iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70278

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

NavDrive: Next-Gen Automated Vehicle Driving with Real-time Intelligent Vision and Engineering

Mr. Bijendra Kumar, Sagar Saini, Devansh Kumar, Eshaant Tyagi
Electronics and Communication Engineering, Shriram Group of Colleges, Muzaffarnagar

Abstract: ***The Development of Autonomousvehicles has increased significant attention due to their potential to enhance safety, efficiency and accessibility in transportation. This study focuses on the development of NavDrive, a low-cost autonomous vehicle prototype, which is possible by using the raspberry pi 3, Arduino and the machine learning concept called Behavioral Cloning(BC) – Imitation with the Convolutional Neural Network(CNN). the methodology includes three key faces, data collection, model making, implementation. The data collection by driving the car manually on the custom design indoor track using the white sheets and black tape. The prototype is controlled using the rf based transmitter and receiver. The prototype is equipped with the raspberry pi camera and various other modules ,for the training of the sequential CNN model to mimic the human driving behaviorand the deployment of the model on the raspberry pi to enable the autonomous navigationThe outcomes of this project demonstrate the feasibility of creating a functional autonomous vehicle using readily available and affordable hardware, offering valuable insights into the challenges and successes of such an implementation and highlighting its potential as an educational prototype for exploring autonomous driving principles.***

Keywords:Autonomous Vehicles, Self driving car, Raspberry Pi, Arduino, Behavior Cloning, Convolutional Neural Networks, Robotics.

I. INTRODUCTION

A. Background and Motivation

The growing global interest in autonomous vehicles (AVs) comes from their transformative ability in various social domains.[1] AVs promises more benefits, with increased traffic safety by reducing human errors, increasing transport efficiency through customized traffic flows and better access to individuals with mobility limits.[1] At the forefront of modern AV technology lies the integration of Artificial Intelligence (AI) and Machine Learning (ML).[5] These computationalparadigms provide vehicles to see their surroundings, make informed decisions and navigate without direct human intervention. At the same time, educational initiatives and even applications are a growing trend in robotics and autonomous systems to take advantage of low cost, built -in platforms such as Raspberry Pi and Arduino for rapid prototyping, educational initiatives, and even certain application deployments.[8] The range and versatility of these platforms make them ideal for the discovery of complex concepts with hands-on manner.

B. Behavioral Cloning for Autonomous Driving

Behavior cloning has emerged as a practical machine learning method for autonomous driving, where a smart agent acquires the ability to imitate the actions of an expert, in this case, a human driver.[12] Learning involves watching and saving the expert's behavior (e.g., steering inputs) against given environmental stimuli (e.g., camera images). By training a machine learning mode from this gathered data, the agent can then make predictions of the right actions in similar circumstances.[12] Behavioral cloning has a number of benefits, such as its relative conceptual simplicity and proven success in tasks like lane following and simple navigation.[14] Yet, it is also necessary to recognize the inherent shortcomings of this method. Its performance also relies a lot on the quality and variety of the training data, and the model will have difficulty generalizing to situations that are not well represented in the training set.[13]

C. Project Overview and Objectives

This research details the NavDrive project, an endeavor focused on developing a next-generation autonomous vehicle driving system. The core aim of this project is to integrate real-time intelligent vision processing with fundamental engineering principles to achieve cost-effective autonomous navigation. To realize this aim, we established the following objectives:

- Collect a comprehensive dataset of driving behavior on custom-designed indoor tracks using a Raspberry Pi and its camera.

- Train a convolutional neural network (CNN) using the behavioral cloning technique to accurately predict steering commands from visual input.
- Deploy the trained CNN model on the Raspberry Pi, enabling the robotic car to navigate the tracks autonomously.

