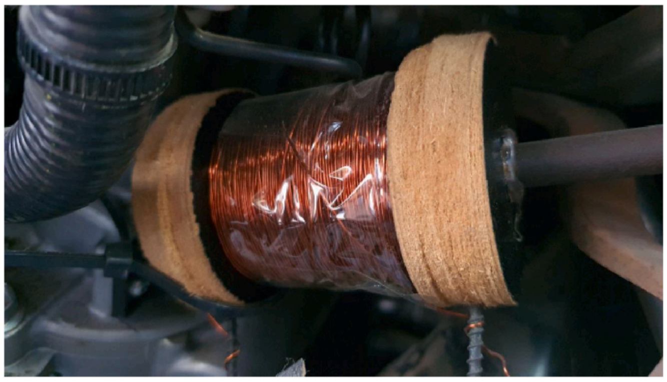


Fig. 10. The magnetic coil used to control the steering.

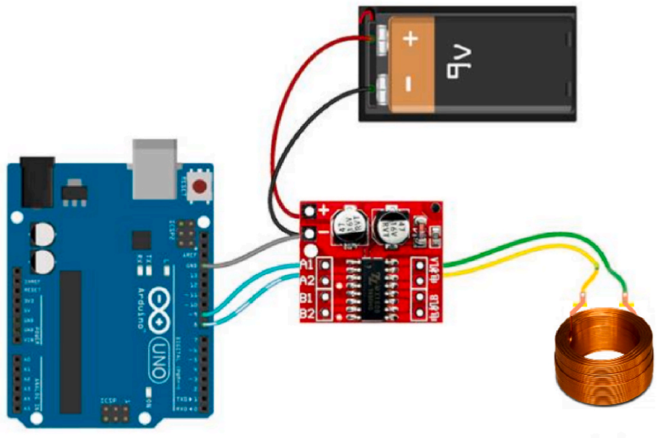


Fig. 11. H-Bridge wiring diagram to controlling an DC motor.

and send it the EPS ECU to possess it and make an output to the DC motor to rotate and help the driver.

The torque sensor is built into the steering column. A multipole magnet is mounted to the input shaft, and a yoke is mounted to the output shaft. The input and output shafts are joined by the torsion bar. A magnetic convergence ring assembly is placed outside of the yoke. The magnetic convergence ring assembly contains two Hall ICs, which face opposite to each other. The system detects the steering direction in accordance with the direction of the magnetic flux that passes between the Hall ICs. Furthermore, the system detects the steering torque in accordance with the amount of change in the magnetic flux density based on the relative displacement of the multipole magnet and the yoke. The EPS ECU monitors the torque sensor signals provided as outputs from the two Hall ICs to detect malfunctions.

When a driver turns the steering wheel to the right or left, the twist that is created in the torsion bar creates a relative displacement between the multipole magnet and yoke. At this time, the magnetic flux from the N to S pole of the multipole magnet passes between the Hall ICs. The system detects the steered direction of the steering wheel in accordance with the direction of the magnetic flux that passes between the Hall ICs. Hall IC1 and Hall IC2 are installed facing opposite to each other. As a result, the output characteristics of the two Hall ICs are constantly opposite each other. The system monitors the different outputs of these Hall ICs in order to detect malfunctions. The magnetic flux density becomes higher as it gets closer to the center of the respective pole. A Hall IC converts these magnetic flux fluctuations into voltage fluctuations, in order to transmit the turning torque of the steering wheel to the EPS ECU. Upon receiving the signals from the torque sensor, the EPS ECU calculates the required assist torque and outputs it to the motor (Refer to Fig. 9).

