PROJECT PHASE II REPORT

on

VISUAL SERVOING BASED ROBOT CONTROL

Submitted by

Advaith S (20319006)
Tom V Babu(20319094)
Elju Gigi(20319099)
Sreelakshmi pradeep(20319088)
Anand Sivadas M (20319015)

in partial fulfilment of requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION

DIVISION OF ELECTRONICS ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI-682022

DECEMBER 2022

DIVISION OF ELECTRONICS ENGINEERING

SCHOOL OF ENGINEERING

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI-682022

CERTIFICATE

Certified that the Project Phase I report entitled ***"VISUAL SERVOING BASED ROBOT CONTROL"*** *is a bonafide work of* ***ADVAITH S, ANAND SIVADAS M , ELJU GIGI, SREELAKSHMI PRADEEP, TOM V BABU*** *towards the partial fulfilment for the award of the degree of B.Tech in Electronics and Communication of Cochin University of Science and Technology, Kochi-682022.*

Project Guide

Dr.Shahana T.K

Head of the Division

Dr. Anju Pradeep

ACKNOWLEDGEMENT

We would like to express our thanks to a whole host of people who supported us in this endeavor of ours. First we would like to thank each and every member of our respective families for their love and support.

We would like to thank Dr.Anju Pradeep, Head of the Division of Electronics and Engineering. Also, we would like to express our gratitude towards Dr. Shahana T.K for guiding us through this project and also towards all the teachers who wholeheartedly supported us along with the Lab staff and EC faculty.

We would like to thank all of our friends who were with us from the beginning till the end.

ABSTRACT

The aim is to create a system of autonomously navigating vehicles by using feedback from a camera that monitors the movement and position of the vehicles. The system will be contained within a predefined area where the camera will be fixed from the top thus giving it full visual range to monitor the vehicles. The visual from the camera will be used to identify our vehicles and control them. Usually, an autonomous vehicle carries proximity sensors to detect and avoid obstacles or other vehicles, but in the system of our design such sensors are avoided instead the central camera is the only element we use to monitor and control the system.

Python language was used as our primary programming language along with C++. There are well documented modules such as OpenCV for the visual servoing aspects of the program on top of that we will make use of multiple modules available for robotic control. At the end of this project we hope to have a system of vehicles controlled solely through visual input.

CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Problem statement

To control and coordinate the working of multiple robots without any sensors placed within them. This will eliminate the use of sensors within the bots that they normally use for navigation to both reduce the size of the robot and to reduce the cost of building a bot. Instead a visual monitoring system will be employed and then coordinating the entire working of the robots through a central computing system.

1.2 Background

Visual servoing is a well-known approach to guide robots using visual information. Image processing, robotics and control theory are combined in order to control the motion of a robot depending on the visual information extracted from the images captured by one or several cameras. With respect to vision issues, different problems are currently under research such as the use of different kinds of image features (or different kinds of cameras), image processing at high velocity, convergence properties, etc. Furthermore, the use of new control schemes allows the system to behave more robustly, efficiently, or compliantly with less delays. Multiple robots could be used with the same visual servoing technology when the other sensors of the robots turn obsolete creating a coordination between them.[1]

In so-called image-based visual servoing systems, the control law is calculated using directly visual information. These last systems do not need a complete 3D reconstruction of the environment. For tasks that require high precision, speed or response, several works suggest that it may be beneficial to take into account the dynamics of the robot when designing visual servoing control laws. The type of visual control systems that consider the dynamics of the robot in the control law are often referred to as direct or dynamic visual control systems. However, for simplicity, indirect visual servoing schemes are mostly used in the literature.Nowadays, the application fields of the visual servoing systems are very wide, and include research and application fields such as navigation and localization of mobile robots, guidance of humanoid robots, robust and optimal control of robots, manipulation, intelligent transportation, deep learning and machine learning in visual servoing and visual guidance of field robotics (aerial robots, assistive Robots, medical robots, etc.)

1.3 Objective

The objective of Visual Servoing-based robot control is to develop a system that enables a group of robots to be controlled solely through the use of a camera and computer system, without the need for any additional sensors on the robots. The ultimate goal is to achieve efficient and precise control of the robots through the Visual Servoing system, with the potential to enhance their performance in a variety of applications.

1.4 Scope

The scope of the "Visual Servoing of Robots" project includes warehouse automation, centralized control over a group of automobiles, and any other scenarios where adding individual sensors to robots may be cost-prohibitive. The project aims to develop a prototype system that can accurately guide and control the robots, improving their overall efficiency and productivity. The key aspect of the project is to explore different ways of integrating an external sensor with the robots, enabling them to perform various tasks autonomously. The project will involve designing, implementing, and testing the system to ensure its practicality and effectiveness in real-world scenarios.

1.5 Methodology

Design:

The first step in the methodology is to design the work area of bots.The design process includes designing the structure of area which is 5*3 matrix,designing of bot,specifications of bot and battery design.

Development:

The second step is to develop machine learning,by using machine learning object detection model also developed.The obstacles and path is separated using object detection model.After this path-finding is developed where path and bot movement integrated together.

Integration and Working:

The fourth step is Integration and Working. This involves conducting functional, integration, and performance testing of the platform to ensure that it meets the requirements and specifications.As a part of testing bot and object detection are integrated together and made work.The bugs formed and problems faced in path finding are solved.

