2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)

Simulation of Autonomous Multifunctional Mobile Robot using Machine Vision

S Gowtham
Student, Department of Mechatronics Engineering
Hindustan Institute of Technology and Science
Chennai, India
gowtham.sridher5@gmail.com

R Praveen
Student, Department of Mechatronics Engineering
Hindustan Institute of Technology and Science
Chennai, India
praveen2000rajendran@gmail.com

P Sai Charan
Student, Department of Mechatronics Engineering
Hindustan Institute of Technology and Science
Chennai, India
saicharancherryroyal@gmail.com

M Parthiban
Student, Department of Mechatronics Engineering
Hindustan Institute of Technology and Science
Chennai, India
parthakash140899@ gmail.com

N Seenu
Assistant Professor,
Centre for Automation and Robotics
Hindustan Institute of Technology and Science
Chennai, India
nseenu@hindustanuniv.ac.in

RM Kuppan Chetty
Associate Professor,
Centre for Automation and Robotics
Hindustan Institute of Technology and Science
Chennai, India
kuppanc@hindustanuniv.ac.in

***Abstract*—In the recent decade, vision-based robotic systems are employed for numerous domestic and industrial applications. Currently, a majority of mobile robots lack multiple capabilities and are confined to perform specific tasks due to the drawbacks of conventional distance sensors. This paper demonstrates the integration of road sign recognition, leader-follower, object tracking and self-driving functionalities into a single multifunctional robot using machine vision. The proposed object tracking algorithm uses dimensions of the bounding box enclosing the object and location of the object's centre in the recognition frame to generate the necessary motion commands for the robot. A 2-D convolutional neural network is developed to predict precise steering angles associated with the captured images for end-to-end steering control in the simulation environment. The simulation results obtained in Webots and Udacity-self driving car simulator platforms prove the robot's adaptability for multiple applications**

Keywords—vision-based system; multifunctional robot; convolutional neural network; machine vision; object tracking; road sign recognition; leader-follower; self-driving; autonomous navigation;

I. Introduction

Vision-based mobile robots have evolved significantly in performance, operational flexibility, and decision-making capabilities due to artificial intelligence. Consecutively, imaging systems have vastly developed with the onset of machine vision by empowering cameras to autonomously recognize, classify, and track moving objects in videos and real-time environments. The capabilities of vision-based systems are further extended with the implementation of artificial neural networks. Convolutional neural networks are widely preferred for classification and prediction of continuously varying quantities preferably on large image datasets among various deep learning models. The CNN model lessens computational requirements by partially linking multi-layer perceptron and selectively performing feature extraction along with the height and width of input images fed into the model. A cost-effective approach is demonstrated in this paper to eliminate the need for a single robot to perform a specific task. Instead, a set of functionalities are integrated in a single robot and the user can select the operation to be performed by the robot. A potential aspect of the proposed multifunctional robot is that; its firmware can be programmed with communication cables to implement additional vision-based functionalities.

A. Related Work

A study has insight into autonomous navigation in environments that are incompatible with GPS. The proposed methodology in [1] involves the application of pattern recognition to detect objects to perform retrieval operations. A similar set of perceived objects were organized within the same class using clustering algorithms. A basic CNN architecture is developed and trained for classifying the perceived objects. Moreover, [2] on emphasizes event-based moving object tracking using dynamic vision sensors. The generated 3-D point clouds were processed using asynchronous cameras. Recently, an HMI of a vision-based human following robot in [3] comprises of a visual controller and a servo controller for modelling the robot's kinematics. Substantially, NVIDIA has proposed a CNN architecture in [4] to collect a training dataset by mapping raw pixels acquired from a single camera to the steering angles to test autonomous navigation in real-time environments. A novel approach of sensory fusion was presented by fusing the data acquired by various sensors present in the vehicle such as LIDAR, RADAR and camera to track pedestrians [5] and other neighbouring vehicles. An extension to the existing 2-D CNNs was adopted by parameterizing an additional depth variable by using a 3-D filter, giving rise to 3-D CNNs having recurrent and residual LSTM (Long Short Term Memory) layers [6]. Besides the single input steering control; another study has presented a reliable multi-modal end-to-end steering control [7] by integrating two distinct inputs, the steering angle values collected in the training dataset and the corresponding speed control from the feedback data of the previous recordings.