As it can be noted from Fig. 9, the Torque sensor is a hall sensor that is affected by magnetic fields. Depending on the intensity of the magnet field that applied near the sensor and depending on the polarity of the

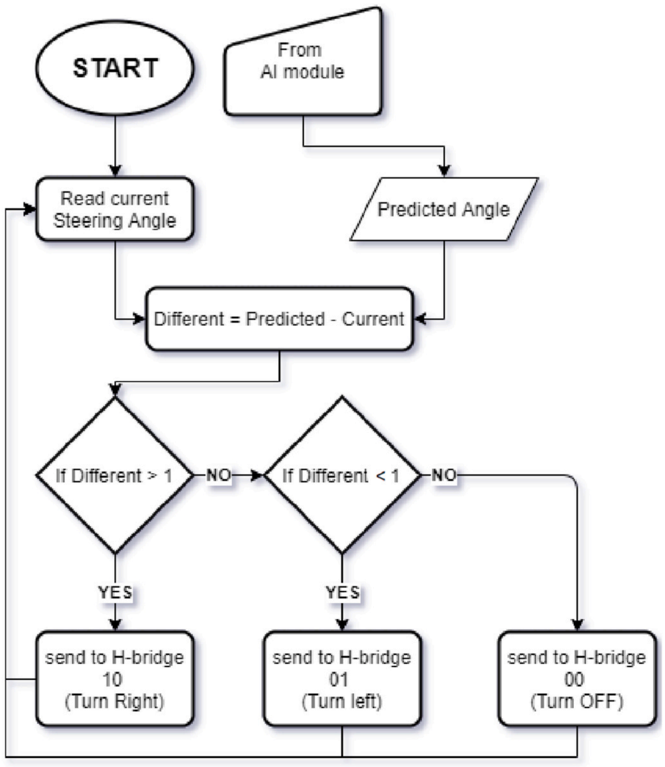


Fig. 12. The flow chart for the SCSP.

magnet or the direction of the magnet field, when a magnet is near this sensor, the parameter will affect the torque sensor and make it simulate the driver torque even if the driver does not apply any torque to the steering wheel. This will rotate the shaft to reduce the torque that is measured from the torque sensor. Thus, to control the EPS motor, torque sensor needs to be controlled by magnet, and to perform this, the magnets intensity and polarity need to be controlled, meaning that an electromagnet is required.

Since we need medium power magnet with low power consumption and small size to fit underneath the steering system, we must make this coil ourselves. Experimentally, we found that a 4.9-gauge copper wire that rotates on the cylindrical core 900 times with a core dimension of 40 mm length and 8 m diameter, is efficient for the torque sensor as shown in Fig. 10. In the event that this coil works under 0.9 A with maximum voltage 32 V, the voltage required to make normal effect on the steering shaft is determined experimentally and it is equal to 13 V.

In electromagnet, to control the intensity of the magnetic field, we need to increase/decrease the voltage that supplied to the coil, and to control the polarity, we need to change the direction of the current that input to the coil. So, the controller itself cannot supply the coil with this current and voltage, the maximum current out from Arduino UNO board is 500 mA, and the maximum voltage is 5 V, for this reason, we need a circuit that work as buffer or relay between the controller and the coil. The H-bridge satisfies all the requirements we need, it is designed specially to control the DC motor, and it is capable to change the current and even change the voltage if its supplied pulse with modulation PWM signal. The H-bridge requires two signals to control the current direction state as shown in Fig. 11.

The steering controller in our work is divided into two parts: the first part responsible for thinking, and the second part response to the first part output. The first part can be represented by raspberry PI that supports python. While the second part was represented by Arduino UNO board. It is noteworthy that in our system and for experimental purposes, we replaced the raspberry PI with a laptop, in general, any

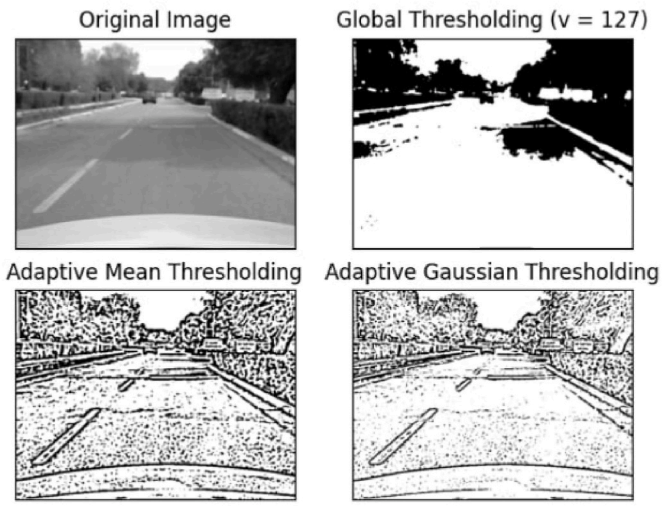

Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

Fig. 14. Applying a median filter and thresholds.