1.6 Limitations

- The system is trained on certain obstacles and it has a limited computational capability so it won't be able to compute dynamically moving obstacles. It can only identify stationary and symmetric obstacles.
- The training data used to identify obstacles is actually predefined for execution in a controlled environment. This training data cannot be used to implement in a real-time scenario.
- Concurrent processing could not be implemented so that multiple bots could be controlled at the same time. The implementation of multiple robot control would allow the possibility of a complex robotic workflow.
- The area of visibility of the camera which is part of the Visual servoing system is quite limited which limits the number of bots that can be controlled and the area that is visible.
- Inconsistent detection of the robot and the obstacles in the system because of the lack of a robust machine learning model which in turn lead to the buffering of the system.

CHAPTER 2

SYSTEM DESIGN

2.1 Block diagram

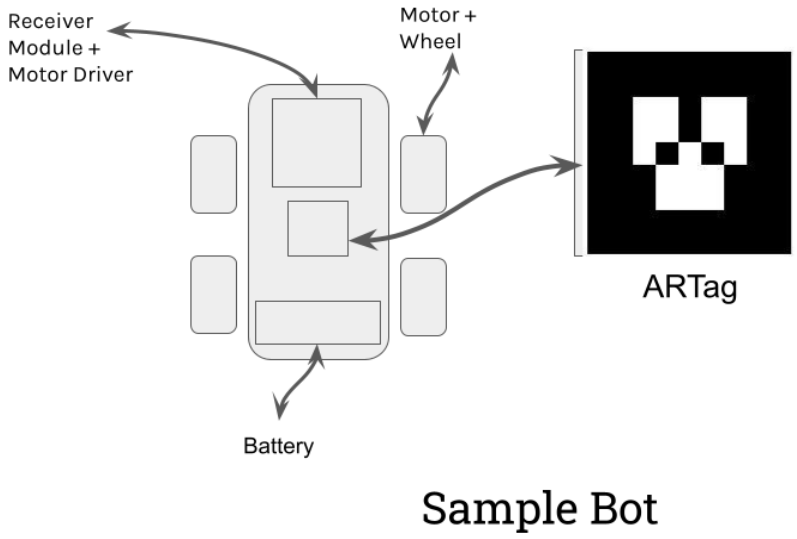

Figure 1 Block diagram of sample bot

The sample bot contains a receiver which will be the ESP-01 that will take the information regarding the navigation that will be received from the NodeMCU placed at the CSS side. The bot will contain two motors and the corresponding wheels along with a castor wheel placed at the front end inorder to balance the bot. The bot will also be separately identified with the help of an ARTag. AR Tag which is an augmented reality tag will provide each of our bots with a unique id which we can use to uniquely identify each of the bot. Such unique identification is necessary in the tracking phase which will help the individual control the bot. Since collision detection between bots should be optimised the AR tag will help in the optimisation. A lithium ion battery will be provided to power the motors, motor driver, the ESP-01 unit. The advantage of the lithium ion battery is that they have the necessary power and they can also be recharged.

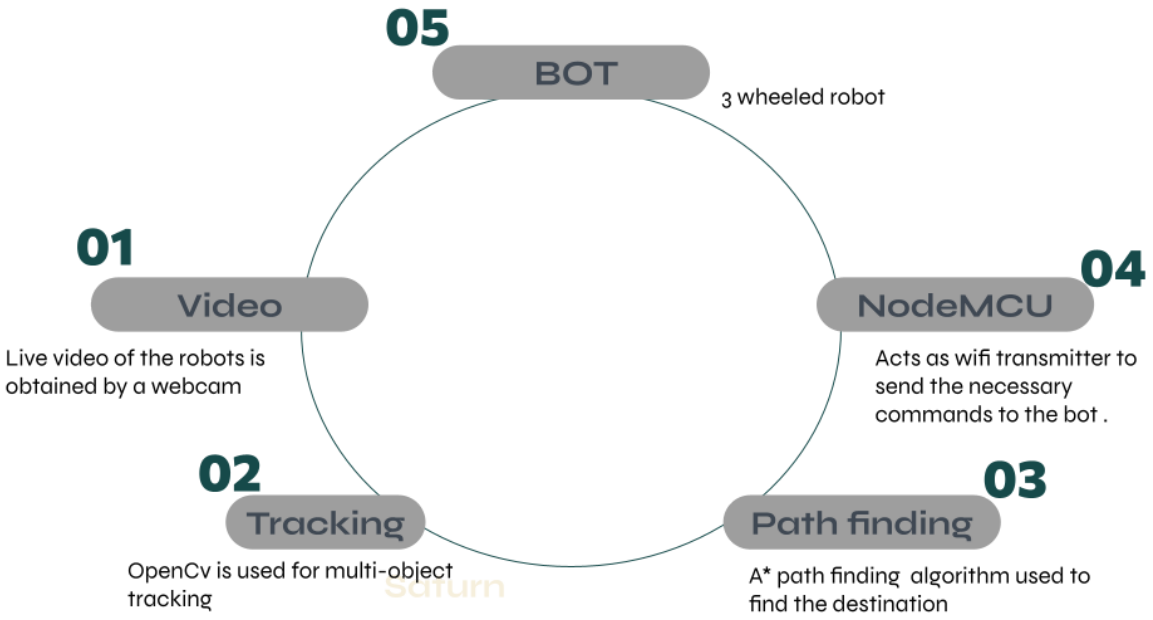

Figure 2 Feedback cycle of system

The input will be taken from the webcam and then forward that central servoing system. After converting it to the required frame rate , an object tracking algorithm will be applied on each frame. After identifying individual bots, bounding rectangles are drawn. Bounding rectangles enable detection whether two or more bots will collide or not. After selecting the start and destination point of the bot we will apply the A* algorithm to find the path and avoid the obstacles. This information will be passed onto the bot with the help of a NodeMCU WiFi module and then the movement of the bot will then be taken in through the video and thus fed back into the rest of the system.