B. *Extension of the work*

Our proposed algorithm presents the working of a vision-based mobile robot capable of road sign recognition and moving object tracking using the bounding box generated by an RGB camera. The major contribution of this paper is the integration of four distinct vision-based functionalities such as road sign recognition, object tracking, leader-follower, and self-driving in a single robot. A feasible solution for autonomous navigation is demonstrated in the paper without the need for prior map knowledge and motion planning algorithms using road sign recognition. The scope of the robot is extended by developing a convolutional neural network to autonomously navigate the model in a known outdoor environment. The paper is outlined as follows: Section II and Section III comprise the methodology adopted in the respective simulation platforms. Section IV demonstrates the simulation results followed by the conclusion in Section V.

II. METHODOLOGY IMPLEMENTED IN THE MULTIFUNCTIONAL ROBOT

The Webots simulation platform provides a convenient user-interface for robot construction by navigating and adding components within several nodes in the scene tree. The IDE supports multiple programming languages and consists of built-in libraries thus, preferable for controller programming. Fig. 1 depicts the isometric view of the modelled four-wheeled robot. It consists of a rectangular chassis, a vision sensor, and a rotational motor for each wheel.

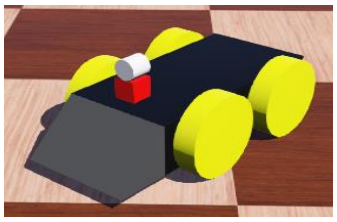


Fig. 1. Isometric view of the robot

Each motor receives individual command signals to impart motion based on the pixel variation in the bounding box generated by the vision sensor. The 1 DoF rotational motors are integrated with the robot via hind joint nodes within the robot's children class.

A. *Road Sign Recognition*

Conventional self-navigating robots comprise expensive sensory components and prove useful for autonomous motion planning and navigation in unknown environments. A feasible alternative is presented in this paper for autonomous navigation using road sign recognition. The approach involves the implementation of a camera node in the robot's children class to recognize four sets of commonly used road sign symbols to change the robot’s course in Webots. Such distinct symbols are labelled with unique tag IDs to change the robot's course until it reaches the destination. The robot undertakes sharp turns, curves and U-turns by rotating 90°, 45° and 180° about its axis respectively. A combination of a U-turn and a sharp turn is implemented by a single road sign to change the robot’s course by 135°. Correspondingly, each set of symbols have their respective counterparts to impart anti-clockwise rotation to the robot model.

B. *Object Tracking*

Object detection is an essential pre-requisite for any object tracking approach. In this section, we implement the marker recognition functionality to detect a spherical object moving on a 2-D plane. The camera library in Webots consists of a vast array of functions to customize image capturing for selectively improving the robot's motion while tracking an object. Our proposed methodology involves mounting an RGB camera node on the robot model to assign a bounding box to a spherical object for capturing the object's image in a 3D window referred to as the recognition frame. Fig. 2 shows various segments in the recognition frame generated by the vision sensor.

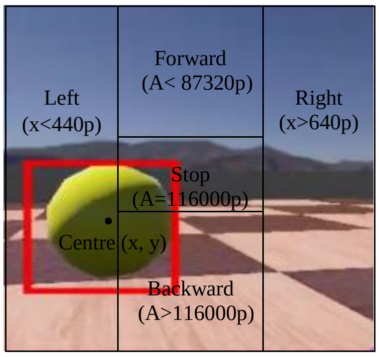


Fig. 2. Various segments in the recognition frame

The algorithm is initiated based on the location of the object's centre in any of the five segments of the frame. The height(y) and width(x) of the recognition frame are parameterized as 1080 pixels with a 180° field of camera view.

