IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Enhancing Autonomous Navigation Through Transfer Learning For Path Detection

1Mahadevi K C, 2ASHWINI K S, 3Sowmya B
[1]Lecturer, [2]Lecturer, [3]Senior Scale Lecturer
[1,2]Department of Electronics & Communication Engineering
[1,2]Department of science
[1,3]Government Polytechnic, Nagamangala, Karnataka, India
[2]Government Polytechnic, Chamarajanagar, Karnataka, India

Abstract: The goal is to demonstrate a proof of concept for using transfer learning to address computer vision challenges in real time. This project involves developing a neural network-based vehicle control and driver assistance system focused on path detection and obstacle detection, enabling the bot to "learn" optimal responses for tasks like obstacle avoidance and path navigation toward a target destination. Initially, the bot will navigate along a given path, with the resulting performance data serving as the learning foundation. This learned information will then be applied in a more complex environment featuring obstacles and traffic elements such as signals and STOP signs. The objective is for the bot to traverse the course, avoid obstacles, and continuously learn as it moves. Transfer learning is advantageous in this context, as it reduces computational resource demands and time, achieving high accuracy efficiently. This problem is particularly challenging, combining areas like computer vision and machine learning, and requiring real-time processing capabilities. Python will be used for implementing the learning algorithms due to its wide availability and extensive library support, which makes it user-friendly and effective. The bot itself will be compact, no larger than a slice of bread, with a mounted microprocessor and microcontroller that enable it to methodically navigate the specified path. Powered by motors, the bot will autonomously and gradually work its way through the environment, showcasing the potential of transfer learning in real-time computer vision applications.

Index Terms **–** Transfer learning, neural networks, network topology.

I. INTRODUCTION

A neural network is a computational model inspired by the structure and functionality of the human brain. Significant progress has been achieved in areas like natural language processing, image processing, and procedural generation using Artificial Neural Networks (ANNs) [1]. These biologically inspired systems perform various machine learning tasks, including clustering, classification, and pattern recognition. Despite these advancements, neural networks tackle relatively "simple" problems compared to their inspiration, the complex human brain. A neural network mirrors the brain in two primary ways: it acquires knowledge (K) through experience (E), or "learning," and this knowledge is retained and improved through synaptic weights, which are connection strengths between neurons [2].

Conceptually, neural networks can be represented as a directed graph, with neurons as nodes and directed edges (with weights) as the connections linking neuron outputs to inputs. The neural network processes information (input) from the external environment, typically in the form of a vectorized image denoted by X(n), where 'n' represents the number of inputs and X is the operating function [3]. Each input is scaled by its associated weight, a value between 0 and 1 that helps the network solve problems. These weights indicate connection strength, and the weighted inputs are summed in a computing unit (an artificial neuron). If this sum

is zero, a bias is added to ensure a non-zero output or to adjust the response intensity. The bias always has a weight and input set to '1'. The result is a numerical value ranging from 0 to infinity [4].

To achieve the desired output, the response is moderated through a threshold value, accomplished by passing the sum through an activation function. This function shapes the output, with options of linear or nonlinear activation functions [5].

1.1 Transfer Learning

Transfer learning is a widely used machine learning technique that addresses today's data science challenges. This approach leverages a model developed for one task as a foundation for solving a new, but related, task. Unlike traditional machine learning, transfer learning relies on pre-trained models—those already trained for a specific task—as a starting point to expedite the development of a model for a new task or problem [6]. By reusing pre-trained components, transfer learning can significantly reduce both the time and resources needed for model training, enabling faster development and more immediate results. This process allows practitioners to achieve effective models quickly, benefiting from the knowledge embedded within previously trained models [6].