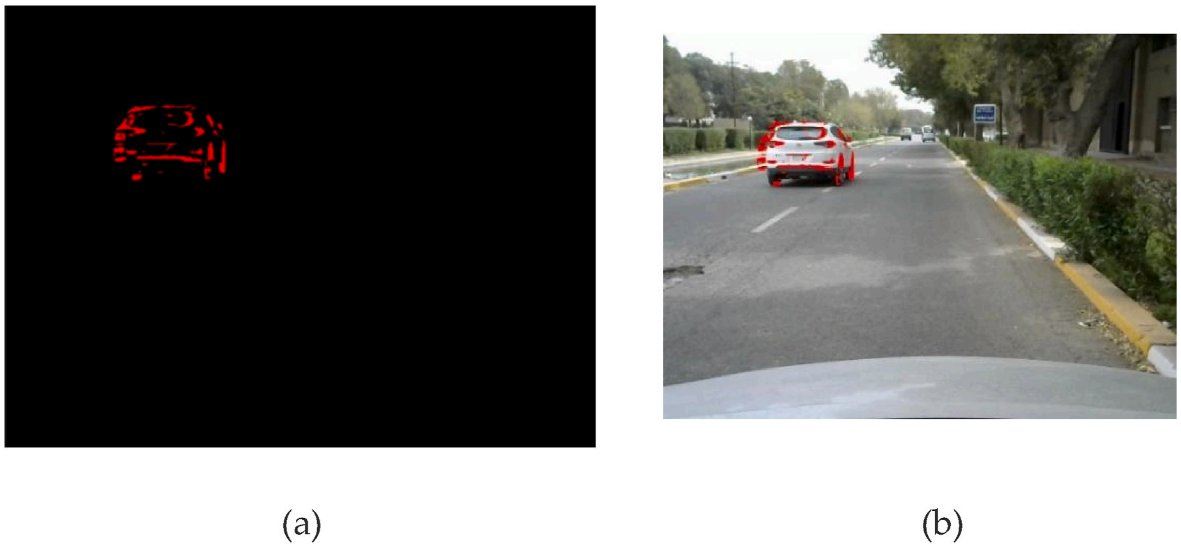
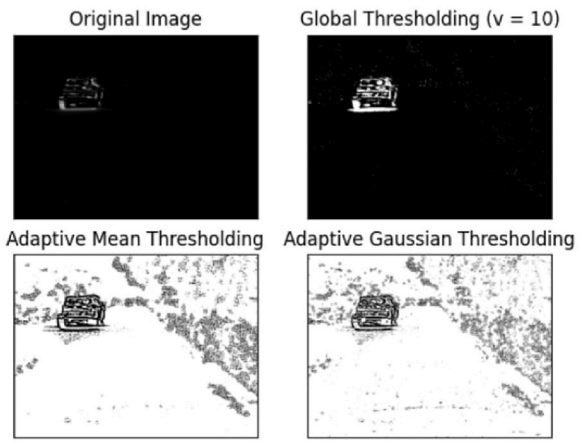
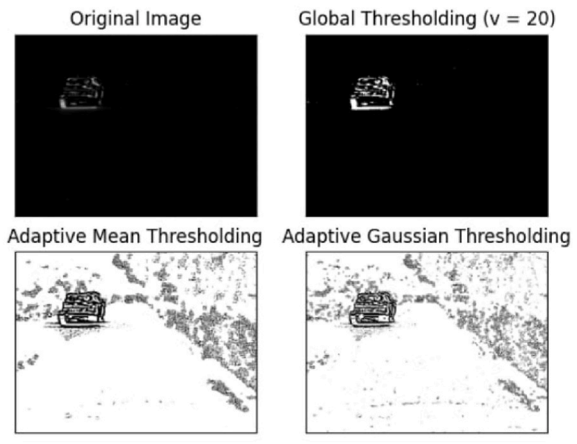