Object tracking

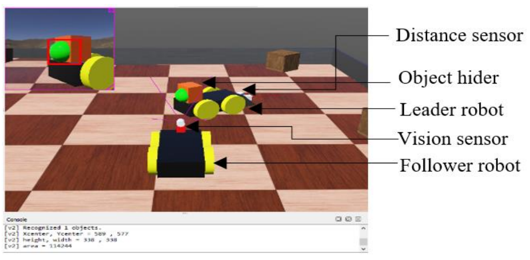
```
1: **Input:** Image of the object within a 1080*1080 recognition frame
2: **Output:** Motion of the robot
3: Assign the mid-point of recognition frame as 540 pixels
4: The enclosed area is a product of height(n) and width(m) of the
   bounding box
5: if x <= 440 then
6:    Initiate left turn
7: else if x >= 640 then
8:    Initiate right turn
9: else if a >= 116000 then
10:    Presence of nearby object is detected and backward motion is
       initiated
11: else if a <= 87320 then
12:    Presence of distant object is detected and forward motion is initiated
13: else
14:    Stop
```

Fig. 3. Pseudocode for object tracking

Correspondingly, the midpoint of X-axis in the recognition frame represents 540 pixels. A threshold value of 100 pixels is assigned away from the midpoint on both sides of the x-axis to prevent wobbling. The enclosed area of the object is calculated as the product of the bounding box's height(n) and width(m) to generate forward and backward motion concerning the object's vicinity to the camera. Fig. 3 shows the adopted decision-based approach for object tracking. The robot motion is influenced by the location of the object's centre in any of the five segments in the recognition frame.

C. Vision-based Leader-Follower Pair

This functionality can extend the robot’s application in mining and agricultural sectors to perform sequential tasks. The leader robot uses a conventional distance sensor to perform obstacle avoidance. This section presents an approach to eliminate a collision possibility of a leader-follower pair in relatively flat indoor environments while receiving visual input from the leader. The follower robot consists of a front-facing RGB camera mounted at the top to locate and follow the leader robot carrying out obstacle avoidance in a 2-D plane. A distance sensor is mounted on the leader to serve the purpose of falling within the range of the vision sensor mounted on the follower robot while collectively avoiding the obstacles in its path placed as wooden boxes.



```
[v2] Recognized 1 objects.
[v2] Xcenter, Ycenter = 589 , 577
[v2] height, width = 338 , 338
[v2] area = 114244
```

Fig. 4. The leader-follower pair in Webots

Fig. 4 depicts a leader-follower pair in the Webots environment. An object hider is an additional shape node mounted on the leader robot to overlap a portion of the 3-D object for preventing visibility during turns. This additional component mitigates the risk of a head-on collision of both robots during sharp turns. In this scenario, the object hider effectively covers a portion of the spherical object within the follower’s field of view. The incorporated algorithm is an extension of the object tracking algorithm discussed in the previous section. Upon considering the leader and follower’s clearance distance, bounding box area, and pixel variation observed within the frame, the follower robot follows the leader with a clearance gap of 30 centimeters.

III. AUTONOMOUS NAVIGATION IN UDACITY SELF-DRIVING CAR SIMULATOR

The road sign recognition functionality of the robot has shown better performance for autonomous indoor navigation although, it is not practically applicable in relatively large environments with varying weather and lighting conditions. Due to this drawback we emphasize the implementation of a self-driving functionality to broaden the applications of the robot in vast outdoor environments. A neural network model is developed to undergo supervised learning using a vast dataset consisting of the environment’s images and vehicle parameters to perform end-to-end steering control autonomously [7]. The Udacity self-driving car simulator is preferred over Webots as it meets the above requirements.