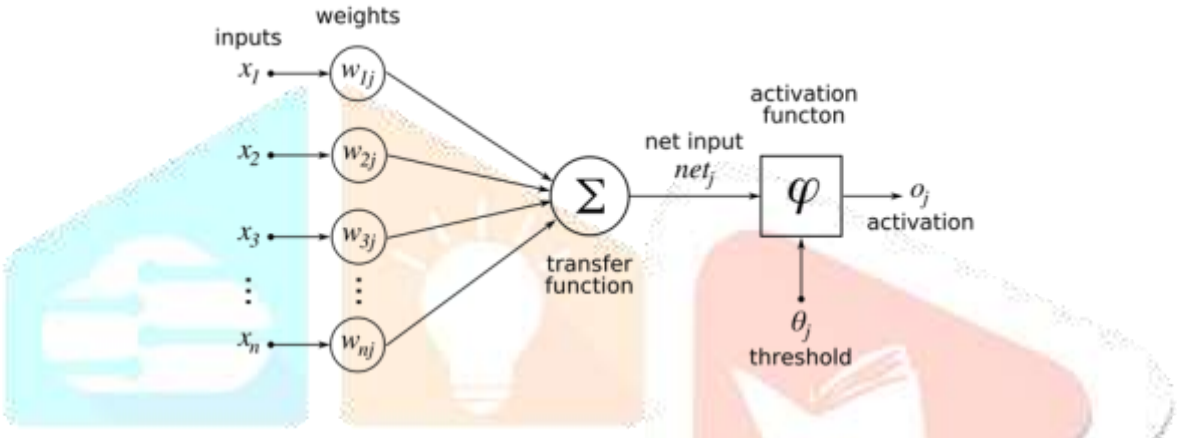


Figure 1.1: Abstract view of a neural network

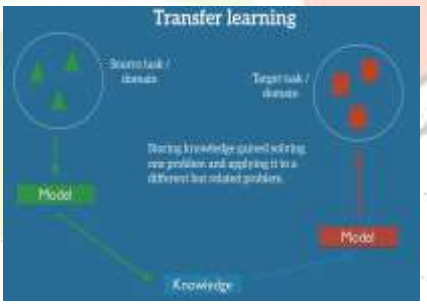


Figure 1.2: Transfer learning process visualization

1.2 Network Topology

A network topology is the arrangement of a network along with its nodes and connecting lines. According to the topology, ANN can be classified as the following kinds: Feed-forward Network: It is a non-recurrent network having processing units/nodes in layers and all the nodes in a layer are connected with the nodes of the previous layers. The connection has different weights upon them. There is no feedback loop means the signal can only flow in one direction, from input to output. It may be divided into the following two types − Single layer feed-forward network − the concept is of feed-forward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer. Multilayer feed-forward network − the concept is of feed-forward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.

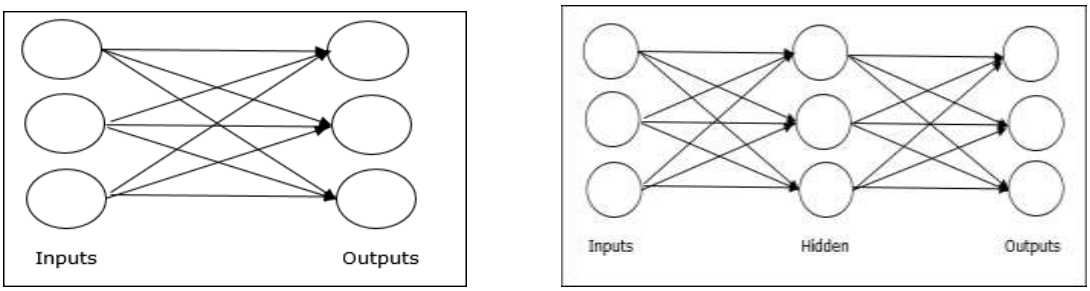


Figure 1.3: ANN general architecture

ANNs have the ability to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex. ANNs can generalize; after learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables (like how they should be distributed). Additionally, many studies have shown that ANNs can better model heteroskedasticity i.e. data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in financial time series forecasting (e.g. stock prices) where data volatility is very high [7].

Path detection plays a critical role in the development of autonomous navigation systems and driver assistance technologies. Accurately identifying a path is fundamental for robotic navigation and self-driving vehicles, where a clear understanding of the route is essential to ensure safety, efficiency, and adaptability in dynamic environments. Traditional approaches to path detection often involve complex, resource-intensive computations and substantial data requirements, posing challenges in real-time scenarios. Transfer learning offers a promising alternative by enabling models to leverage knowledge from pre-trained neural networks, significantly reducing training time and computational demands while maintaining high accuracy[8].

In this work, we explore the application of transfer learning to path detection in real-time navigation contexts. By adapting a pre-trained model for the task of path detection, the system can efficiently identify routes while adjusting to environmental variations without the need for extensive retraining. This paper discusses the methodology and implementation of transfer learning in path detection, highlighting how leveraging previously acquired knowledge enables faster adaptation to new environments. Furthermore, this approach enhances the system's performance by reducing the dependence on massive labelled datasets, a common limitation in traditional machine learning approaches[9].

Self-driving vehicles have been in the imaginative centers of human minds since the invention of automobiles. It is realized today due to the improvement in computer science and technology in the past couple decades. Path detection and real time object detection are crucial aspects of self-driving vehicles and this project aims to act as a proof of concept of transfer learning in the real time path and object detection department.

An automatic "bot" that is not controlled by anything but the computer system that does real time processing, will navigate its way through a given obstacle course. This might be seemingly unimpressive, but the fact that it is real time object detection, processing, path detection and processing, is operation intensive. It learns from absolutely nothing, about the path and the obstacles it should avoid.

When the bot is in a completely new environment, it will learn to apply what it has previously learned in that environment. Thus, it navigates through the new environment in lesser time, as it already possess the data output from the previous task. The input will be the real-time video feed which will be broken into their respective frames and broken down further into individual images. The processing will happen on a remote machine which will communicate with the bot wirelessly. The processing will be done by a neural network and there are various other sub-tasks, apart from path detection, such as object detection (STOP sign), and effective wireless communication with the bot. Human input for new tasks every time is exhaustive. Every time there is a possibility of encountering new obstacles and it is error prone. The application areas are endless, ranging from space rovers to nanobots inside the human body for medicine delivery.

The findings of this study contribute to advancements in autonomous navigation, offering a streamlined, resource-efficient method for achieving robust path detection, even in challenging, unpredictable scenarios.