D. Contributions of this Research

This work makes a contribution to the area of low-cost autonomous robotics in a number of significant ways. It offers a realistic and step-by-step implementation of behavioral cloning for autonomous driving using easily available and low-cost hardware components, that is, the Raspberry Pi and Arduino. This provides a valuable blueprint for researchers, educators, and hobbyists who wish to investigate this domain. Secondly, the work represents an all-encompassing methodology involving custom track data gathering, specific training procedures of models, and deployment on an embedded system. Such methodological richness provides a sense of practical insights and difficulties related to creating the system. Lastly, the success of the project in autonomous navigation, though in a controlled setting, highlights the promise of these low-cost platforms for prototyping higher-level autonomous systems and for use in educational settings. The recorded difficulties and solutions put into place also add to the body of knowledge in this fast-changing field.

II. LITERATURE REVIEW

A. Autonomous Vehicles: From theory to Practice

The idea of autonomous vehicles has developed considerably in the last century, moving from theoretical concepts to concrete realities[4] Early advancementsin history laid the foundation for current advanced systems, with contemporary AVs now being able to perceive and understand their surroundings through an array of sensors and respond to them with little or no human interaction.[1] Institutions such as SAE International have laid down standardized driving automation levels from Level 0 (no automation) to Level 5 (full automation under all situations).[2] Autonomous driving is supported by diverse technological strategies, such as sensor fusion that integrates input from a combination of sensors such as cameras, radar, and LiDAR to present a holistic representation of the environment.[1] Path planning algorithms selects the best course to a destination, while control systems implement the actions required to steer, accelerate, and brake the vehicle.[1]

B. Small-Scale Autonomous Driving Prototypes

The use of small-scale autonomous vehicle prototypes, particularly those built with Raspberry Pi and Arduino, has become increasingly prevalent in both research and educational settings.[8]These systems offer a cost-effective way to experiment with the fundamental principles of autonomous driving without the complexities and risks associated with full-scale vehicles.Projects have explored various aspects of autonomy, including lane detection, object recognition, and basic navigation.[8] The motivations behind such work often include educational goals, allowing students and researchers to gain hands-on experience with robotics, sensor integration, and software development for autonomous systems.[18] Furthermore, these prototypes serve as valuable testbeds for developing and evaluating new algorithms and techniques before their potential implementation in larger, more complex systems.

C. Behavior Cloning in Autonomous Driving

Behavior cloning has been extensively investigated as a method for enabling autonomous driving by mimicking observed human driving behavior.[12] This technique involves training a machine learning model to map sensory inputs, typically camera images, directly to control outputs, such as steering angles.[15] Research has shown the effectiveness of behavior cloning in tasks like lane keeping, where the model learns to predict the steering adjustments necessary to stay within lane markings.[14] Various neural network architectures, including Convolutional Neural Networks (CNNs), have been employed for behavior cloning in autonomous driving due to their ability to learn complex patterns from visual data.[14] While effective in many scenarios, behavior cloning models can face challenges when encountering situations not well-represented in the training data, potentially leading to unsafe driving behaviors.[13] Advancements in behavior cloning research include efforts to improve generalization by using larger and more diverse datasets, incorporating attention mechanisms, or combining behavior cloning with other learning paradigms.[14]

D. Convolutional Neural Networks (CNNs) for Vision-Based Autonomous Driving

Convolutional Neural Networks (CNNs) have become a cornerstone in vision-based autonomous driving due to their remarkable ability to extract hierarchical features from image data.

[14] The architecture of CNNs, with their convolutional layers, pooling layers, and fully connected layers, is particularly well-suited for processing spatial data like images, allowing them to learn patterns relevant for tasks such as object detection, lane segmentation, and steering angle prediction.[14] In the context of behavior cloning, CNNs can be trained to directly map raw pixel data from a camera to the desired control output, effectively learning an end-to-end driving policy.[15] Researchers have explored various CNN architectures for autonomous driving, often adapting or building upon well-established models like AlexNet, VGG, and ResNet to optimize performance for specific driving tasks.[14]