A. Collecting the Dataset

The data-acquisition vehicle model in the simulator consists of three on-board cameras to capture time-stamped images of the track from different perspectives. A dataset comprising of the captured front camera images and its corresponding steering values was obtained after covering four laps in the simulation environment by means of manual control. The labelled dataset consists of front camera images and corresponding y-labels that include steering angle, throttle, speed and braking. The steering commands in [7] are represented as 1/r where ‘r’ denotes the turning radius. This way, the singularity condition that arises due to infinite turning radius during straight path traversal is effectively nullified to 0. The turning angles associated with all possible turns in the environment are scaled to range between -1 and 1. A smooth transition of steering angles occur from the left turns, i.e. negative values to right turns being positive values. A total of 13,074 captured images of the track were imported into the workspace to perform data analysis and image processing.

B. Visualizing the Steering Angle Distribution

Consistent distribution of the collected steering angle data is necessary for stabilizing the model to avoid contact with the edges of the lanes. Since the model is observed to move along a straight path for longer intervals, it is crucial for maximizing the weightage of neutral angle values for optimal navigation. About 9,569 redundant images of the model moving in a straight path are discarded, and the dataset is left with 3,505 images. Data redundancy can affect the overall training process by increasing computational load on subsequent layers in the network and lengthening the time taken to train the CNN model.

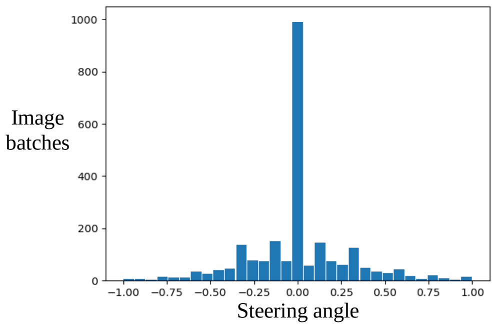


Fig. 5. The overall distribution of steering angle values

The steering angle of the data-acquisition robot model is a continuous quantity as it can take up infinite possible decimal values in the range of -1 to 1. A CNN model is developed to map the image data to the corresponding y-label (steering angle) to establish relationship for the regression problem. Element-wise addition of the steering values was implemented in subsequent matrices of steering values. The resultant matrix is multiplied with a factor of 0.5 to compensate for the doubling of values. This approach effectively scales the least decimal values to zeroes for effectively increasing the null angle distribution to orient the model along the center of the road. Fig. 5 shows the histogram bar graph plotted to visualize the steering angle distribution.

C. *Image Augmentation*

In this stage, the images are augmented to artificially expand the distribution of front camera images in the training dataset using OpenCV image processing library. The acquired RGB images are converted into YUV colour space for improved track visualization. This process aims to closely mimic human visual perception so that the model can interpret the road during real-time scenarios such as varying lighting conditions and road lanes.

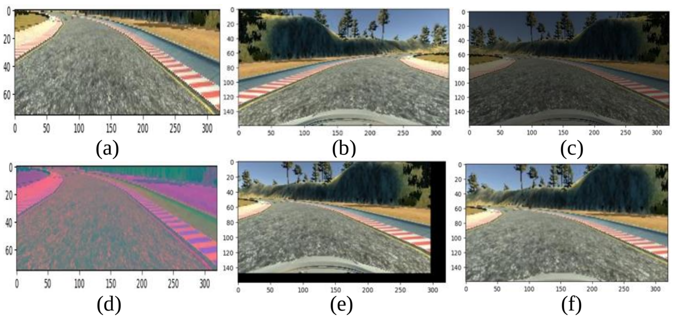


Fig. 6. Image pre-processing and augmentation

Initially, the images are cropped to abstract the track and lane information from redundant scenery features. Fig. 6(a & b) depicts the result of cropping and flipping (mirroring) techniques, respectively. Fig. 6€ depicts an image darkened by a factor of 0.3 to circumvent varying lighting conditions in the environment. Fig. 6(d) shows the image in YUV color space for better lane-edge interpretation. Fig. 6€ represents panning, where, the x-y dimensions of the image are translated by 10% toward the top-left corner. Fig. 6(f) shows the zoomed image with a 1.4 zooming scale factor. This way, the augmentation is carried out to create additional images for expanding the steering angle distribution to achieve faster and precise motion responses during autonomous navigation. The images are split in an 80:20 fashion for training and validation, respectively.