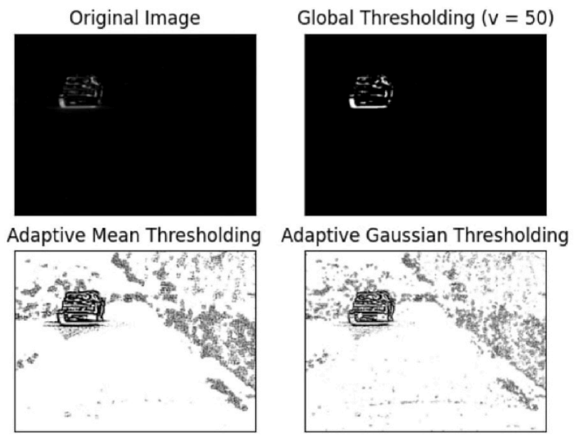
Fig. 13. (a) Difference between two consecutive frames, (b) Original image with difference between consecutive frames shown in (a).


Original Image
Global Thresholding (v = 10)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

(a)


Original Image
Global Thresholding (v = 20)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

(b)


Original Image
Global Thresholding (v = 50)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

(c)

Fig. 15. (a) A global threshold value of 10, (b) A global threshold value of 20, and (c) A global threshold value of 50.

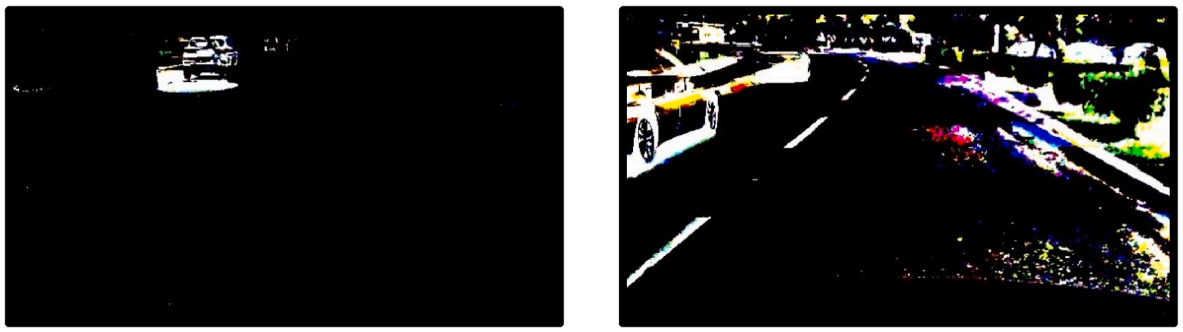


Fig. 16. Samples of frames after subtracting and applying the threshold.

platform run python program language can be considered.

To perform the communication between the two parts, Firmata is used which is a generic protocol for communicating with micro-controllers from software on a host computer. It is intended to work with any host computer software package [29]. Firmata can be uploaded to the UNO. On the other side, the host must have the Firmata package to communicate with UNO, since we use python, pyFirmata is the package utilized to communicate throw the Firmata protocol with the UNO. Firmata protocol can reduce the programing time and enhance the speed of the communication, even that, it eliminates the error that occurred in the controller due the programing issues, just uploading the standard code of Firmata in the controller without editing any line, and with simple setup in python side, all the ports of the controller can be uploaded. This method will give the following advantages.

- Our laptop does not have GPIO port like raspberry PI.
- All the controller ports are in the python environment, and we can call them as needed.

