INTELLIGENT AUTOMOBILES FOR INDUSTRIES

Lokesh M
Asst.Professor EEEDept
NIE Institute Of Technology
Mysuru

Akshata G Kademane
Elecrical and Electronics dept
NIE Institute Of Technology
Mysuru

B R Shreyas
Electrical and Electronics dept.
NIE Institute Of Technology
Mysuru

Ananya M Gowda
Elecrical and Electronics dept
NIE Institute Of Technology
Mysuru

Apoorva S
Elecrical and Electronics dept
NIE Institute Of Technology
Mysuru

***Abstract*—**AGVs(Automated Guided Vehicles) are autonomousvehicles or driverless vehicles used to transfer materials efficiently in a facility or an industry. This project aims to improve the AGVs into intelligent automobiles. The autonomous driving feature is accomplished with the help of behavioural cloning which tries to mimic the driving of the user which was used to train the vehicle. The map of the environment is generated using monocular visual odometry during the training and also during autonomous driving.The alternative path is selected using the map generated.The processing related to the autonomous driving is done on the controller and the processing related to the image processing is done on the server. The vehicle proposed here aims to reduce the overall cost of an intelligent vehicle used in an industry environment and use the resources as efficiently as possible.

Keywords—cloning; machine learning; reward; reinforcement; action

I. INTRODUCTION

Industries are becoming more and more efficient with the help of machines. These machines include conveyor belts, automatic guided vehicles, etc. Although these are making industries faster humans are still needed in the transitional periods. So, the industries cannot run all day long (i.e., not for 24 hours) and also not to mention the breaks taken by the workers. The highest level of automation achieved with the help of automatic guided vehicles (AGV). AGVs are preprogramed to take the objects to a particular destination and follow a path fed by the user. AGVs can transport only one type of object to any destination. This project aims to improve the AGVs into intelligent automobiles. These vehicles will be able recognize an object and transport to its corresponding destination for further processing or shipping. The vehicle will be able to detect the lane or path. The vehicle will also be able to avoid obstacles and take necessary actions. These tasks are performed with the help of image processing techniques and machine learning algorithms implemented in OpenCV.

II. LITERATURE SURVEY

DeepPicar is a low-cost Deep Neural Network based Autonomous Car [1] which is a small-scale replication of a real self-driving car called DAVE-2by NVIDIA. DAVE-2 usesa deep convolutional neural network (CNN), which takes images from a front-facing camera as input and produces car steering angles as output. Using Deep Picar, we analyse the Pi 3's computing capabilities to support end-to-end deep learning based real-time of autonomous vehicles. Deep Pi cars neural network architecture is identical to that of NVIDIA's real self-driving car DAVE-2. The vehicle and lane detection algorithm presented [2] is successful constructed and unstructured roads with a variety of features including curbs, shoulders, median dividers, and solid or dashed lane markings. The vehicle is capable of autonomously traversing long stretches of straight road in a variety of conditions with the same set of algorithmdesign parameters. Better performance is hampered by slowly updating inputs to the steering control system. In this paper lane detection is done using image processing, featureextraction, temporal integration and inverse perspective algorithms are used. The majority of lane detection research does not supply information about, which is essential for real-time control of autonomous vehicles. Miniature Autonomous Vehicle Development on Raspberry Pi [3] presents a novelmethodology of designing a miniature self-driving vehicle. Using a simulated GPS, we were able to position the car on the racetrack, and the navigation was done by a lane following method that implied lane markings detection and tracking. Two types of traffic signs were detected: stop and parking. Whenever the stop sign was found, the car stopped for a short period of time. If a parking traffic sign was encountered, the car performed the lateral parking procedure.Autonomous navigation of an automated guided vehicle in industrial environments [4] describes the navigation system of a flexible AGV intended for operation in partially structured warehouses and with frequent changes in the floor plant layout. Thisis achieved by incorporating a higher degree of on-board autonomy. In order to develop this flexible system, we have developed series of modules organized by layers. We have adopted the use of topological and grid-based maps for navigation; specifically, the use of a topological map for a high-level description of the environment and a grid map for a local description of the environment including currently sensed obstacles Topological maps are used for fast planning and high-level planner. This representation is automatically generated by the use of a rough description of the environment, which facilitate the adaptation when floor layout changes.

III. METHODOLOGY

The implementation of the intelligent automobile consists of two types of machine learning models which are as follows:

• Supervised Machine learning (Behavioural cloning)
• Reinforcement learning (Double Deep Q learning).