D. *Developing the CNN model*

Among various deep learning neural networks, CNN (Convolutional Neural Network) is preferable for steering angle prediction from a vast dataset of 2-D images. CNNs have widely been implemented for incorporating self-driving capabilities that especially involve more massive datasets. Batch normalization is applied to reduce the memory requirements and time taken to train the overall model effectively. Keras API consists of a wide array of deep learning functions that have been selectively implemented for constructing the overall CNN model. In this model, batch normalization has been applied to the input layers to preserve stability and reduce the computational load in consecutive layers. Fig. 6 illustrates the methodology adopted for steering angle prediction. At the initial stage of the training process, the captured images are led into the input layers that undergo batch normalization before getting passed into the convolutional layers. The model utilizes a 5x5 kernel filter in the initial convolution layers followed with a 3x3 kernel filter in the final convolution layer for maximizing feature extraction [5].

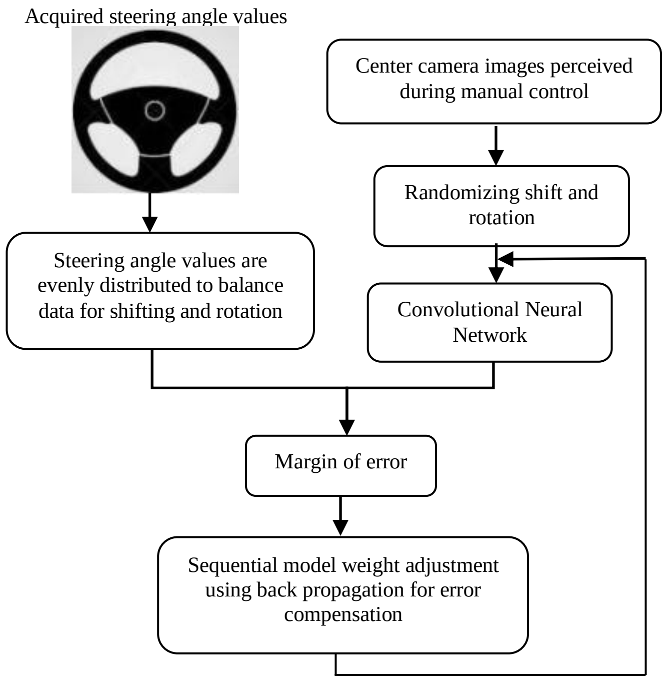

Acquired steering angle values
Center camera images perceived during manual control
Randomizing shift and rotation
Steering angle values are evenly distributed to balance data for shifting and rotation
Convolutional Neural Network
Margin of error
Sequential model weight adjustment using back propagation for error compensation

Fig. 7. The methodology incorporated for steering angle prediction

The weighted summation of multi-perceptron model is led into an activation function to toggle the perceptron's state, thereby effectively feeding the transformed image output to the subsequent convolution layers. The ReLU (Rectified Linear Units) activation function is commonly employed for rectifying gradients within the hidden neural network [7]. The function outputs positive values directly, whereas the negative values produce a sharp curve toward 0. Since the dataset comprises negative steering values, we have incorporated a reliable alternative to activate the multi-layer perceptron using ELU (Exponential Linear Unit) activation function [9]. The positive values are retained directly, whereas negative values are led into an exponential function to smoothen the curve. Consecutively, the data passed from the final convolutional layer is flattened and converted into a 1-D array to be passed into three fully connected layers to combine the convolved data to establish autonomous steering control.

IV. SIMULATION RESULTS

A. *Implementing Road Sign Recognition for autonomous navigation*

The standardized road signs utilized for navigation are imported into Webots and mounted according to the robot's field of view. As the robot comes across the familiarized road sign, it will automatically change its course by rotating at an angle pre-defined for each road sign. In Fig. 8, the first row illustrates left and right sharp turns associated with a rotation angle of 90°. In Fig. 8, the second row depicts right and left curves which directs the robot 45° toward either directions.