2.2 Hardware components

2.2.1 Camera

A wide angle webcam with a field of view will be used so that the entire area of interest is under its view. A Lenovo 500 webcam is suggested.

2.2.2 Battery

A lithium-ion battery is a type of rechargeable battery that is charged and discharged by lithium ions moving between the negative (anode) and positive (cathode) electrodes.Because lithium-ion batteries are suitable for storing high-capacity power,they are used in a wide range of applications,including consumer electronics such as smartphones and PCs,industrial robots,production equipment and automobiles.

- Voltage:3.7 Volts
- Capacity: 1200 mAh
- Rechargeable: Yes
- Battery Size: Diameter- 18mm x Length-67mm
- Charging Method CC-CV

Figure 3. lithium-ion battery[2]

2.2.3 NodeMCU esp8266

The NodeMCU ESP8266 development board comes with the ESP-12E module containing the ESP8266 chip having Tensilica Xtensa 32-bit LX106 RISC microprocessor. This

microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.

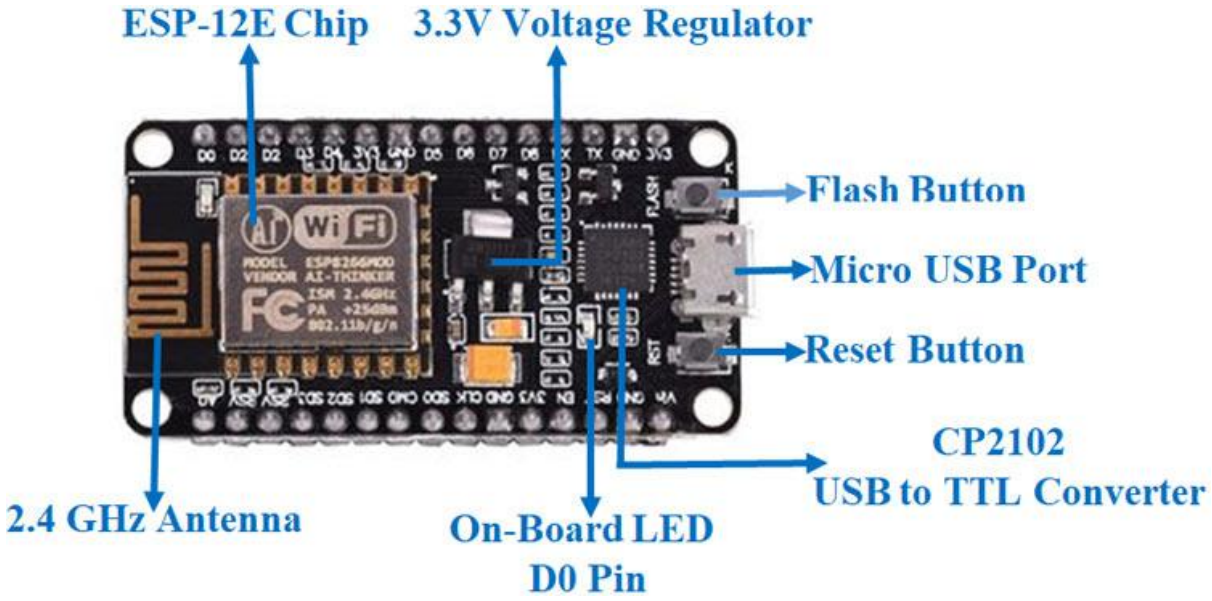

Figure 4. NodeMCU esp8266 [3]

2.2.4 Motor Driver

L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

The L298N Motor Driver module consists of an L298 Motor Driver IC, 78M05 Voltage Regulator, resistors, capacitor, Power LED, 5V jumper in an integrated circuit. 78M05 Voltage regulator will be enabled only when the jumper is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator and the 5V pin can be used as an output pin to power the microcontroller. The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through a 5V terminal to power the internal circuitry.

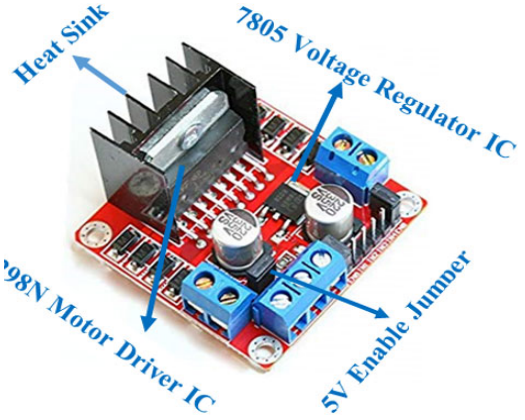

Figure 5. L298N [5]

Features & Specifications

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Motor Supply Voltage (Maximum): 46V
- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Driver Current:2A
- Logical Current:0-36mA
- Maximum Power (W): 25W
- Current Sense for each motor
- Heatsink for better performance
- Power-On LED indicator

2.3 Total Cost

Table 1. Components and cost

Components	**Nos**	**Cost**
Camera	1	1500
Motor	6	660
Wheel	6	456
Li-ion battery	6	600
chassis	3	540
L298N 2A Motor driver	3	150
NodeMCU esp8266	3	822
Total cost		4,728