III. METHODOLOGY

A. Hardware Components

The NavDrive project utilizes a modular hardware component comprising a Raspberry Pi 3 Model B as the high-level processing unit, a Raspberry Pi Camera Module V2 for visual data acquisition, an Arduino Uno r3 for low-level motor and steering control, an L298N motor driver to interface with DC motors, four DC motors to power the RC car chassis, a standard servo motor SG90 for steering, and a 2.4GHz , CT6B transmitter/receiver (TX/RX) unit for data collection and optional manual control. These components are interconnected to facilitate data flow and control signals, as illustrated conceptually:

Component	Function
Raspberry Pi 3 B	High-level control, image processing, ML model execution
Pi Camera Module V2	Capturing visual data of the environment
Arduino Uno	Low-level motor control, steering control, manual driving input handling
L298N Motor Driver	Interface between Arduino and DC motors
LM2596S buck converter	Providing power to the components
DC Motors (4)	Providing locomotion to the RC car chassis
SG90-Servo Motor	Controlling the steering mechanism
CT6B TX/RX Unit	Enabling manual control of speed and steering
11.1 Volt 3S battery	To give power to the system
USB Cable	Facilitating serial communication between Raspberry Pi and Arduino

The Raspberry Pi, equipped with the Pi Camera, serves as the "brain" of the system, responsible for capturing and processing images, executing the trained machine learning model, and making high-level decisions regarding steering. The Arduino acts as a microcontroller dedicated to the precise control of the motors and the steering servo. The L298N motor driver acts as an intermediary, allowing the low-voltage signals from the Arduino to control the higher-power DC motors. The servo motor is responsible for the physical act of steering the vehicle. The CT6B unit provides a mechanism for manual intervention, allowing for human control of the car's speed and direction when autonomous mode is not engaged. Crucially, a USB cable establishes a serial communication link between the Raspberry Pi and the Arduino, enabling the Raspberry Pi to send predicted steering commands to the Arduino for execution.

B. Software Implementation

The software implementation of the NavDrive project spans two primary platforms: the Raspberry Pi and the Arduino. On the Raspberry Pi, the primary programming language is Python, chosen for its extensive libraries supporting image processing and machine learning. Key libraries utilized include OpenCV for real-time image acquisition and processing from the Pi Camera, and TensorFlow lite or Keras for building, training, and deploying the Convolutional Neural Network model, pySerial for serial communication, time, math for sensor data handling,threading for Running Code in Parallel.

The software architecture on the Raspberry Pi involves capturing images, preprocessing them, feeding them to the trained CNN model to predict steering values, and then transmitting these values to the Arduino via serial communication over the USB cable. On the Arduino, the programming language is C/C++, which is well-suited for real-time control of hardware components.
The Arduino code is responsible for receiving the steering commands from the Raspberry Pi over the serial interface and then using these commands to control the angle of the servo motor. Additionally, the Arduino manages the signals to the L298N motor driver to control the speed of the DC motors, potentially based on a fixed value or future predictions.

C. Behavioral Cloning Framework

The core of the autonomous driving capability in the NavDrive project lies in the application of the behavioral cloningframework. This involves establishing a direct mapping between the visual input captured by the Pi Camera and the desired control output, which in this case is the steering angle of the RC car. The underlying principle is to train a machine learning model to mimic the steering actions of a human driver in various driving scenarios. During the data collection phase, as a human operator manually drives the RC car around the custom tracks, synchronized data is recorded, consisting of images from the camera and the corresponding steering values at each point in time. This collected data then forms the training dataset for the behavioral cloning model. The goal of the training process is to enable the model to learn the complex relationship between the visual features in the images and the steering commands that resulted in successful navigation of the tracks. The choice of behavioral cloning was driven by its relative simplicity in implementation and its proven track record in enabling basic autonomous navigation tasks, particularly in well-defined environments.