Consecutively, U-turn signs impart a 180° rotation to the robot. In Fig. 8, the fourth row depicts the combination of a sharp turn and U-turn to initiate a single motion command for a rotation of 135° about its axis. A conditional structure is designed to accommodate all eight events that can occur while formulating a path for the robot. Each road sign is assigned with a unique tag ID that corresponds to a rotation about its axis, after which the robot must traverse the path.

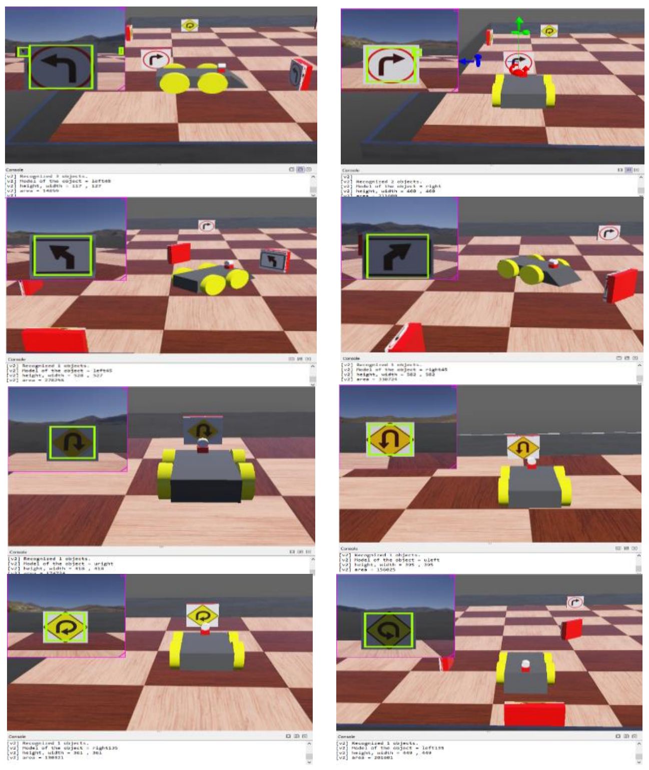


Fig. 8. Autonomous marker-based navigation in Webots

B. *Object tracking using an RGB Camera*

The object's center coordinates in a segment of the recognition frame initializes the condition-based structure to synchronize the robot motion relative to the object. Once the imported nodes have been configured, the camera node is mounted on the robot model by adding the hingejoint node to the tree.

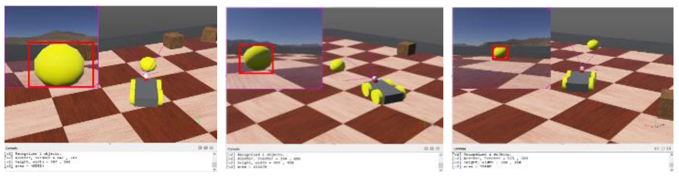


Fig. 9. Moving object tracking with RGB camera in Webots

The simulation is carried out using a set of functions from the 27 in-built functions in the Webots camera library. The camera's horizontal field of view is constrained to pi radians considering the absence of a spherical field. The camera's point of view can be calculated using the pixel parameters of the image's height. Anti-aliasing is performed to smoothen the edges of the object for distant recognition and efficient tracking. The robot turns toward the left to compensate a lower pixel value in the x-axis. Likewise, the model shifts toward the right as the pixel values are higher than the mid-point value, i.e. 540 pixels. Fig. 9 shows gradually lesser bounding box dimensions as the spherical object moves farther away from the robot. The bounding box area value of 116,000 pixels indicates the object's closeness to the RGB camera, which signals the model to move backwards. Correspondingly, the bounding box area value of 87,320 pixels is acquired while the object is distant. Thus, the command is sent to move the mobile robot model forward by reversing motor rotation direction.

C. *Training and Validating the CNN Model*

The neural network model's hidden layers consisting of five 2-D convolutional layers are implemented for feature extraction by sliding the kernel filters along the height and width of input images [10]. The input image dimensions are parameterized initially are convolved using 5x5 kernel filters in the first three layers to extract minor information such as track curvature and lane edges. The transformed images obtained from corresponding multi-perceptron layers are further extracted with 3x3 kernel filters to attain refined details. The overall convolved images are flattened to get converted into a 1-D array before being led into three fully connected layers. In the output layer, the predicted steering angle is obtained from a single perceptron.

The neural network model is compiled with the Adam optimizer, and the loss function is applied. The optimizer accounts to the learning rate at which the weights get updated within the overall network. Currently, Adam is most preferred among other optimization algorithms due to its straightforward implementation and fewer memory requirements, and it significantly surpasses the classical stochastic gradient descent method [7]. The learning rate in Adam optimizer is set to 0.0001 and initialized using mean-square-error loss function. This optimization strategy adapts with the learning rates of first and second gradient moments to effectively minimize the training and validation losses. Epochs represent the number of forward and backward passes of the entire dataset through the learning algorithm to update the weights and reduce the mean-square-error [8]. The dataset is grouped with a batch size of 100 images to undergo normalization and to accelerate the learning process.

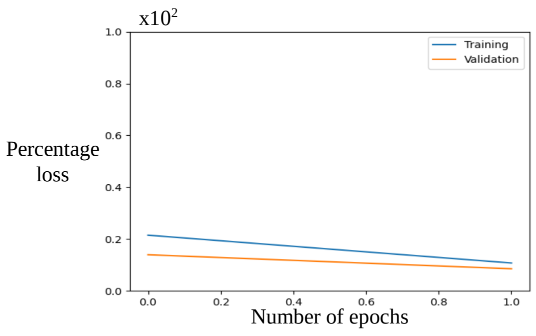


Fig. 10. Training and validation losses after one epoch

Fig. 10 shows that the losses have significantly decreased to less than 30% after one epoch. This way, the model is continued to be trained for ten epochs with the same learning rate. Fig. 11 represents a gradual decrease in both losses to less than 10% by passing the dataset through the learning process for ten epochs. The model is inferred to be stable as significantly less differences in training and validation losses were observed throughout the training period. It can be inferred from both graphs that the model does not undergo overfitting as the training loss is still marginally greater than the validation loss.

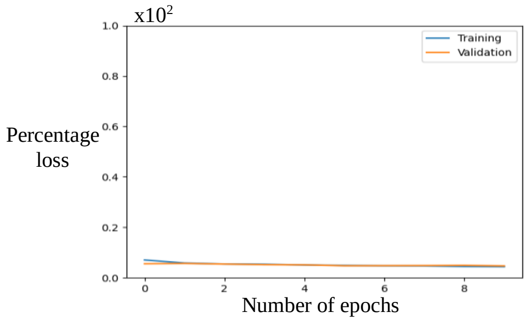


Fig. 11. Training and validation losses after ten epochs

The vehicle's speed and throttle parameters are pre-defined before launching the model in the track environment. The CNN model will continuously compare the images in the dataset and its associated steering data with the instantaneously perceived images during self-driving.

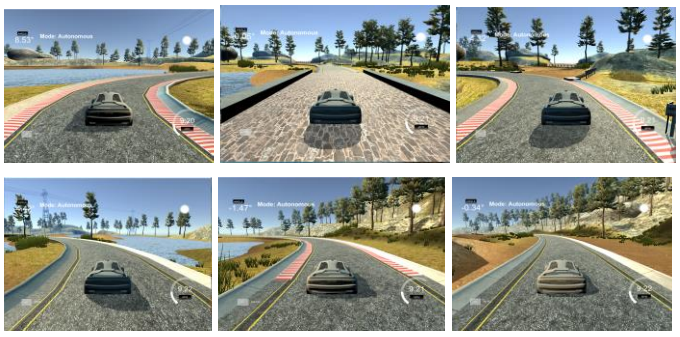


Fig. 12. Self-driving in various locations of the simulation environment

Fig. 12 depicts self-driving in various lighting conditions and several locations throughout the simulation environment. The road lanes are non-uniform throughout the map, and there are several instances where no lanes exist, as shown in the bottom row. The model did not get in contact with the edges of the lanes in all situations, and no significant wobbling was present. Fig. 12 illustrates the output of autonomous navigation using lane-edge detection from the journalized data. The model provides convincing results by orienting the vehicle along the center of the road and, at brief intervals, marginally align itself closer to the lane edges while encountering curved roads.