II. Related Work

R-CNN and its successor, Fast R-CNN, use a "classical" approach known as selective search to locate potential object regions in images. However, selective search is a slow process (around two seconds per image), creating a bottleneck for Fast R-CNN. To improve this, a neural network-based method called the Region Proposal Network (RPN) was introduced, which reuses the same features utilized by Fast R-CNN, reducing region proposal time to about 10 milliseconds. The Fast R-CNN network has three key components: a "feature extraction network" (FEN), initialized with pre-trained weights (e.g., from VGG16) to extract features; the Region Proposal Network (RPN), which produces region proposals using these features [10]; and a "classification network" (CN), which classifies each region proposal and adjusts bounding boxes. While only the RPN is explicitly named in the literature, the other components are essential to the system's operation. The testing process involves converting the image to features via FEN, applying RPN to generate region proposals, using Region of Interest Pooling (RoI-Pooling) to resize each proposal's features, and applying CN to predict object classes (from K classes and 1 background class) and adjust bounding box dimensions.

One drawback of R-CNN is its lengthy training time due to the need to classify approximately 2000 region proposals per image, making real-time implementation infeasible (47 seconds per test image). Since selective search is a fixed algorithm, it lacks learning capabilities.

This paper presents methods addressing high-precision vehicle detection and distance measurement within driver assistance systems, based on the principles of monocular vision. This two-step process includes vehicle area extraction and verification, providing a viable alternative to existing detection and distance measurement methods, such as laser radar, ultrasonic ranging, RFID, and vision-based techniques. As the number of vehicles has increased, so has traffic congestion and accident rates, particularly rear-end collisions. Vision-based methods are affordable and information-rich, but binocular vision-based distance measurements have complex algorithms and application limitations [11]. Recent studies have proposed single-camera distance measurement, yet these methods largely depend on lane line detection and are impractical on unstructured roads.

This study proposes a monocular vision-based vehicle detection and distance measurement method, combining high precision, ease of application, and versatility. To improve detection accuracy, it uses Uniform Local Binary Pattern (ULBP) texture histogram features and a k-nearest neighbour classifier to verify vehicle presence. Distance measurement uses an off-line camera calibration and an on-line calculation method for accuracy. The vehicle distance is calculated from the shadow point at the rear of the preceding vehicle to the projection point on the roadbed at the head of the detecting vehicle, with a mean absolute error of 0.8058 m, surpassing traditional methods. The method comprises three steps: off-line camera calibration, preceding vehicle detection, and vehicle distance measurement. Distance between vehicles is derived as d=d1+d2d = d1 + d2d=d1+d2, where d1d1d1 is the straight-line distance from point A (vehicle head) to Q, and d2d2d2 is the distance from Q to B, with Q representing the closest visible road point [12].

A convolutional model is trained under a fully supervised setting, and its features are then used in general vision tasks. Proper training of this model is essential, as it must apply learned features to generic challenges that may vary significantly from initial tasks. This raises questions such as: Do CNN-extracted features generalize well to other datasets? How does feature performance compare with depth? These questions are addressed both quantitatively and qualitatively [13].

III. Proposed Methodology

The figure shows block diagram of the system. The ultrasonic sensor, servo motor and L298N motor driver are connected to Raspberry pi 3 through GPIO pin while Pi camera is connected to Raspberry pi 3 through USB 2.0. The raspberry Pi 3 was powered by a 5V power bank and the DC motor powered up by a 7.2V battery. The DC motor was controlled by L298N motor driver, but Raspberry Pi 3 send the control signal to the L298N motor driver and control the DC motor to turn clockwise or anti clockwise. The servo motor is powered up by 5V on board voltage regulator at L298N motor driver from 7.2V battery. The ultrasonic sensor is powered up by 5V from Raspberry Pi 3 5V output pin. Raspberry pi 3 itself acts as a computer in our project. The movement of the RC car is controlled by Raspberry pi. Along with all these a pi camera is required that perform the computer vision with the help of OpenCV library. The algorithm for path detection is written in Python. The algorithm is written to capture the image/video using the pi camera and then process it to detect the path so that it helps the servo motors to decide whether to take left or right. The position of pi camera is very important which allows the algorithm to get us the best results. An ultrasonic sensor is attached to the car. When an obstacle is detected at certain range while the car is moving, the car tries to avoid the obstacle either by stopping or by overtaking it by changing the lane. Henceforth the car can overcome obstacles. Hardware components required are Raspberry PI Board (B+ Model), PI Camera module V2, Arduino, HC-SR04 Ultrasonic sensor,

Remote computer, BOT and software requirements are: TCP Server setup, Arduino IDE, Raspbian OS, Python library - OpenCV

For Software parts, Python (IDE) is used in our project. The L298N motor driver, ultrasonic sensor and servo motor and the pi camera which are controlled by Raspberry pi in which coding's are done in Python language. The Raspberry Pi 3 will act as the TCP server and the monitor will act as the TCP client. So, all the user input data sent from the camera and sensors will be received by Raspberry Pi 3 TCP server through the Raspberry Pi 3 hotspot and this will make the project portable without connect through a router.