CHAPTER 3
IMPLEMENTATION

3.1 OVERVIEW OF THE SYSTEM

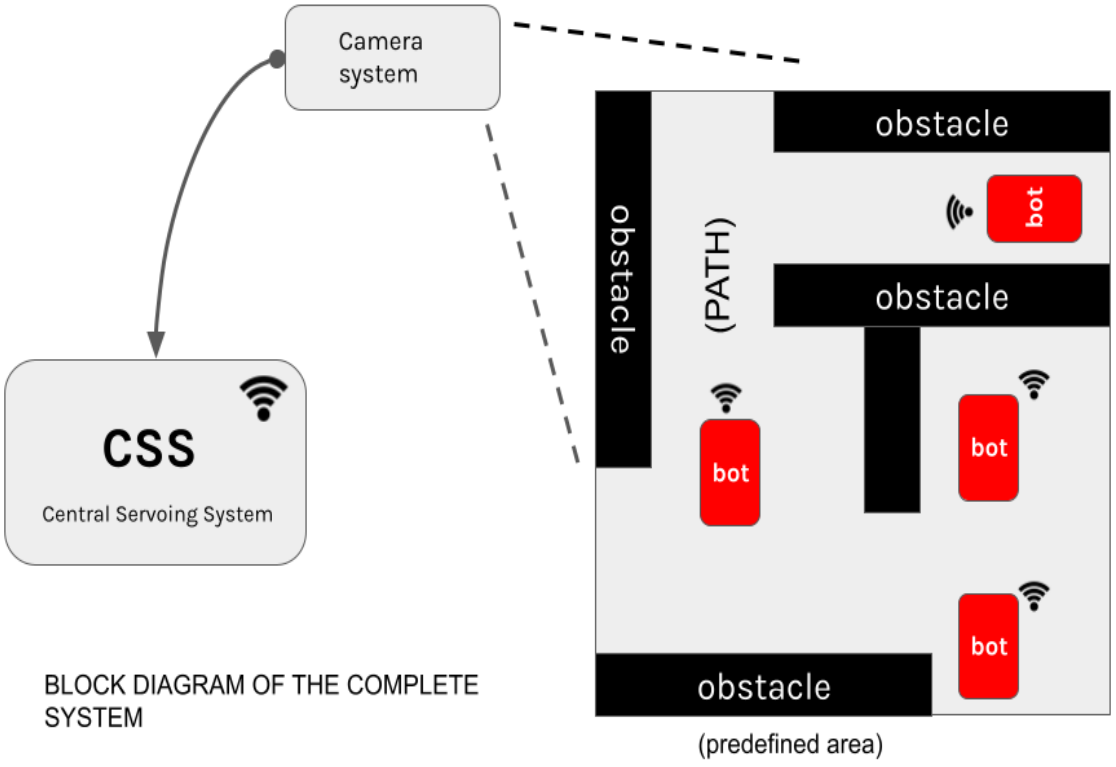

Figure 6. Block diagram of system

The block diagram defines the overall system that will be executed in this project. We will demarcate a predefined area and set up obstacles within it. There will be 3 bots that will be controlled by the CSS. The video input will be obtained from the top view by setting up the camera from the top. The camera chosen will have a wide angle field of view so that the entirety of the demarcated area can be captured. The input from the camera will be fed into the CSS (The Central Servoing System). The CSS is a laptop system with GPU capability inorder to process the input image so that we can detect the obstacles and then track the bots by drawing bounding boxes over them. These detected objects will then be employed into the A* path finding algorithm which will be used to navigate through the obstacles and to avoid the other bots. It is from the CSS that we will get the complete data regarding the system that

we will then use for navigation and coordination of the bots. The information regarding the navigation of the bots will be communicated to the bots through the nodeMCU wifi module.

3.2 Subsystems

The entire system is divided into multiple sub-systems and integrated together. We will further describe each subsystem in the following pages.

3.2.1 Servoing System

The servoing system consists of a wide angle camera that will be set up to obtain the top view of our region of interest with the help of a tripod arrangement. The video input will then be fed into our laptop system which has a built-in GeForce GTX 1650 GPU that is capable of processing the entire input. The processing consists of the object detection and tracking of the bots and obstacles. The system will also process the path finding of the bots within the area and then communicate the path of navigation to the bots with the help of a wifi module.

3.2.2 Object Detection & Tracking

Training The Model

Images of our bots will be obtained which have to be trained for the project and then use the image labeling software called labelImg. The labelImg will produce an xml file which will contain the labels and the associated position in terms of the pixel location of the image objects. Once labeled data has been acquired it will be split into training and testing data. A pre-trained model's pipeline config path will be used and modified this configuration file to suit the training of our projects.The training set will be used to train a new object detection model that will contain the necessary weights and the config path which we will use in the object tracking. The model is trained with the help of yolov5 object detection algorithm and the pre-trained model used in this situation is the yolo v5l weights. Pre-trained weights are used since it is not possible to create a model from the bottom up completely, all the lines and curves and other fundamental parameters. Once the model was trained with this pre-trained weight as the basis the resultant.

Object Tracking Using The Model

Once the weight is ready they will be used to build an object detection program and from the configuration file and use the weights that have been built with the help of the model. Once

initiated the object detection function then will use the positions which will be the output of the object detection function. Using these positions the bounding boxes will be built. Thus in each frame the objects will be detected and we will get the bounding box and in order to connect the different bounding boxes. The center of the bounding box will be found and then this will identify each new position to which the bounding box moves by identifying the new pixel position of the center of the bounding box. Once the information regarding the center of the bounding box is known, each object can be identified with a different id and with this id we can keep track of each new object.