D. Project Phases

1) Phase 1: Data Collection

The initial phase of the NavDrive project focused on gathering the necessary data to train the behavioral cloning model. This involved several key steps. First, a standard car chassis was selected as the base platform. The various hardware components, including the Raspberry Pi, Pi Camera, Arduino, motor driver, dc motors, servo motor, and CT6b unit, were carefully integrated onto this chassis. The placement of the camera was crucial, ensuring a forward-facing view that captured the track ahead. The Arduino was positioned to be easily connected to the motor driver and servo motor, while the Raspberry Pi was placed in a location allowing for convenient connection to the camera and the Arduino via USB.
Next, custom indoor tracks were designed and constructed. These tracks included an oval shape, a circle shape, and an S-shape, providing a variety of driving scenarios to capture diverse steering behavior. The tracks were designed to be within a manageable indoor space, allowing for repeated data collection runs under consistent lighting conditions.
With the hardware setup complete and the tracks prepared, the process of data collection commenced. A human operator manually drove the RC car around each of the track's multiple times. During these runs, the Raspberry Pi simultaneously captured images from the Pi Camera and recorded the corresponding steering values and speed. The steering values were obtained from the manual controller CT6B Transmitter inputs, while the speed was also recorded based on the controller input. Data was collected at a consistent sampling rate to ensure temporal coherence between the visual input and the control actions. All collected data, consisting of the images and their associated steering and speed values, was logged into separate CSV files for each run. To ensure consistent results during the data collection process, threading was implemented on the Raspberry Pi. This allowed for the simultaneous capture of images and recording of sensor data without one process blocking or delaying the other, leading to a more synchronized and reliable dataset. Each data collection run involved a specific number of laps or a defined duration on each track, and the data from each run was stored in a distinct CSV file, facilitating organization and later processing.

2) Phase 2: Training Model

The second phase involved training a deep learning model using the data collected in the previous phase. The initial step was to consolidate and prepare the dataset. This involved taking the CSV files generated from each data collection run and organizing the image paths along with their corresponding steering and speed values. Preprocessing steps were then applied to the image data to enhance the training process. This typically included resizing the images to a consistent resolution suitable for the neural network input and normalizing the pixel values to a specific range (e.g., 0 to 1) to improve training stability and performance. To further enhance the model's ability to generalize to unseen situations and prevent overfitting to the training data, data augmentation techniques were employed. These techniques involved applying various transformations to the training images, such as horizontal flipping, slight rotations, and adjustments to brightness and contrast.

These augmentations artificially increased the size and diversity of the training dataset, making the model more robust.

The core of the training phase was the development and training of a Convolutional Neural Network (CNN) sequential model. The architecture of this model typically consisted of several convolutional layers responsible for extracting spatial features from the input images, followed by pooling layers to reduce dimensionality and increase robustness to small shifts and distortions. The extracted features were then fed into one or more fully connected layers to learn the mapping between the visual features and the steering output. Activation functions, such as ReLU (Rectified Linear Unit), were applied after each convolutional and fully connected layer to introduce non-linearity into the model, enabling it to learn complex relationships. The specific number of layers, the size of the convolutional filters, the number of neurons in the fully connected layers, and other architectural parameters were determined based on experimentation and common practices in the field.

The training process was performed on a laptop equipped with sufficient computational resources (e.g., a dedicated GPU) to accelerate the training of the deep learning model. Software frameworks like TensorFlow or Keras, which provide high-level APIs for building and training neural networks, were utilized. An optimization algorithm, such as Adam or Stochastic Gradient Descent (SGD), was selected to update the model's weights during training, aiming to minimize a chosen loss function. The loss function, typically Mean Squared Error (MSE), quantified the difference between the model's predicted steering values and the actual steering values from the training data. The training data was divided into batches, training data and validation data the model's performance on this validation set was monitored during training to detect and prevent overfitting. Once the training process was complete and a satisfactory level of performance was achieved, the trained model, containing the learned weights and biases, was saved to a file for later deployment. Finally, the training progress was visualized by plotting the training and validation loss and accuracy over the epochs. These plots provided insights into the model's learning behavior and helped in assessing its overall performance.