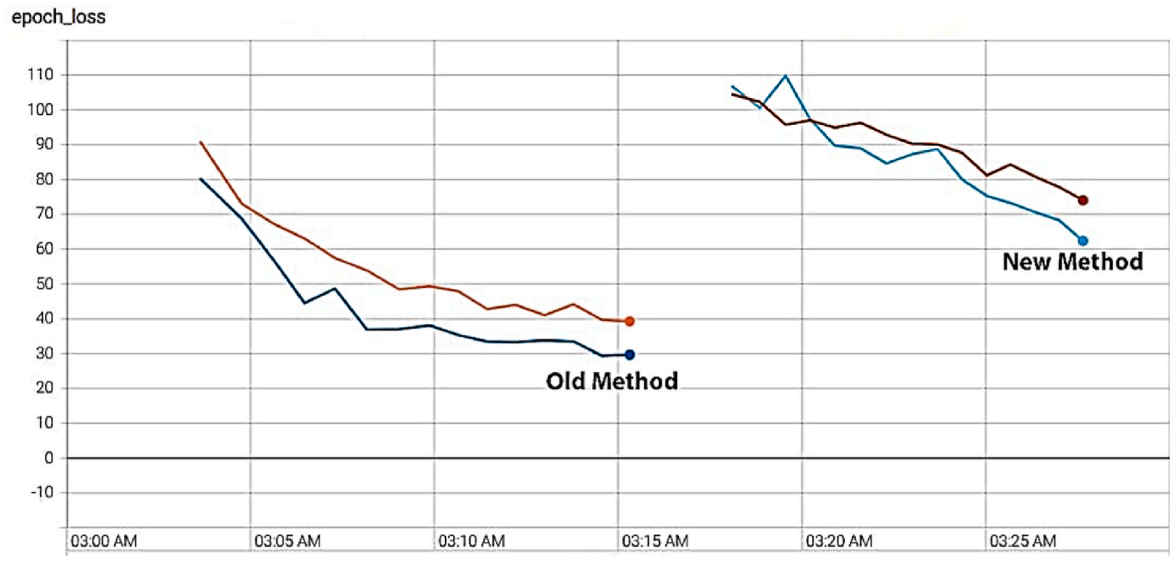


Fig. 17. The comparison between the old Method and the New Method based on the loss.

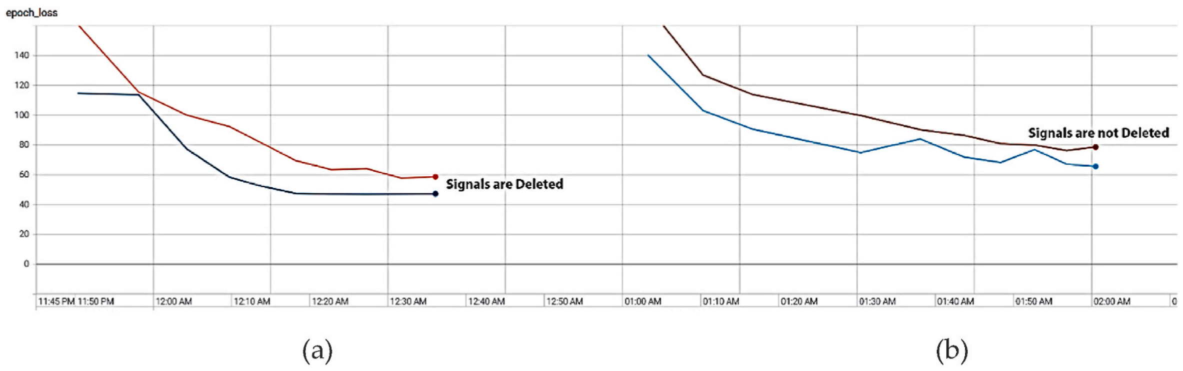


Fig. 18. The Loss results for the models (a) without changing lanes signal, and (b) with changing lanes signal.

Table 2
Three Models Comparison for three different datasets.

Comparison Metrics	M-10 K	M-170 K	M-400 K
Total Images	10,259	171,441	391,888
Recording Time	12 min	1 h & 45 min	4 h
Epochs	50	50	50
Loss	2.282	13.86	18.98
CSV File Size	0.95 MB	15.8 MB	33.8 MB
Frames Total Size	1.1 GB	16.7 GB	29.7 GB

Table 3
Comparison between Loss, Epochs and Training Time of the models.

Comparison Metrics	M-100E [a]	M-200E
Number of Epochs	100	200
Loss	15.4	10.3
Approximate Training Time	4 Hours	8 Hours

[a] M: Model, E: Epochs.

- The transmutation rate is very fast.

Once all parts of the SCS are ready to operate, starting from the angle sensor connecting to the laptop throw USB or wireless connection, the broadcast data of the current angle of the steering shaft will be used for the python program, the predicted angle from the CNN network will be passed to the steering control system procedure (SCSP) which compares among the current value and predict value of the angle sensor. Depending on this operation, one function of three will be called, those function responsible for the output state (turn left, turn right, no output), if the value is positive, turn right will be called, if its negative, turn left is called, if the value is 0 then the no-output will be called. This function will call the Firmata and will change the ports connected to the H-bridge to change the current directions as illustrated in Fig. 12.