Building the matrix

The center of each bounding box is taken and then from the length and breadth of the entire area the top right center grip point is obtained. From this the rest of the matrix is build which again is dynamically made depending on the length and breadth of the entire area. Once the grid which in the case of this project is 3x5, the ML model will detect the obstacles and the center of the obstacles will be identified. Then the centers will be listed in a list which will be compared with the list of the grid centers. Once the comparison is done then the list with just the paths will be finalized.

3.2.3 Communication

Communication of the Bots with the Central servoing system is necessary to control and coordinate them to solve their objective and avoid collision between them. After the commands are generated by the CSS the next step is to successfully transmit data onto each of the bots.

CSS and Bot Communication

Hardware setup :

CSS is connected to Esp8266 Module. Each of the bot has its own ESP8266 Module.The Esp8266 connected to CSS is called Master Node.This Master Node act as the transmitter of data from CSS.Each of the bots have Esp8266 Module on it and they are called Slave Node.These act as receiver of commands for each bot.

ESP-NOW :

Parameters to consider in communication for visual servoing are :

a. Latency and Data transmission rate
b. Reliability
c. Concurrent data delivery
d. Compatibility
e. Power consumption
f. Security

ESP-NOW is a wireless communication protocol developed by Espressif, the manufacturer of the ESP8266 and ESP32 microcontrollers. It allows two or more ESP devices to communicate directly with each other without the need for a Wi-Fi network or router. ESP-NOW is designed for low-latency and low-power communication between ESP devices. It can send and receive data packets of up to 250 bytes and provides reliable and secure communication between ESP devices. ESP-NOW uses a pre-shared key to authenticate devices and encrypt data packets.

To use ESP-NOW, the ESP devices , the Master Node and other Slave Nodes must be configured with the same channel and pre-shared key. After configuration, the ESP devices communication will be unidirectional. There will be no bot to bot communication but only Master Node to Slave Node communication.

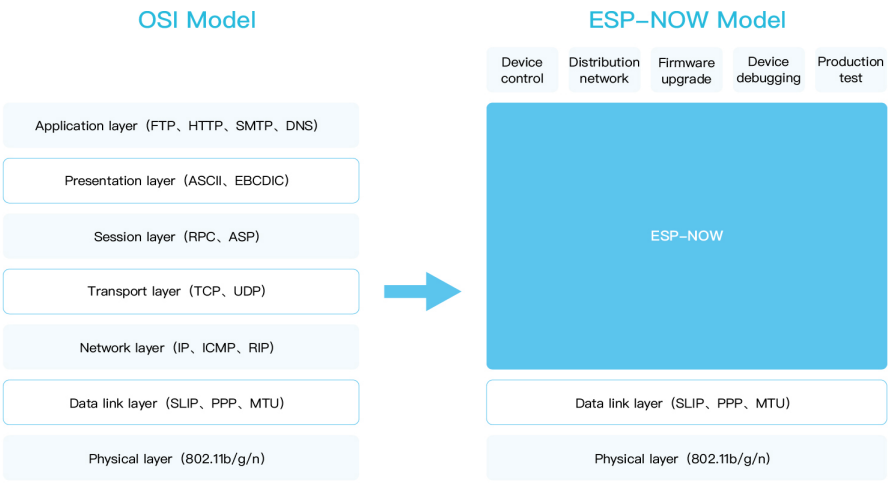

Figure 7.Comparison of OSI Model and ESP-NOW Model

ESP-NOW Frame format

Different from traditional Wi-Fi protocols, the first five upper layers in OSI are simplified to

one layer in ESP-NOW, so the data doesn't need to be transmitted through the network layer, the transport layer, the session layer, the presentation layer, and the application layer, which reduces the delay caused by packet loss under congested network, and leads to quickly response time.

The data transmission mode of ESP-NOW is flexible including unicast and broadcast, and supports one-to-many device connection and control.ESP-NOW uses a vendor-specific action frame to transmit ESP-NOW data. The default ESP-NOW bit rate is 1 Mbps. The format of the vendor-specific action frame is as follows:

MAC Header	Category Code	Organization Identifier	Random Values	Vendor Specific Content	FCS
24 bytes	1 byte	3 bytes	4 bytes	7~255 bytes	4 byte

Category Code: The Category Code field is set to the value(127) indicating the vendor-specific category.

Organization Identifier: The Organization Identifier contains a unique identifier (0x18fe34), which is the first three bytes of MAC address applied by Espressif.

Random Value: The Random Value field is used to prevent relay attacks.

Vendor Specific Content: The Vendor Specific Content contains vendor-specific fields as follows:

Element ID	Length	Organization Identifier	Type	Version	Body
1 byte	1 byte	3 bytes	1 byte	1 byte	0~250 bytes

Element ID: The Element ID field is set to the value (221), indicating the vendor-specific element.

Length: The length is the total length of Organization Identifier, Type, Version and Body.

Organization Identifier: The Organization Identifier contains a unique identifier(0x18fe34), which is the first three bytes of MAC address applied by Espressif.

Type: The Type field is set to the value (4) indicating ESP-NOW.

Version: The Version field is set to the version of ESP-NOW.

Body: The Body contains the ESP-NOW data.

The MAC header is a little different from that of standard frames. The FromDS and ToDS bits of the FrameControl field are both 0. The first address field is set to the destination address. The second address field is set to the source address. The third address field is set to broadcast address (0xff:0xff:0xff:0xff:0xff:0xff).

Master Node - Slave Node Data transfer

The Master Node sends data in : " <1 1 100 >" where first number represent the ID no. of the Slave Node the data is send to and second number (0 : stop ; 1 : forward ; 2 : backward ; 3 : left ; 4 : right) represent the direction of Slave bot and finally third number represent duration in that direction. Once data is received successfully by the Slave node it returns an acknowledged signal 0 to Master Node.