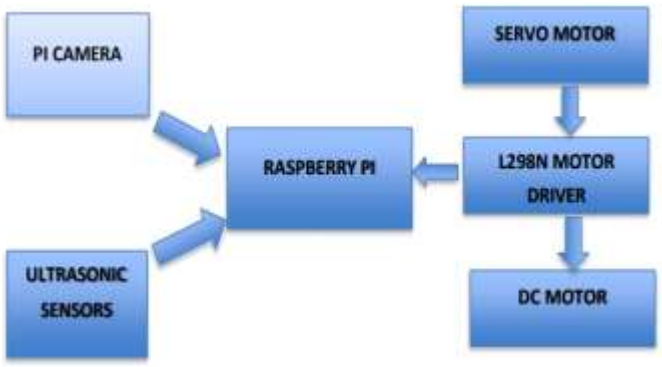


Figure 3.1: System block diagram

All the parts of hardware's and software are combined. The python cod.es will be executed at Raspberry Pi 3. Pi camera will be connected to the monitor through Raspberry pi 3 using TCP server. Pi camera send live feeds to the monitor.

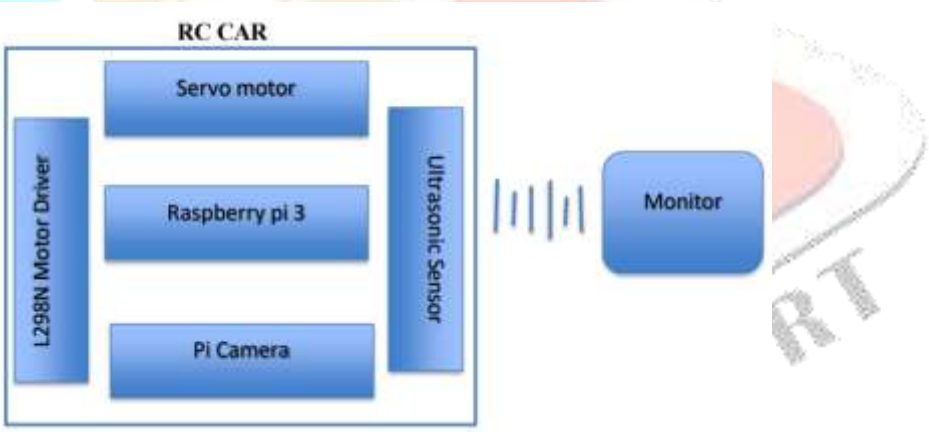


Figure 3.2: Final product block diagram

Prototyping model was used as the methodology for this paper. A prototype is the demo version of the final product and the prototype act as a sample to test the process. Prototype model is defined as a method of system development. Prototype model consists the following phases as Figure below.

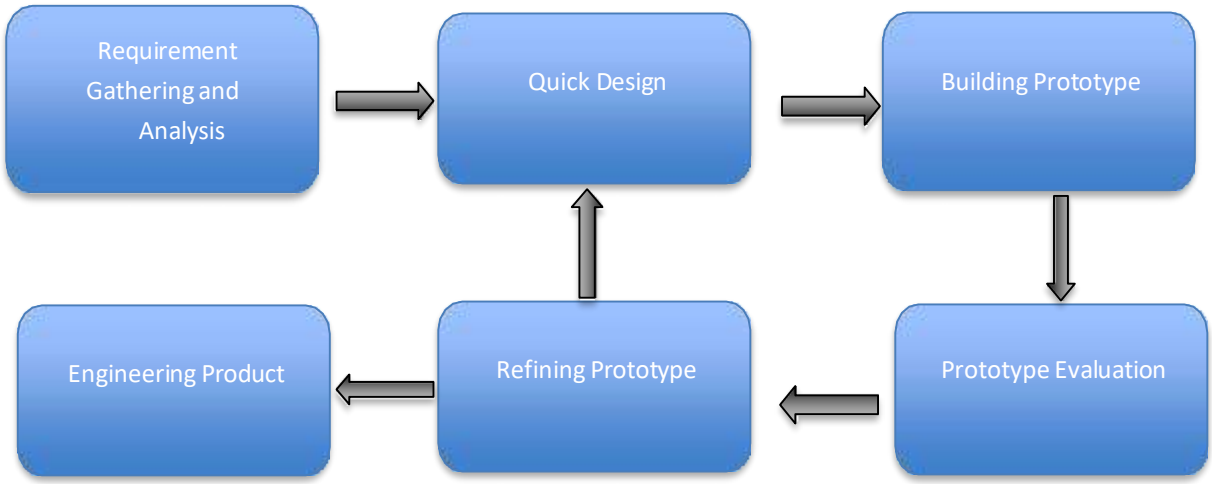


Figure 3.3: Block Diagram of proposed system

3.1 Requirement Gathering and Analysis

Before a project has started, all the requirement hardware's and software's must be gathered. And then it is to be analyzed if we have all the required things in hand and everything is working properly. This involved buying of hardware's such as Raspberry pi 3, Pi Camera, chassis/RC car, Arduino, L298N Motor Driver, DC motors, SD card, Servo motor, and the Ultrasonic Sensor. Then we had to download some software's like Noob and Raspbian OS for Raspberry pi 3, Arduino IDE for Arduino.

3.2 Quick Design

A quick design about the system has been generated after the requirement gathering and analysis stage. The important components of the system must be included in the design. By using the quick design, we were able to know how the system works. This involved building the bot, fixing up pi camera, booting up Noob and Raspbian OS into Raspberry pi 3, and majorly designing of the algorithm required. Then we fixed the pi camera to the raspberry pi 3 and tested it to check if the camera works properly and to see the quality of the image obtained.