3. Results

This section provides the results of the proposed SDV system. The first approach was to try to subtract images from each other and to train the model on the difference between the two images only. In Fig. 13 (a) the red shadow represents the difference between the two images while Fig. 13 (b) shows the difference only.

We also used a threshold to remove the small changes on the images. For example, a tree leaf that moves slightly should be ignored. Fig. 14 shows how the threshold affects the images by applying three types of thresholds. It is obvious the effect of the using Global threshold, Adaptive Mean threshold, and Adaptive Gaussian threshold compared with the original image.

Fig. 15 shows the results after applying the subtractions, with a global threshold of three different values, an Adaptive Mean Threshold, and an Adaptive Gaussian Threshold. Visually the best threshold is the Global threshold with a value of 20. In Fig. 16, the results of the new

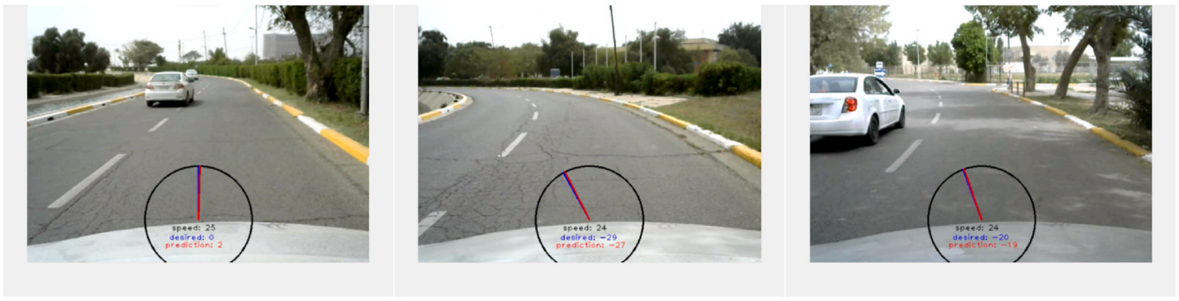


Fig. 19. Samples of the on-road testing.

frames after subtracting and applying the threshold are illustrated. We trained the CNN model on this new method and compared it to the old normal method. In Fig. 17, it can be shown that the old method got better results.

In Fig. 18, we have benchmark two models trained on the same dataset, however, the difference is one of them was trained on a dataset in which the images and data were removed when the car was changing lanes. We analyzed Fig. 18 and indicated that the CNN model got better results when the data for the changing lanes were removed.

Table 2 compares three models trained on three different datasets, namely, M − 10 K, M − 170 K, and M-400 K. The M − 10 K is trained with 10,000 frames, M − 170 K is trained with 170,000 frames, and M − 400 K is trained with ~400,000 frames.

After 100 Epochs the loss of the model reached 15.4 and after 200 Epochs the loss reached 10.3 as indicated in Table 3.

By analyzing Table 3, it can be shown that double the time is required to achieve a gain of 33.11%. Fig. 18 shows some samples of the trained model results on-road for different cases.

Using feedback from the steering shaft to represent the position and based on target position, the location of the steering shaft can be updated through rotating it in the desired direction as shown in Fig. 19. The results of AI model vary based on the data set that was used for the training. We trained the model on a track with circular paths, and the environment of the data set has a significant influence on the outcome. Our analysis of previous images showed that the road in the data set has two ways, and the presented model has learned to always take the left and slower path. This is reflected in the negative range of the steering angle, which indicates a leftward bias.

It is important to note that the data set we used to train the model is limited, as we only captured 8 h of data recording the driver's response on the same track. As a result, the model is expected to perform well on this track in 20% of driving situations, which is considered good performance. On other roads, the performance of the model is likely to be poor, with only 5% of driving situations handled effectively. Furthermore, the steering controller was built to handle the steering angle is basic and relies on an electromagnetic coil to make calculated noise on the hall sensor in the ESP system. The magnetic fields are used to control the movement of the steering column by inverting the polarity of the magnetic coil. This approach eliminates the need for wiring with the car's system, but its performance is weak and takes time to take effect. To enhance the overall performance of our system and increase the speed of the car while maintaining its safety, it would be beneficial to use an open car that allows direct access to the ECU. This would enable control of the EPS from the ECU or an external system built specifically for this purpose.