3.2.4 Path finding

The A* algorithm, also known as A-star algorithm, is a widely used search algorithm in computer science and artificial intelligence that is commonly used for finding the shortest path or optimal path between two points in a graph or a grid. The algorithm is particularly well-suited for pathfinding problems in which the graph or grid has a large number of nodes or vertices, making it computationally expensive to search through all possible paths.

The A* algorithm is a combination of two other popular algorithms, Dijkstra's algorithm and a heuristic search algorithm. It uses a heuristic function to estimate the cost or distance from the current node to the goal node, and combines this heuristic estimate with the actual cost of the path from the starting node to the current node. This combined cost is used to prioritise the order in which nodes are explored during the search, with the goal of finding the optimal path more efficiently.

Cost of each cell or grid ,

$$f(n) = g(n) + h(n)$$

Where g(n) is calculated using Dijkstra's algorithm, the cost function g(n) is initialized by setting the cost of the starting node to 0, and the cost of all other nodes to infinity. This is typically done using a priority queue or a min-heap to keep track of the nodes to expand in the order of their accumulated costs. The starting node is then set as the current node, and for each neighboring node of the current node, the cost of reaching that neighboring node from

the starting node through the current node is calculated by adding the cost of the edge between them to the cost of the current node. If the calculated cost is less than the current cost of the neighboring node, the cost of the neighboring node is updated with the calculated cost, and the current node is set as the parent of the neighboring node. This step ensures that the algorithm always chooses the path with the minimum cost to reach a node. The current node is then marked as visited and removed from the priority queue or min-heap. The next unvisited node with the minimum accumulated cost is selected as the new current node, and steps 3-5 are repeated until the goal node is reached or all reachable nodes have been visited. Once the goal node is reached, the accumulated cost of the path from the starting node to the goal node is the actual cost of the path, which can be obtained from the cost of the goal node.

Heuristic function ,

$$h(n) = \sqrt{(x1-x2)^2 + (y1-y2)^2}$$

Where (x1,y1) represents the current cell or grid and (x2,y2) represents the cell or grid that is the goal.

The current cell of a robot is determined by identifying the Aruco marker attached to the robot using a camera. When the camera captures an image of the Aruco marker, the center pixel value of the marker, which represents its x and y position in the image, is extracted. This pixel value is then compared against the predefined points of the grid to determine the location of the robot on the grid. This approach allows for accurate localization of the robot within the grid environment based on the visual detection of the Aruco marker. The destination or goal cell on the grid, which the robot needs to navigate to, is specified by the user through an interface that has been created for the project. The user interacts with the interface by manually selecting the desired position on the grid, typically by clicking on a graphical representation of the grid. This input from the user serves as the goal cell or destination for the robot to reach during its navigation tasks. This approach allows for user-defined goal specifications, providing flexibility and adaptability to the robot's navigation objectives based on user input through the interface.

The path finding algorithm starts with the initial node (the starting point) and adds it to the open list. It then iteratively selects the node with the lowest priority from the open list and expands it by examining its neighboring nodes. For each neighboring node, the algorithm calculates the actual cost of reaching that node from the current node (g(n)) and the estimated

cost of reaching the goal from that node (h(n)). The total cost of reaching the neighboring node (g(n) + h(n)) is compared to the current cost of the neighboring node. If the new cost is lower, the algorithm updates the cost of the neighboring node and sets the current node as its parent.The algorithm repeats this process until the goal node is reached or there are no more nodes left in the open set. Once the goal node is reached, the optimal path can be reconstructed by following the parent pointers from the goal node to the starting node.