The process can be explained using the block diagram as shown in the Fig 1.The two types of algorithm with computer vision technique flow diagram. The input and output can be clearly seen. Input is images captured by camera and output is steering angle and throttle.

*A. Computer vision Algorithm:*This algorithm is developed using python OpenCV.It has following steps.

1. Detect and extract all edges using Canny Edge Detector.
2. Identify the straight lines through Hough Line transform.
3. Separate the straight lines into positive sloped and negative sloped (candidates for left and right lines of the track).
4. Reject all the straight lines that do not belong to the track utilizing slope information.

B. Behavioural cloning:Behavioural cloning is a method by which human subcognitive skills can be captured and reproduced in a computer program.As the human subject performs the skill,his or her actions are recorded along with the situation that gave rise to actions.A log of these records is used as input to a learning program.The learning program outputs a set of rules that produce skilled behaviour.

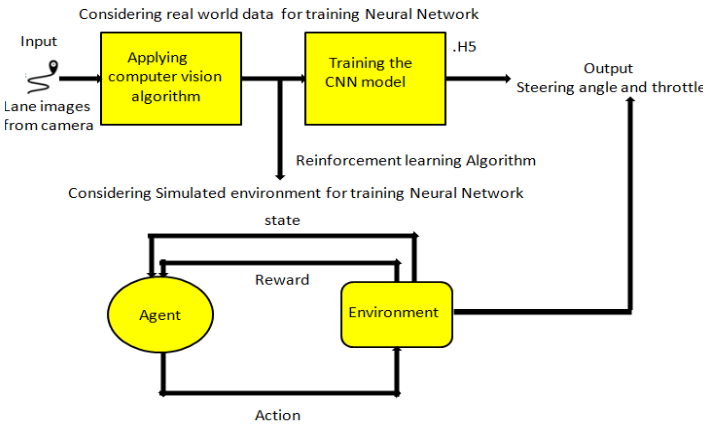

Considering real world data for training Neural Network
Input
Lane images from camera
Applying computer vision algorithm
Training the CNN model
.H5
Output
Steering angle and throttle
Reinforcement learning Algorithm
Considering Simulated environment for training Neural Network
state
Reward
Agent
Environment
Action

Fig 1.Block Diagram

C. Reinforcement learning:The agent (Intelligent automobile) takes action and interacts with the environment (lane or track). The environment returns rewards (points)and moves to the next state. Through multiple interactions, the agent gains experienceand seeks the optimal strategy in experience. This interactive learning process is similarto the human learning style. Its main features are trial and error and delayed return. The learning process can be represented by the Markov decision process. The Markov decision process consists of triples 'S, A, P, r'.

• S-State (images)
• A-Action (Based on the set of throttle values and steering angle robot tries to take every action)
• r-Reward (Positive points are given if the robot moves in the track negative point for going out of the track)

$$S=S_1,S_2,S_{3,\ldots}(1)$$
$$A=a_1,a_2,a_{3,\ldots}(2)$$
$$P_{ss'}^{a}=P[S_{t+1}=s'|S_t=s,A^t=a](3)$$
$$r=r(s,a)(4)$$

S is a collection of all states; A is a collection of all actions; P is the state transitionprobability; means the transitionprobability when the agent takes action A and change states tos'. The r is reward function, which means the reward of taking action a under states. The agent tries to increase the cumulative reward during each episode in a simulation environment. Thejson file generated during training in simulation is used to evaluate our model in real world.