As a summary: the proposed method developed a model that serves as the brain of the self-driving car's assistant system is novel for two reasons. Firstly, it utilizes a convolutional neural network (CNN) to predict the steering angle. By training the CNN model with frames from a camera as inputs, it can accurately determine the steering angle and make driving easier and safer. Secondly, the proposed system is quite affordable, as it only requires the installation of a high-cost system. This is important, as 85% of manufactured vehicles do not include these supplementary features. In addition, the proposed method also involves the use of a simulated environment to test and train the model before deploying it in the real world. This is important, as the simulator can generate a variety of scenarios and conditions that the self-driving car may encounter in the real world. Furthermore, the Toyota Corolla 2009 vehicle with 4 h of raw vehicle sensors data was used to build the dataset, which is then used to train the model. This provides a more accurate and reliable model. Overall, the proposed method is novel in that it utilizes a combination of efficient and affordable technology to make driving easier and safer. The proposed system is a prototype that requires more testing for safety, and this represents an important step forward in the development of self-driving cars.

4. Conclusion

The development of a driving assistant system is a complex process that requires a combination of hardware, software, and artificial intelligence (AI) algorithms. However, converting the old car into self-driving cars is a challenge. This work presents a simple, and effective method for SDV, an old Toyota Corolla 2009 equipped with digital camera for capturing live video stream. Our proposed system can be used on any vehicle with the intension of modernizing old driving system cars with new technology to achieve reliable driverless system. We used object detection for the detection of objects using Deep Learning algorithms. In this respect, the outputs from the DL model are used to control the steering control system. A database for various road conditions was collected and will be publicly available for the SDV researchers. The results of the experiments showed that the CNN model can be successfully used to accurately predict the steering angle of a vehicle. The model achieved an average accuracy of good results on the on-road experiments. This demonstrates that the proposed model can be used to successfully develop a driving assistant system. In the future, we plan to 1) extend the model to other tasks such as object detection and lane detection, 2) collect more data from different vehicles to increase the accuracy of the model, and 3) use other AI algorithms such as recurrent neural networks (RNNs) and reinforcement learning to further improve the performance of the driving assistant system.

Funding

This research received no external funding.

CRediT author statement

Mohammad S. Mohammed: Methodology, Software, Writing- Original Draft **Ali M. Abduljabar:** Writing- Original Draft**,** Formal Analysis **Mustafa M. Faisal:** Software, Resources, Vizualization **Basheera M. Mahmmod:** Software, Data Curation **Sadiq H. Abdulhussain:** Conceptualization, Methodology, Supervision, Writing- Original Draft, Writing- Review & Editing **Wasiq Khan:** Conceptualization, Software **Panos Liatsis:** Methodology, Writing- Original Draft, Writing-

Review & Editing **Abir Hussain**: Methodology, Writing- Original Draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgments

The authors would also like to thank the University of Baghdad, Liverpool John Moores University, Khalifa University of Science and Technology, and University of Sharjah for their help and support.