3.2.5 Interface

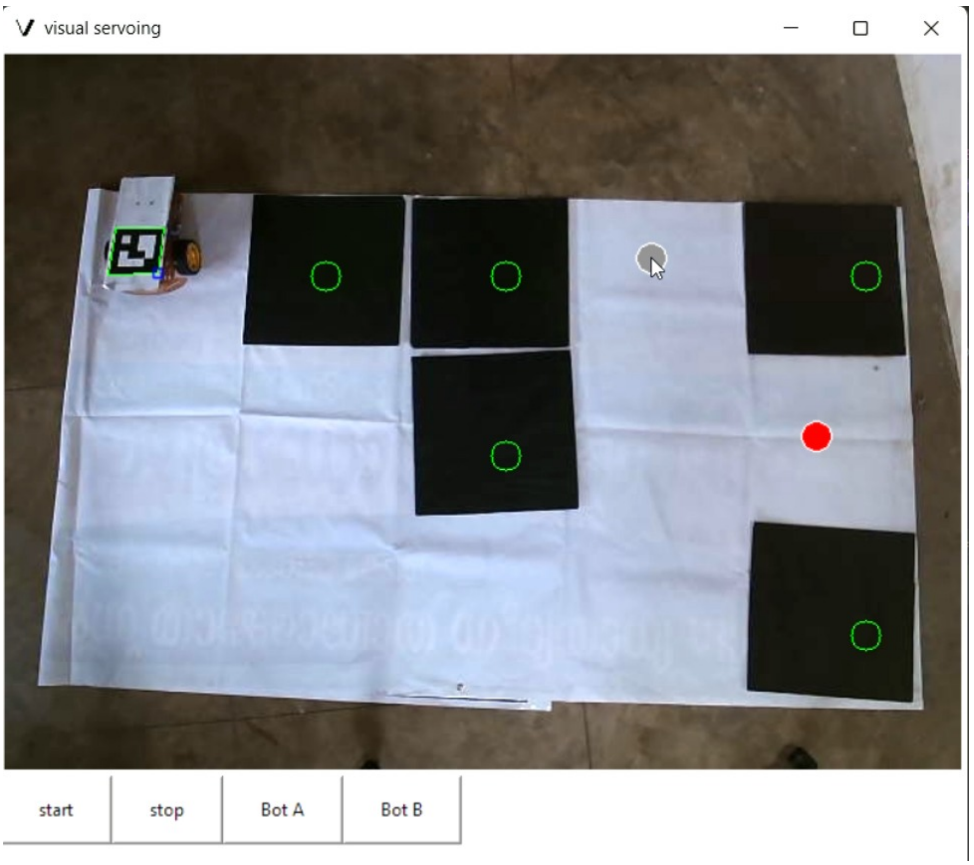

Figure 8.Interface of System

The development of an interface for assigning start and end points to bots in the system is of paramount importance to ensure efficient operation. This interface serves as a user-friendly means for the central control system to initiate the pathfinding algorithm, allowing for seamless integration with the bots' navigation process. Through the use of the Tkinter framework in Python, which is a widely-accepted and standard Python binding to the Tk GUI toolkit, a robust and reliable interface can be built.

The interface is meticulously designed with the end user in mind, aiming to provide a seamless and intuitive experience. The inclusion of a live video streaming viewport allows

the system operator to monitor the movements of the bots in real-time. This real-time feedback enables effective debugging and troubleshooting, as the operator can closely observe the bots' behaviors and make necessary adjustments as needed.

The interface also incorporates buttons for starting and stopping the operation, as well as individual controls for each bot. This allows the system operator to have fine-grained control over the bots, enabling them to initiate or halt operations as required. This level of control provides flexibility and adaptability in managing the bots' movements, making it convenient for the operator to modify the system's behavior on-the-fly.

Furthermore, the interface facilitates the selection of end points for the bots. This dynamic assignment of destinations based on system requirements enables greater flexibility and versatility in the bots' navigation. The system operator can easily specify different end points for each bot, depending on the specific tasks or goals they need to accomplish. This feature allows for efficient customization of the bots' trajectories, making the system adaptable to different scenarios and requirements.

Overall, the carefully designed interface with features such as live video streaming, intuitive controls, and dynamic assignment of start and end points enhances the usability and effectiveness of the system. It provides the system operator with a powerful tool to efficiently manage and monitor the bots' movements.

By utilizing Tkinter as the foundation for the interface, compatibility and ease of use are ensured across different operating systems, including Linux, Microsoft Windows, and macOS. Tkinter is widely recognized as the de facto standard GUI toolkit for Python, making it a suitable choice for academic research and application development. Moreover, the utilization of established best practices in human-computer interaction principles in the interface design further enhances the usability and effectiveness of the system.

Overall, the incorporation of the Tkinter-based interface in the system allows for efficient assignment of start and end points to the bots, facilitating seamless integration with the pathfinding algorithm. The user-friendly and visually appealing design, compatibility across diverse operating systems, and adherence to established best practices in human-computer

interaction make the interface a valuable addition to the system for advanced research and practical applications.

3.3 EXECUTION

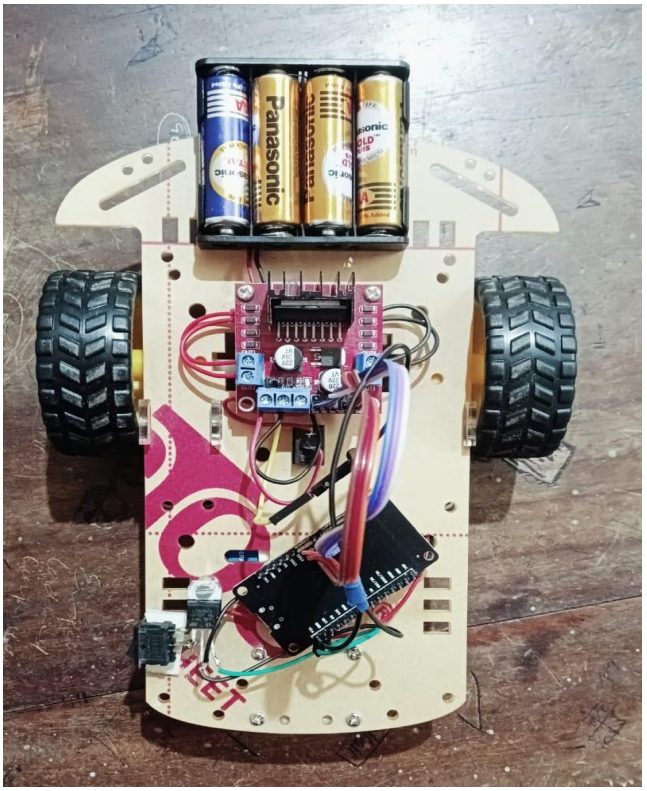

Figure 9. Image of the bot

The central servoing system receives live video feed from the camera, which is then processed to identify the bounding area and detect obstacles using an advanced image classifier. The identified area is converted into a coordinate system in pixels by overlaying a grid, enabling precise localization and mapping of obstacles in the environment. The starting location of the bot is determined using an Aruco marker, and the destination is inputted through a user interface that has been specifically designed for this purpose.

Once the start and end points are established,by using A* pathfinding algorithm, is utilized to find the optimal path for the bot. The algorithm takes into consideration the map with the identified obstacles and computes the most efficient route for the bot to reach its destination. The resulting path is then converted into a series of x, y points corresponding to the center of each cell on the grid, and this information is stored in a list for further processing.