IV. SOFTWARE ARCHITECTURE

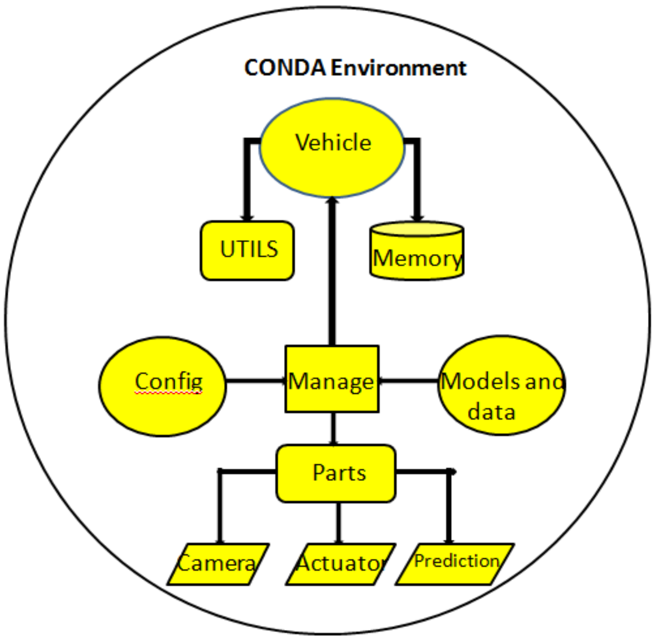
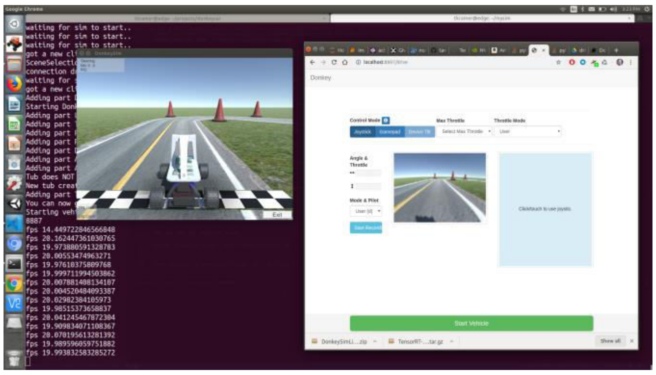


Fig 2. Software Architecture

• Vehicle: This is a main loop where all the parts having run () method is called here.
• Utils: Image processing utils from OpenCV, Linear algebra operations like vector operation and normalising the pwm values between -1 to 1 is provided here. These operations can be easily computed in Manage.py by inheriting this class.
• Memory: All the inputs and outputs variables passed as parameter to various parts are initialised here and the memory for the same is allocated. This is like a global variable configuration.
• Manage: All the parts like Actuator, object detection, web controller are called in drive loop. The drive loop frequency is 30HZ.
• Config: Various global variable configuration like CAMERA TYPE= "PICAMERA" can be set here.
• Models and Data:Models folder contains .h5 file which is the trained model file obtained from GoogleColab used to predict steering and throttle. Data folder contains various TUB folder which has .jpg images along withjson files containing
Steering, throttle, timestamp and its corresponding image file reference.
• Parts: All the actuators like camera, servo, and esc should have python files to fetch and publish data using GPIO pins through which they are connected to RPI or PCA board. Here in parts folder all these related python files are stored. All the business Logic files required for working of autonomous mode like keras model python file is contained within this folder.

V. SIMULATION

The entire software stack is written in Python 3.6 except for the view layer of the web server, which had to be written using HTML and JavaScript. Standard libraries for working with threads and sockets have been used.Machine learning libraries have been used to train the network. Those were NumPy,Sklearn, Pandas, Tensorflow and Keras.



```
waiting for sim to start..
waiting for sim to start..
waiting for sim to start..
got a new cl
SceneSelecti
connection d
waiting for
got a new cl
Adding part
Starting Don
Adding part
Adding part
Adding part
Adding part
Adding part
Adding part
Adding part
Tub does NOT
New tub creat
Adding part
You can now
Starting veh
8887
fps 14.449722846566848
fps 20.162447361030765
fps 19.973880591328783
fps 20.00553474963271
fps 19.97610375809768
fps 19.999711994503862
fps 20.007881408134107
fps 20.004520484093387
fps 20.02982384105973
fps 19.98515373658837
fps 20.041245467872304
fps 19.909834071108367
fps 20.070195613281392
fps 19.989596059751882
fps 19.993832583285272
```

Fig 3.Simulation

The simulation was done to train the vehicle or can also be used to check the trained model using a package as shown in Figure 3. When the vehicle is trained to mimic the driving of the user, the training aims to reduce the losses by increasing the number of the iterations as shown in Figure 4.

VI. RESULTS

The blue line indicates the desired nature (user) and the orange line (trained model) is the attempt to match with the blue line. The orange line should be as near as possible but should not coincide with the blue line.

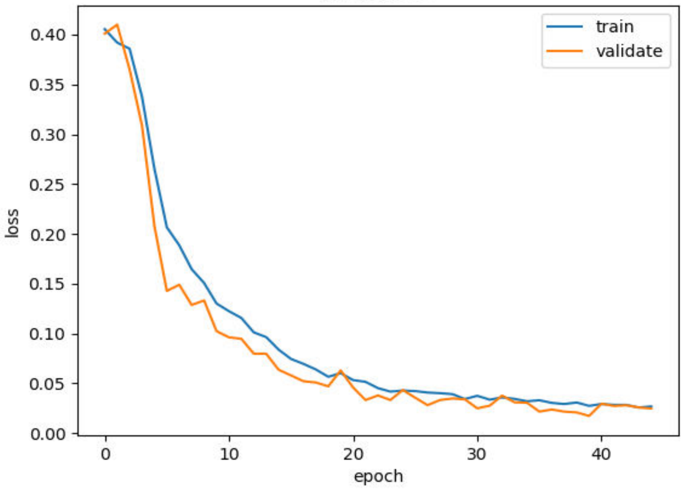


Fig 4.Model Loss

The lines must not overlap each other which may result in biasing.The Figure 5 shows the comparison between the user data and the trained data. The steering and throttle data which the user had used to drive through the track. The orange coloured line is the trained model which has attempted to match the user driving which is represented by the blue coloured line. As it can be seen the model has almost accurately tried to mimic the driving features of the driving of the user.

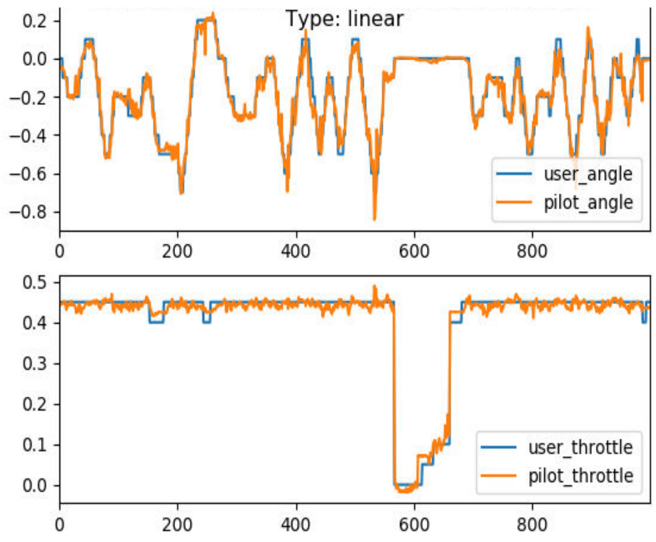


Fig 5. Comparison between user data and trained data

VII. CONCLUSION

Due to implementation of machine learning, the need of writing every rule in the code is eliminated which optimizes the memory usage. As the system can train itself, the vehicle can detect the obstacles just with the help of computer vision. This eliminates the need for expensive sensors like LIDAR[7]. The industry environment can be simulated and the vehicle can be trained in this simulation which reduces the possibility of crashes or wear and tears. Due to all the above mentioned advantages compared to the previous works, the proposed methodology is very cost effective and can be implemented with the help easily available hardware and software components. The vehicle proposed here aims to reduce the overall cost of an intelligent automobile used in an industry environment and use the resources as efficiently as possible.

VIII. FUTURE SCOPE

The vehicle can be equipped with more number of sensors which in turn gives the vehicle better data and hence the autonomous driving is further smoothed. IMU sensors can be incorporated into the vehicle to improve the obstacle avoidance or be better prepared for an obstacle. Using the map generated implementing A* algorithms such that robot can take the shortest path.

REFERENCES

[1] M. G. Bechtel, E. Mcellhiney, M. Kim and H. Yun, "DeepPicar: A Low-Cost Deep Neural Network-Based Autonomous Car," 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Hakodate, 2018, pp. 11-21. doi: 10.1109/RTCSA.2018.00011

[2] McFall K. (2016) Using Visual Lane Detection to Control Steering in a Self-driving Vehicle. In: Leon-Garcia A. (eds) Smart City 360°. SmartCity 36 2016, SmartCity 360 2015. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 166. Springer, Cham

[3] B. Blaga, M. Deac, R. W. Y. Al-doori, M. Negru and R. D_anescu, "Miniature Autonomous Vehicle Development on Raspberry Pi," 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, 2018, pp. 229-236. doi: 10.1109/ICCP.2018.8516589

[4] Robotics and Computer-Integrated Manufacturing Volume 26, Issue 4, August 2010, Pages 296-311

[5] www.donkeycar.com

[6]Blaga, M. Deac, R. W. Y. Al-doori, M. Negru and R. D_anescu, "MiniatureAutonomous Vehicle Development on Raspberry Pi," 2018 IEEE 14th InternationalConference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, 2018, pp. 229-236. doi: 10.1109/ICCP.2018.8516589

[7] Robotics and Computer-Integrated Manufacturing Volume 26, Issue 4, August2010, Pages 296-311