References

[1] S. Karnouskos, Self-driving car acceptance and the role of ethics, IEEE Trans. Eng. Manag. 67 (2) (2018) 252–265.
[2] V. Anupama, A.G. Kiran, SIDA-GAN: a lightweight generative adversarial network for single image depth approximation, Res. Eng. 16 (2022) 100636.
[3] C. Urmson, And others, "Self-driving cars and the urban challenge, IEEE Intell. Syst. 23 (2) (2008) 66–68.
[4] P. Bansal, K.M. Kockelman, Forecasting Americans' long-term adoption of connected and autonomous vehicle technologies, Transport. Res. Part A Policy Pract. 95 (2017) 49–63.
[5] X. Zhou, Y. Fang, Y. Mu, Learning single-shot vehicle orientation estimation from large-scale street panoramas, Neurocomputing 367 (2019) 319–327.
[6] Z. Chen, Analysis and self-driving algorithm decision mode design, in: 2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA), 2022, pp. 93–97.
[7] J.M. Anderson, K. Nidhi, K.D. Stanley, P. Sorensen, C. Samaras, O.A. Oluwatola, Autonomous Vehicle Technology: A Guide for Policymakers, Rand Corporation, 2014.
[8] S. Singh, Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey, 2015.
[9] S. Molina, S. Ruiz, J. Gomez-Soriano, M. Olcina-Girona, Impact of hydrogen substitution for stable lean operation on spark ignition engines fueled by compressed natural gas, Res. Eng. 17 (2023) 100799.
[10] I. Ourya, S. Abderafi, Clean technology selection of hydrogen production on an industrial scale in Morocco, Res. Eng. 17 (2023) 100815.
[11] P. Undesa, World Urbanization Prospects: the 2018 Revision, vol. 26, 2018, p. 2018. Retrieved August.
[12] M.K. Rohil, Y. Ashok, Visualization of urban development 3D layout plans with augmented reality, Res. Eng. (2022) 100447.
[13] R. Ben Abdessalem, S. Nejati, L.C. Briand, T. Stifter, Testing vision-based control systems using learnable evolutionary algorithms, in: 2018 IEEE/ACM 40th International Conference on Software Engineering, ICSE, 2018, pp. 1016–1026.
[14] A. Arcuri, L. Briand, A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering, Softw. Test. Verif. Reliab. 24 (3) (2014) 219–250.
[15] J. Fritsch, T. Kuehnl, A. Geiger, A new performance measure and evaluation benchmark for road detection algorithms, in: 16th International IEEE Conference on Intelligent Transportation Systems, ITSC 2013, 2013, pp. 1693–1700.
[16] A. Khanum, C.-Y. Lee, C.-S. Yang, Deep-learning-based network for lane following in autonomous vehicles, Electronics 11 (19) (Sep. 2022) 3084.
[17] Z. Chen, X. Huang, End-to-end learning for lane keeping of self-driving cars, in: 2017 IEEE Intelligent Vehicles Symposium, vol. IV, 2017, pp. 1856–1860.
[18] M.K.A. Chy, A.K.M. Masum, K.A.M. Sayeed, M.Z. Uddin, Delicar, A smart deep learning based self driving product delivery car in perspective of Bangladesh, Sensors 22 (1) (Dec. 2021) 126.
[19] T.-H. Wu, T.-W. Wang, Y.-Q. Liu, Real-time vehicle and distance detection based on improved yolo v5 network, in: 2021 3rd World Symposium on Artificial Intelligence, WSAI, 2021, pp. 24–28.
[20] K. Cumali, E. Armagan, Steering control of a vehicle equipped with automated lane centering system, in: 2019 11th International Conference on Electrical and Electronics Engineering, ELECO, 2019, pp. 820–824.
[21] A. M P, G. R, M. Panda, Steering angle prediction for autonomous driving using federated learning: the impact of vehicle-to-everything communication, in: 2021 12th International Conference on Computing Communication and Networking Technologies, ICCCNT), 2021, pp. 1–7.
[22] A. Folkers, M. Rick, C. Buskens, Controlling an autonomous vehicle with deep reinforcement learning, in: 2019 IEEE Intelligent Vehicles Symposium, vol. IV, 2019, pp. 2025–2031.
[23] T. Baumeister, S.L. Brunton, J. Nathan Kutz, Deep learning and model predictive control for self-tuning mode-locked lasers, J. Opt. Soc. Am. B 35 (3) (Mar. 2018) 617.
[24] Ó. Pérez-Gil, et al., Deep reinforcement learning based control for Autonomous Vehicles in CARLA, Multimed. Tool. Appl. 81 (3) (Jan. 2022) 3553–3576.
[25] C. Rablau, LIDAR–A new (self-driving) vehicle for introducing optics to broader engineering and non-engineering audiences, in: Education and Training in Optics and Photonics, 2019, 11143_138.
[26] H. Yin, C. Berger, When to use what data set for your self-driving car algorithm: an overview of publicly available driving datasets, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems, ITSC, 2017, pp. 1–8.
[27] M. Bojarski, et al., End to End Learning for Self-Driving Cars, 2016 *arXiv Prepr. arXiv1604.07316*.
[28] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.
[29] C.C. Arduino, Arduino. Cc, 2005 l{\'\i}nea]. Available, https://www.arduino.cc/en/Main/ArduinoBoardNano. (Accessed 25 November 2016) *[Último acceso*.