3) Phase 3: Implementation

The final phase of the NavDrive project involved implementing the trained model on the Raspberry Pi and testing the autonomous driving capabilities of the system. The first step was to deploy the saved trained model onto the Raspberry Pi. This involved ensuring that the necessary software libraries (e.g., TensorFlow Lite or a similar optimized framework for embedded systems) were installed on the Raspberry Pi to execute the model efficiently. OpenCV was then utilized to handle the real-time image acquisition from the Pi Camera. The software on the Raspberry Pi was designed to continuously capture frames from the camera, preprocess these frames in the same manner as the training data (e.g., resizing, normalization), and then feed the processed image to the loaded trained model. The model, in turn, predicted a steering value based on the visual input. This predicted steering value was then transmitted from the Raspberry Pi to the Arduino via serial communication over the USB cable. The Arduino, upon receiving the steering command, used this value to control the angle of the servo motor, effectively steering the RC car. The control of the motor driver and the DC motors was also managed by the Arduino. In this implementation, the speed of the motors might have been set to a fixed value for simplicity, or in a more advanced implementation, the model could potentially also predict a speed value to be sent to the Arduino. The entire process, from image capture to motor control, was designed to run in real-time, allowing the NavDrive system to navigate the tracks autonomously based on the learned driving behavior.

IV. DISCUSSION

The results obtained from the NavDrive project indicate the feasibility of using a Raspberry Pi, Arduino, and behavior cloning with a CNN to create a functional, albeit basic, autonomous vehicle. The decreasing training and validation loss, along with the increasing accuracy (where applicable), suggest that the CNN model successfully learned to mimic the steering behavior demonstrated during the data collection phase. The qualitative assessment of the autonomous driving performance on the oval and circular tracks further supports this conclusion. However, the challenges encountered on the S-shaped track highlight the limitations of the current implementation and the complexity of more intricate driving scenarios.

The behavior cloning approach proved to be effective in enabling the NavDrive system to perform basic autonomous navigation. Its simplicity allowed for a relatively straightforward implementation of the data collection, model training, and deployment pipeline. The CNN model, trained through behavior cloning, successfully learned to associate visual cues from the track with appropriate steering actions. However, the observed deviations on the S-shaped track suggest that the model's ability to generalize to more complex trajectories might be limited by the diversity and scope of the training data. These findings align with existing literature on behavior cloning, which acknowledges its effectiveness for specific tasks but also points out its potential limitations in handling unseen scenarios or recovering from errors.[13]Several challenges were encountered during the development of the NavDrive system.

Ensuring consistent and synchronized data acquisition during the data collection phase required the implementation of threading on the Raspberry Pi. Initial attempts at training the CNN model revealed the need for data augmentation techniques to improve the model's robustness and prevent overfitting.

Deploying the trained model onto the resource-constrained Raspberry Pi required careful consideration of model size and computational efficiency, potentially necessitating the use of model optimization techniques or a lighter version of the TensorFlow framework. Overcoming these challenges involved a process of experimentation, debugging, and referring to best practices in the fields of robotics and machine learning.

V. CONCLUSION AND FUTURE WORK

A. *Conclusion*

The NavDrive project successfully demonstrated the development of a next-generation automated vehicle driving system utilizing a Raspberry Pi, Arduino, and real-time intelligent vision through the application of behavioral cloning and a Convolutional Neural Network. The three-phase approach, encompassing data collection, model training, and implementation, resulted in a low-cost prototype capable of autonomous navigation on custom-designed indoor tracks. The project highlights the potential of readily available and affordable hardware for exploring complex concepts in autonomous robotics and provides valuable insights into the practical aspects of implementing behavioral cloning for autonomous driving.