As the bot navigates along the path, it encounters challenges due to variations in the motor speed of the off-the-shelf motors used. To overcome this, a feedback loop is established by drawing three vectors from the bot's current position to the next point it needs to travel.After retrieving the next point from the list created earlier using the pathfinding algorithm, three vectors are drawn from the left, right, and center of the Aruco marker to the immediate next point that the bot needs to travel to. By comparing the location of the bot with these points, the magnitude of each vector is calculated. This magnitude serves as a crucial indicator for steering the bot accurately along the desired path.

To achieve precise control, a decision-making process based on the comparison of the left and right vectors is employed. If the magnitude of the left vector is found to be less than that of the right vector, the bot is turned towards the left until the magnitudes of both vectors become equal and vice versa. This approach ensures that the bot follows a smooth and stable trajectory, while avoiding oscillations that may arise due to rapid transmission of commands.

To further enhance the control strategy, a relaxing value is often employed. Instead of relying solely on the absolute magnitudes of the vectors, a range of values is utilized to account for variations and fluctuations in the bot's motion. This relaxing value helps to mitigate any potential disturbances and uncertainties in the system, thereby improving the overall stability and reliability of the bot's navigation.

In addition to the aforementioned control strategies, the system also employs a sophisticated approach to determine the direction of the bot's movement. Leveraging the inherent characteristics of the Aruco marker, which always provides its left top corner values regardless of its orientation, two vectors are defined. One vector points towards the next immediate point that the bot needs to travel to, while the other vector is drawn to the top edge of the Aruco marker.

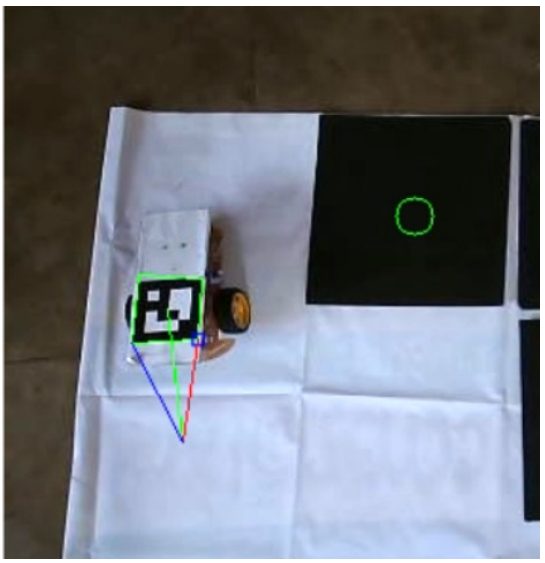

Figure 10.Bot navigating with the help of vectors

To obtain accurate directional information, both vectors are converted into unit vectors, ommiting their magnitudes. The determinant and dot product of these vectors are then calculated, enabling the determination of the cosine and sine of the angle between them. By taking the tangent of the angle, a series of angles in the desired range of 0 to 180 and -180 to 0 are obtained. This process allows for precise determination of the direction of the bot's movement without the need for an IMU sensor or other external references.

Once the crucial control data has been generated, it is meticulously organized into a control data frame and stored in a priority queue, meticulously prioritized based on the system's requirements. Communication between the Central Servoing System (CSS) and the bots is established using the state-of-the-art ESP-NOW protocol, renowned for its low-power and low-latency communication capabilities.

The control data frame, encapsulating vital information for each bot's operation, is broadcasted from the CSS to the bots. To ensure efficient data handling, each bot filters out and processes only the relevant information tailored to its unique requirements. The data for each bot is meticulously packaged onto the frame in a positional manner, with designated bit positions reserved for each bot's data. For instance, the first three bits are allocated for the first bot, followed by the next three bits for the second bot, and so forth, adhering to a systematic approach.

To enable smooth transmission, the data frame is dispatched from the priority queue to the ESP8266 module, meticulously integrated into the system for seamless wireless

communication. Subsequently, each bot receives the transmitted information and interprets it to execute precise movements and actions as dictated by the control data. This meticulous approach to data transmission and bot communication exemplifies the system's advanced technical prowess

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

The aim of this project was to build a multi-robot system which could be controlled through a servoing system, which consisted of a camera and computational system. The idea was to reduce or remove the use of individual sensors in all the bots included and to use a centralized camera to map the entire environment of the bot. The implementation of a single bot system was successful. The multi-bot integration faced a couple of issues specifically on the part where multi-processing had to be implemented for simultaneous movement. In conclusion the navigation and control of a single bot with the help of servoing system was successful.

This system could be implemented in a warehouse thus automating the warehouse in a cost effective manner since we have avoided using sensors on the vehicle.In real world scenario using sensors on each robot car would be costly and consume time to implement on each robot.It is easily avoided by replacing them with our visual servoing. The visual servoing aspect of the system can be improved by incorporating input from different angles as in a CCTV placed along the roadside. This system could be extended to the working of different robotic systems as in a robotic arm.

REFERENCES

[1] [https://www.researchgate.net/publication/220836474_Plan-Based_Configuration_of_a_Group_of_Robots]

[2] [https://quartzcomponents.com/products/18650-li-ion-rechargeable-battery]

[3] [https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet]

[4] [https://jayconsystems.com/blog/getting-started-with-the-esp8266-esp-01]

[5] [https://components101.com/modules/l293n-motor-driver-module]

[6] [https://www.researchgate.net/figure/Voltage-regulator-circuit-LM7805_fig3_355099864]