V. CONCLUSION

Due to the growing need for autonomous robots in various fields, artificial intelligence is steadily evolving as a crucial requirement for mobile robots. Simulations were carried out in Webots and Udacity-self driving car simulator to emphasize the multi-functionality of the robot. A feasible low-cost solution for autonomous navigation using road sign recognition was proposed. The adopted object tracking algorithm can efficiently detect and track a spherical object moving on a 2-D plane. Each functionality namely object tracking, leader-follower, self-driving and road sign recognition have visual perception in common and thus, can be integrated into a single vision-based robot.

The CNN model was successful to autonomously navigate the robot in the simulation environment by end-to-end steering control using lane-edge detection with a single sparse training parameter (steering angle). The model can be trained well beyond ten epochs to achieve much lower training and validation losses, which in turn will improve the steering angle prediction. In the future, a low-cost multifunctional robot will be fabricated and experimented with a Raspberry Pi camera module. An additional keypad will be integrated with the Raspberry Pi to allow the user to select the desired functionality to be performed by the robot.

VI. REFERENCES

[1] B. A. Erol, A. Majumdar, J. Lwowski, P. Benavidez, P. Rad, and M. Jamshidi, "Improved Deep Neural Network Object Tracking System for Applications in Home Robotics," Studies in Computational Intelligence, pp. 369–395, 2018.

[2] A. Mitrokhin, C. Fermüller, C. Parameshwara and Y. Aloimonos, "Event-Based Moving Object Detection and Tracking," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, 2018, pp. 1-9, doi: 10.1109/IROS.2018.8593805.

[3] M. Gupta, S. Kumar, L. Behera and V. K. Subramanian, "A Novel Vision-Based Tracking Algorithm for a Human-Following Mobile Robot," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 7, pp. 1415-1427, July 2017, doi: 10.1109/TSMC.2016.2616343.

[4] Bojarski, Mariusz, et al. "End to end learning for self-driving cars." arXiv preprint arXiv:1604.07316 (2016).

[5] A. Buyval, A. Gabdullin, R. Mustafin and I. Shimchik, "Realtime Vehicle and Pedestrian Tracking for Didi Udacity Self-Driving Car Challenge," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 2064-2069, doi: 10.1109/ICRA.2018.8460913.

[6] Du, Shuyang, Haoli Guo, and Andrew Simpson. "Self-driving car steering angle prediction based on image recognition." arXiv preprint arXiv:1912.05440, 2019.

[7] Z. Yang, Y. Zhang, J. Yu, J. Cai and J. Luo, "End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perceptions," 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, 2018, pp. 2289-2294, doi: 10.1109/ICPR.2018.8546189.

[8] M. Duong, T. Do and M. Le, "Navigating Self-Driving Vehicles Using Convolutional Neural Network," 2018 4th International Conference on Green Technology and Sustainable Development (GTSD), Ho Chi Minh City, 2018, pp. 607-610, doi: 10.1109/GTSD.2018.8595533.

[9] M. V. Smolyakov, A. I. Frolov, V. N. Volkov and I. V. Stelmashchuk, "Self-Driving Car Steering Angle Prediction Based On Deep Neural Network An Example Of CarND Udacity Simulator," 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), Almaty, Kazakhstan, 2018, pp. 1-5, doi: 10.1109/ICAICT.2018.8747006.

[10] A. Agnihotri, P. Saraf and K. R. Bapnad, "A Convolutional Neural Network Approach Towards Self-Driving Cars," 2019 IEEE 16th India Council International Conference (INDICON), Rajkot, India, 2019, pp. 1-4, doi: 10.1109/INDICON47234.2019.9030307.