B. *Future Work*

Building upon the success of the NavDrive project, several avenues for future research and improvement can be explored. One direction involves investigating different CNN architectures, potentially incorporating recurrent layers like LSTMs to capture temporal dependencies in the driving data.[15] Exploring reinforcement learning techniques could also lead to more adaptive and robust autonomous driving capabilities.[9] Integrating additional sensors, such as ultrasonic sensors for obstacle detection and avoidance [11], or more advanced sensors like LiDAR, could enhance the system's perception of its environment.[1] Testing the system in more complex and dynamic environments, including outdoor scenarios and varying lighting conditions, would provide a more rigorous evaluation of its performance. Expanding the training dataset with more diverse driving scenarios and track configurations could improve the model's generalization capabilities. Incorporating a manual override mechanism would enhance the safety of the system. Furthermore, investigating methods for improving the interpretability of the learned driving behavior could provide valuable insights into the model's decision-making process.[7] Finally, exploring the use of simulation environments for data generation and model training could offer a cost-effective and scalable approach for future development.[9]

REFERENCES

[1] Abdhalin, abdul Hafiz &Khairunizam, Wan & Ali, Hasimah. (2021). Autonomous Vehicle: Introduction and Key-elements. Journal of Physics: Conference Series. 1997. 012016. 10.1088/1742-6596/1997/1/012016.https://www.researchgate.net/publication/354186664_Autonomous_Vehicle_Introduction_and_Key-elements

[2] Pandey, Navneet & Harsh, & Jayant, Himanshu & Kumar, Narendra & Kumar, Vinod. (2022). Behavioural Cloning in Autonomous Vehicle. 10.3233/ATDE220771.https://www.researchgate.net/publication/365341113_Behavioural_Cloning_in_Autonomous_Vehicle

[3] Othman K. Exploring the implications of autonomous vehicles: a comprehensive review. Innov. Infrastruct. Solut. 2022;7(2):165. doi: 10.1007/s41062-022-00763-6. Epub 2022 Mar 1. PMCID: PMC8885781.https://pmc.ncbi.nlm.nih.gov/articles/PMC8885781/

[4] Bimbraw, Keshav. (2015). Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology.ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings. 1. 191-198. 10.5220/0005540501910198. https://www.researchgate.net/publication/283757446_Autonomous_Cars_Past_Present_and_Future_-_A_Review_of_the_Developments_in_the_Last_Century_the_Present_Scenario_and_the_Expected_Future_of_Autonomous_Vehicle_Technology

[5] Morga-Bonilla SI, Rivas-Cambero I, Torres-Jiménez J, Téllez-Cuevas P, Núñez-Cruz RS, Perez-Arista OV. Behavioral Cloning Strategies in Steering Angle Prediction: Applications in Mobile Robotics and Autonomous Driving. World Electric Vehicle Journal. 2024; 15(11):486. https://doi.org/10.3390/wevj15110486