3.3 Building prototype

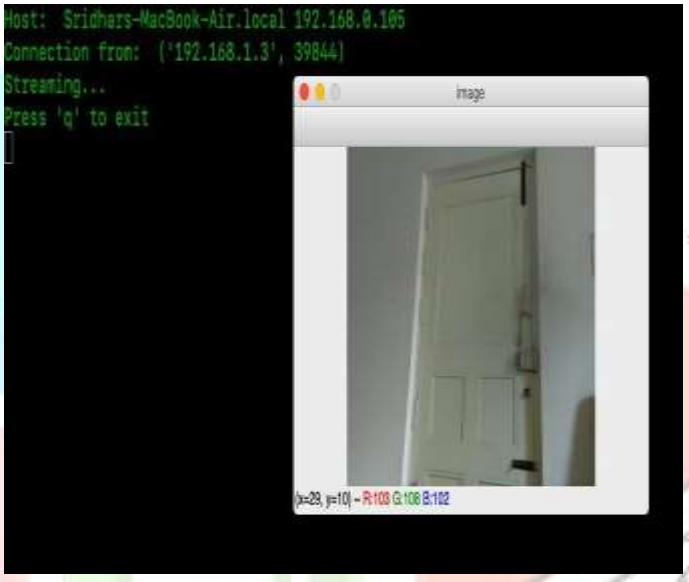


Figure 3.4: Pi Camera working test

Before the prototype is developed, all the information that get from the previous stage design is gathered and modified. That information become the minimum requirement for the system. The first step in building a prototype is to connect Raspberry pi 3, L298N and DC motors.

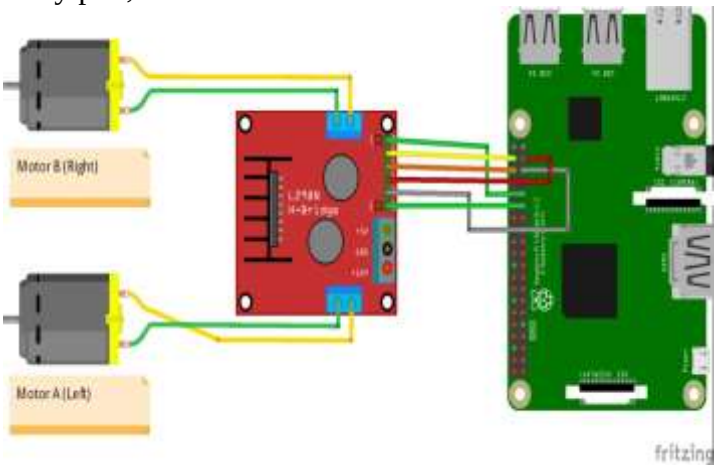


Figure 3.5: Connection of Raspberry Pi 3, L298N and DC motor

The second step was to design a bot which basically would be a RC car, and then fix the Arduino, DC Motors, L298N Motor driver, sensors and a Pi Camera.

Figure 3.6: Fixing up the BOT

- Prototype Evaluation:
  In the process of developing the prototype, the prototype is evaluated with results obtained by testing. The hardware parts are connected to each other and to the BOT along with the sensors and Pi camera. First it is tested if the connections are done right, and then it is to test if the data is flowing properly. Sensors are to be tested and the Pi camera is checked for proper quality of video & image.
- Refining Prototype:
  In this phase, the prototype will be modified depending on the requirement. If the user evaluation is good, the finalized system will be created with the finalized prototype specification. And if there is any failure or misconnection, it is corrected. If there is chance of improvising the prototype, it is to be done.
- Engineering Product:
  In this stage, the evaluation and testing must be done on the final product to minimize the maintenance. The Bot is tested again and again to make it learn and to adapt to different cases. The Bot keeps learning and uses its knowledge for new challenges faced.

3.4 Flowchart

In this paper we have hardware's like Ultrasonic Sensors, Pi camera, Raspberry Pi, Arduino, and A bot which monitors the surroundings to detect the path and obstacles. All the hardware's are interconnected. The sensor signals and videos from pi camera are sent from Arduino to Raspberry Pi. Raspberry Pi is a Linux based operating system works as a small pc processor system. Raspberry pi processes the captured information and releases the appropriate results to the bot. Raspberry pi is connected to internet and a monitor which will have live footage running which will be captured from the pi camera. This enables one to monitor the bot easily. The raspberry pi acts as a server once it is connected to internet. Then the server automatically sends the data to the bot. between image/video being captured and being sent to the raspberry pi, there comes some more tasks that involves Open CV Neural Network. The tasks like distortion removal, boundary & edge detection and obstacle detection & verification.

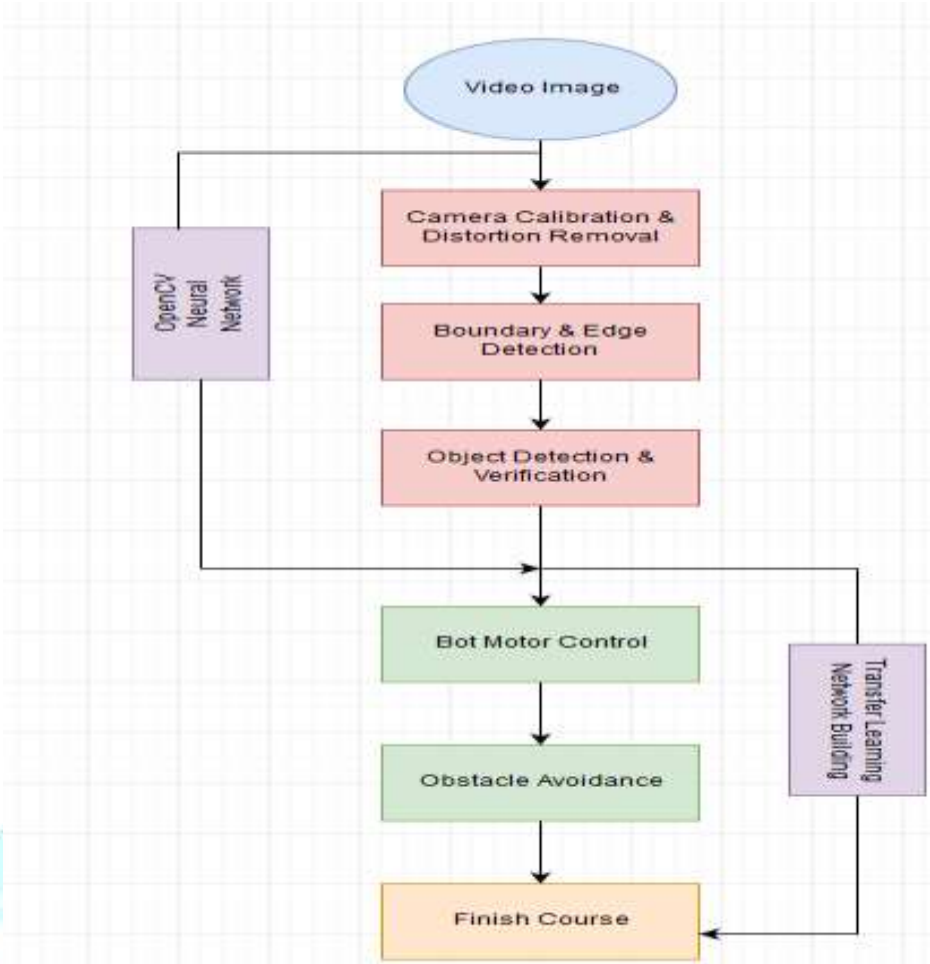


Figure 3.7: Fixing up the BOT

IV. TEST CASES AND RESULTS

4.1 Implementation of open CV

Here we make use of OpenCV libraries for Image Processing. Hence, OpenCV library is to be installed in IDLE/Python IDE Audio Model.

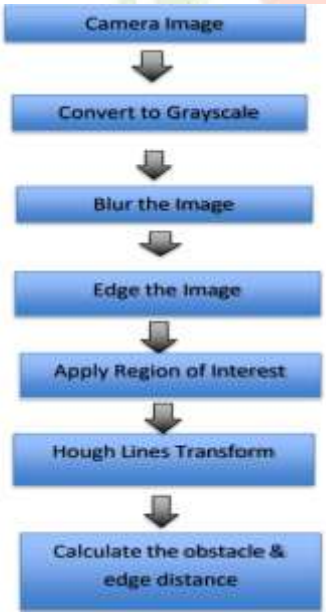


Figure 4.1: Implementation of open CV

- Using the OpenCV library we can capture the video. VideoCapture() would be the function used. The image is captured from pi camera and then it has to be sent to get converted to grayscale.
- The below figure shows the captured image being converted to grayscale image.

Figure 4.2: Grayscale Image

- Then the function cv2.GaussianBlur(img, (kernel_size, kernel_size), 0) was used to blur the image to reduce the noisy lines in the image. The kernel size used was based on the input image.

Figure 4.3: Blurred Image

- Then the cv2.Canny(img, low threshold, high threshold) was used to get the edge of all the lines in the image. The low threshold and high threshold are the values used to filter the lines in the image.

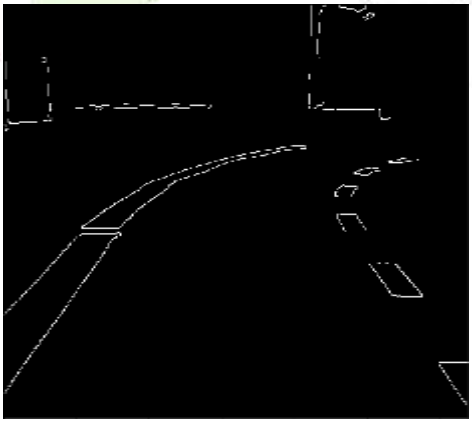


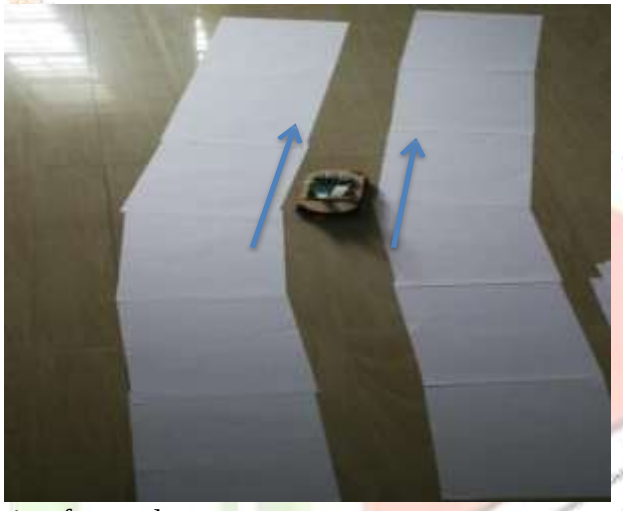
Figure 4.4: Edge Image

4.2 Test case

- Test case 1: The RC car is placed in the track. The diagram shows that the wheels are placed parallel to the tracks. The Bot starts

Figure 4.5: Test case 1

- Test case 2: The processing starts with pi camera sending the image to the pc through Arduino. The Raspberry pi 3 processes it and gives back the direction accordingly. Here, since the path



is clear the car starts moving forward.

Figure 4.6: Test case 2

- Test case 3: The diagram shows the car stopped due to an obstacle being in front. The car uses sensors and pi camera to detect the obstacle. If found the processor tells the bot to stop.

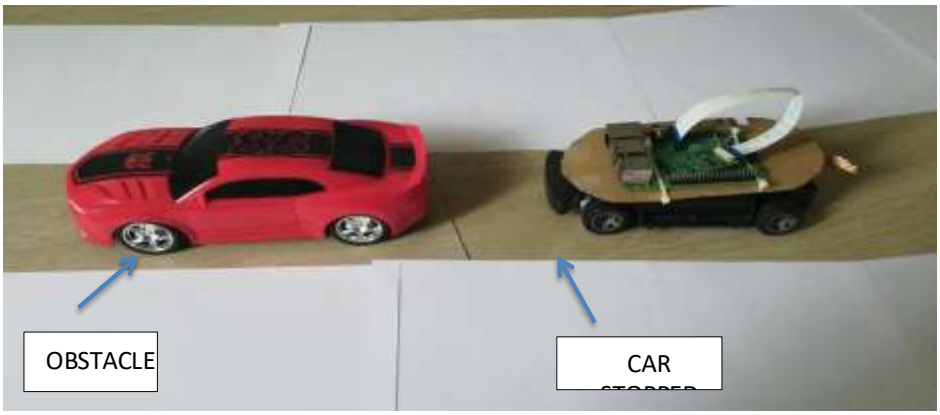


Figure 4.7: Test case 3

- Test case 4: Below figure shows simulation of neural network. Below are three simulations of neural network at 3 different generations. The generation 6 has more training than other two and hence the cars in that are more sustainable.

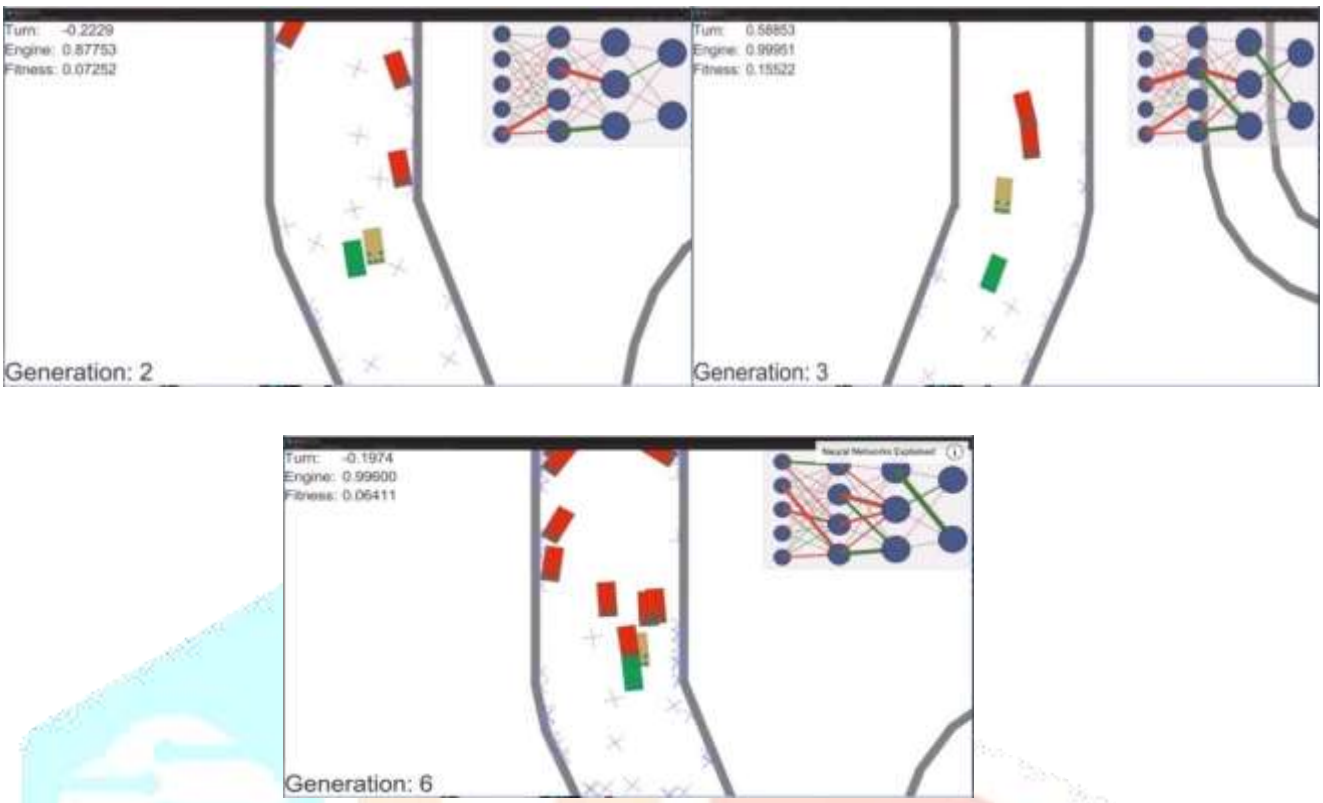


Figure 4.8: Simulations

V. CONCLUSION

All hardware components were mounted on the RC car and tested with basic code to ensure each component functioned correctly. Some objectives have already been met, including creating a low-power, low-cost prototype and establishing a TCP connection between a keyboard and the BOT/RC car for manual mode control. The primary objective, however, is for the RC car to navigate independently, following the track and avoiding obstacles using computer vision techniques. Additional features will also be incorporated, focusing on enhancing path detection and edge/boundary detection. Path detection and edge detection have been successfully implemented, but the obstacle detection algorithm requires further refinement, as test results indicate it is not yet optimal. Currently, the path detection and ultrasonic sensor run on the same core, which limits the RC car's efficiency. For smoother and faster track completion, a more advanced lane-tracking algorithm and a more powerful processor are necessary for image processing tasks. Since the Raspberry Pi 3 has limited processing power and operates on a single core, the system experiences some bottlenecks. Additionally, as this is a prototype, the RC car's components—Raspberry Pi 3 and Motor Driver L298N—may overheat during extended operation. In autonomous mode, the Raspberry Pi must continuously process images from the Pi Camera, further straining its capacity. Moreover, the autonomous mode struggles in low-light environments or when there are distracting elements beside the track, which interfere with the lane-tracking algorithm.

VI. ACKNOWLEDGMENT

We extend our heartfelt gratitude to our mentors and colleagues for their invaluable support and guidance throughout this work. We also appreciate the resources provided by the research community that facilitated our work. Finally, we thank our families and friends for their unwavering encouragement during this journey.

REFERENCES

[1] Training very deep networks (2015), R. Srivastava et al.
[2] How transferable are features in deep neural networks? (2014), J. Yosinski et al.
[3] Decaf: A deep convolutional activation feature for generic visual recognition (2014), J.Donahue et al.
[4] You only look once: Unified, real-time object detection (2016), J. Redmon et al.
[5] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (2015), S.Ren et al.
[6] An analysis of single-layer networks in unsupervised feature learning (2011), A. Coates et al.
[7] A novel method of preceding vehicle detection and vehicle distance measurement based on monocular vision, Chunhong Zhang et al
[8] TensorFlow: Large-scale machine learning on heterogeneous distributed systems (2016), M. Abadi et al.
[9] Imagenet large scale visual recognition challenge (2015), O. Russakovsky et al.
[10] Ask your neurons: A neural-based approach to answering questions about images (2015),M. Malinowski et al.
[11] D.N. Perkins and G. Salomon, Transfer of Learning. Oxford, England: Pergamon, 1992.
[12] [2] S.J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[13] Z. Wang, Z. Dai, B. Poczos, and J. Carbonell, "Characterizing and ´ avoiding negative transfer," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, Jun. 2019, pp. 11293–11302.
[14] K. Weiss, T.M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," J. Big Data, vol. 3, no. 1, Dec. 2016.
[15] J. Huang, A.J. Smola, A. Gretton, K.M. Borgwardt, and B. Sch¨olkopf, "Correcting sample selection bias by unlabeled data," in Proc. 20th Annual Conference on Neural Information Processing Systems, Vancouver, Dec. 2006, pp. 601–608.
[16] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. Bnau, and M. Kawanabe, "Direct importance estimation for covariate shift adaptation," Ann. Inst. Stat. Math., vol. 60, no. 4, pp. 699–746, Dec. 2008.
[17] O. Day and T.M. Khoshgoftaar, "A survey on heterogeneous transfer learning," J. Big Data, vol. 4, no. 1, Dec. 2017.
[18] M.E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," J. Mach. Learn. Res., vol. 10, pp. 1633–1685, Sep. 2009.
[19] H.B. Ammar, E. Eaton, J.M. Luna, and P. Ruvolo, "Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning," in Proc. 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Jul. 2015, pp. 3345–3351.
[20] P. Zhao and S.C.H. Hoi, "OTL: A framework of online transfer learning," in Proc. 27th International Conference on Machine Learning, Haifa, Jun. 2010, pp. 1231–1238.
[21] O. Chapelle, B. Schlkopf, and A. Zien, Semi-supervised Learning. Cambridge: MIT Press, 2010.
[22] S. Sun, "A survey of multi-view machine learning," Neural Com-put. Appl., vol. 23, no. 7–8, pp. 2031–2038, Dec. 2013.
[23] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," 2013, arXiv:1304.5634v1.
[24] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," Inf. Fusion, vol. 38, pp. 43–54, Nov. 2017.