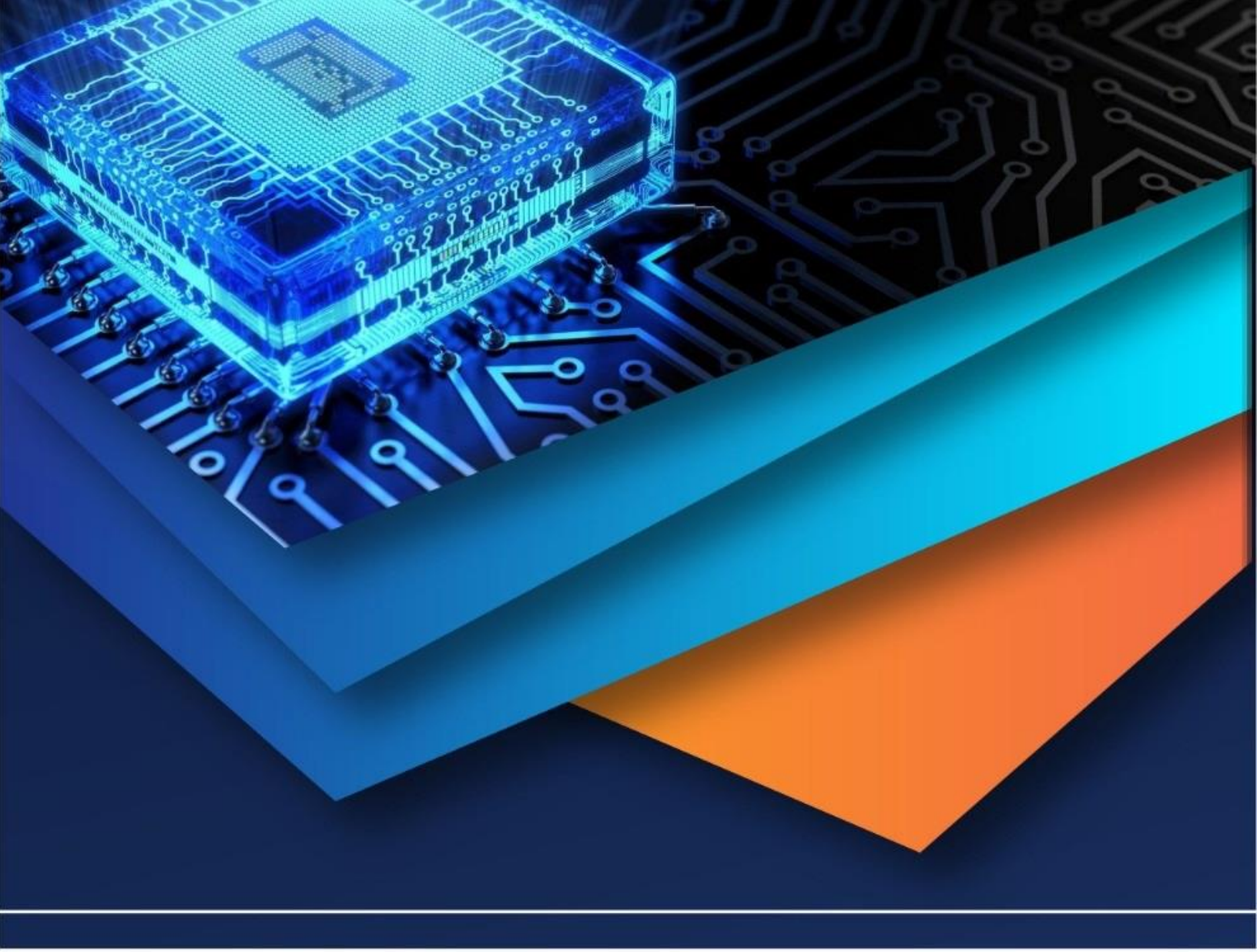
[6] Shahane, Vikrant & Jadhav, Hrushikesh &Sansare, Mihir &Gunjgur, Prathmesh. (2022). A Self-Driving Car Platform Using Raspberry Pi and Arduino.1-6. 10.1109/ICCUBEA54992.2022.10010814. https://www.researchgate.net/publication/367196472_A_Self-Driving_Car_Platform_Using_Raspberry_Pi_and_Arduino

[7] A, Prof & Salim, Aiman & Dileep, Arya & S, Anjana. (2020). Autonomous Car using Raspberry PI and Ml.International Journal of Recent Technology and Engineering (IJRTE). 9. 1067-1071. 10.35940/ijrte.B4033.079220. https://www.researchgate.net/publication/363668338_Autonomous_Car_using_Raspberry_PI_and_Ml

[8] M. A. Hossain, S. Hossain, and M. A. Rahman, “Implications of autonomous vehicles on different sectors: A review,” J. Adv. Transp., vol. 2022, Art. no. 8885781, 2022, doi: 10.1155/2022/8885781. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC8885781/

[9] M. A. P. Mahmud, M. S. Hossain, and M. A. Rahman, "Autonomous Vehicle: Introduction and Key-elements," SSRN Electron. J., 2021, doi: 10.2139/ssrn.3918854. [Online]. Available: https://www.researchgate.net/publication/354186664_Autonomous_Vehicle_Introduction_and_Key-elements

[10] F. Karjanto and A. Yesufu, "Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology," SSRN Electron. J., 2015, doi: 10.2139/ssrn.2698833. [Online]. Available:((https://www.researchgate.net/publication/283757446_Autonomous_Cars_Past_Present_and_Future_-_A_Review_of_the_Developments_in_the_Last_Century_the_Present_Scenario_and_the_Expected_Future_of_Autonomous_Vehicle_Technology.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY