Object Following Autonomous Bot

Bot Follows the Yellow Ball

Man Singh
Electronics Engineering
Svnit, Surat
Gujarat, India
u22ec130@eced.svnit.ac.in

Smit B Pandya
Electronics Engineering
Svnit, Surat
Gujarat, India
u22ec133@eced.svnit.ac.in

Prince Solanki
Electronics Engineering
Svnit, Surat
Gujarat, India
u22ec139@eced.svnit.ac.in

***Abstract*—This paper presents the development of an autonomous object-following robot designed to track and follow a yellow ball using real-time image processing and wireless motor control. The robot combines the capabilities of digital image processing with Python and the OpenCV library to detect and track the target object based on its color and position within a video feed. The image processing algorithm utilizes HSV color space segmentation to identify the yellow object, and additional techniques such as Gaussian blur, morphological operations, and distance transforms are employed to enhance accuracy and reduce noise in detection.**

The system integrates an ESP32 microcontroller, configured as a web server, to control the robot's movement based on commands received wirelessly from the Python program via HTTP requests. Motor control is achieved using an L239D motor driver, enabling precise directional movement of the robot, including forward, left, right, and slow rotations when the target is not detected. By leveraging wireless communication, the system eliminates the need for wired connections between the processing unit and the robot, resulting in a flexible and scalable design.

Experimental results demonstrate the robot's ability to reliably detect, track, and follow the target object in dynamic environments. This research provides a framework for the development of autonomous robotic systems capable of real-time object tracking and paves the way for future advancements in intelligent robotic applications.

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The development of autonomous robots capable of performing specific tasks has garnered significant attention in recent years due to their applications in various fields, including logistics, surveillance, healthcare, and industrial automation. Among these, object-following robots, which are designed to track and follow moving objects, have emerged as a valuable subset of robotics, offering solutions for tasks like automated delivery, target tracking, and intelligent transportation. This research focuses on the design and implementation of an object-following robot that uses real-time image processing and wireless control mechanisms. The robot tracks a yellow ball as its target object, leveraging the power of digital image processing (DIP) for object detection and decision-making. Python, along with the OpenCV library, serves as the primary platform for implementing the image processing algorithms, enabling the extraction of object features such as color, size, and position.

This work was conducted under the esteemed guidance of Dr. Jignesh N. Sarvaiya, Department of Electronics Engineering, Sardar Vallabhbhai National Institute of Technology, Surat, India. Man Singh , Smit B Pandya and Prince Solanki are with the Department of Electronics Engineering at Sardar Vallabhbhai National Institute of Technology. All authors are affiliated with the same institution.

This research focuses on the design and implementation of an object-following robot that uses real-time image processing and wireless control mechanisms. The robot tracks a yellow ball as its target object, leveraging the power of digital image processing (DIP) for object detection and decision-making. Python, along with the OpenCV library, serves as the primary platform for implementing the image processing algorithms, enabling the extraction of object features such as color, size, and position. These features are processed in real time to determine the target's location within the camera's field of view, guiding the robot's movements accordingly.

The robot's control system is built around an ESP32 microcontroller, which acts as the central node for receiving commands from the Python program and controlling the motors via an L239D motor driver. The ESP32, configured as a web server, facilitates wireless communication with the Python-based control system through HTTP requests. This approach eliminates the need for physical connections between the processing unit and the robot, making the system entirely wireless and more versatile. The adoption of a wireless framework ensures greater scalability and flexibility, allowing the system to operate efficiently in diverse scenarios without being constrained by wired connections.

The image processing workflow is at the heart of this system, employing robust techniques to detect and track the target object under varying environmental conditions. The color segmentation in the HSV (Hue, Saturation, Value) color space isolates the yellow ball from its surroundings, offering better performance than traditional RGB segmentation, especially in dynamic lighting conditions. To improve detection robustness, additional techniques such as Gaussian blur for noise reduction, morphological operations for cleaning the segmented image, and distance transform for refining the object's shape are integrated. These steps ensure that the detection remains reliable, even in the presence of noise, shadows, or overlapping objects.

Once the object is detected, its position relative to the

camera's frame is used to generate movement commands. These commands correspond to directions such as forward, left, or right and are transmitted to the ESP32 for execution. The motor driver (L239D) translates these commands into appropriate voltage signals to control the robot's DC motors, enabling precise and responsive movements. Additionally, a fallback mechanism is implemented, where the robot rotates slowly when the target object is not detected for a prolonged period, ensuring continuous exploration of the environment.

This paper highlights the modular and cost-efficient nature of the system, which relies on readily available hardware components such as a motor driver (L239D), motors, wheels, a Tamya chassis, and a camera. The integration of these components demonstrates how a simple yet effective system can be designed to perform complex tasks like autonomous object tracking and following. Furthermore, the proposed system addresses challenges such as noise interference, real-time responsiveness, and seamless wireless control, making it adaptable to various use cases, from household robotics to industrial applications.

The experimental results validate the effectiveness of the proposed solution, showcasing the robot's ability to reliably detect, track, and follow the target object in dynamic environments. Its ability to adapt to varying lighting conditions, handle noisy backgrounds, and maintain consistent wireless communication underscores its potential for practical deployment. Moreover, the modularity of the system enables future enhancements, such as incorporating multiple object tracking, obstacle avoidance, or advanced control algorithms for smoother navigation.

This research contributes significantly to the field of robotics by presenting a scalable and efficient solution for autonomous object tracking and following. The system's real-time responsiveness, wireless architecture, and cost-effectiveness make it suitable for a wide range of applications, including robotic surveillance, warehouse automation, and intelligent navigation systems. By addressing critical challenges and providing a robust framework, this study lays the groundwork for further advancements in autonomous robotics and their application in modern technological ecosystems.

II. Literature Review

The field of autonomous robots, specifically object-following systems, has seen substantial progress due to advancements in digital image processing (DIP), microcontroller technologies, and wireless communication frameworks. This section reviews existing studies and technologies relevant to object-following robots, highlighting their contributions, limitations, and how the proposed work addresses the identified gaps.

A. Object Tracking and Detection Techniques

Several object-tracking techniques have been developed over the years, focusing on image processing and sensor-based methods. A significant proportion of these studies leverage color-based segmentation, contour detection, and template matching for object detection.

1) *Color-Based Object Tracking:* Researchers have widely adopted the HSV color space for object tracking due to its robustness against lighting variations compared to the RGB color model. For instance, [Author et al., Year] demonstrated how HSV segmentation could improve detection of brightly colored objects under varying illumination. However, noise and overlapping objects remain challenges in these systems.
2) *Deep Learning for Object Detection:* With the advent of convolutional neural networks (CNNs), object detection accuracy has improved significantly. Frameworks like YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector) have been integrated into robotic systems for real-time tracking. However, these methods require substantial computational power, which limits their application in resource-constrained systems like embedded robots.
3) *Hybrid Approaches:* Combining traditional methods with machine learning has shown promise. For example, integrating contour-based detection with a Kalman filter improves tracking in dynamic environments.

B. Wireless Communication in Robotics

The use of wireless technologies for robot control has evolved significantly, with protocols such as Bluetooth, Zigbee, and Wi-Fi enabling remote control and monitoring.

1) *Bluetooth-Based Control:* Bluetooth offers a low-power solution for short-range communication. While suitable for small-scale systems, its limited range restricts applications in dynamic or large environments.
2) *Wi-Fi and HTTP Communication:* Wi-Fi-based solutions, such as the ESP32's web server configuration, have gained popularity for their range and scalability. Studies have shown that HTTP protocols provide a reliable and straightforward mechanism for transmitting control signals, particularly for educational and research projects.

C. Robotic Navigation and Control Systems

Motor control in robotics often involves microcontrollers and motor drivers for seamless actuation. Research into PID controllers and pulse width modulation (PWM) techniques has significantly enhanced robot stability and responsiveness.

1) *L239D Motor Drivers:* The L239D motor driver is commonly used in research for its cost-effectiveness and compatibility with DC motors. While efficient for basic operations, its current limitations make it less ideal for high-torque applications.
2) *Advanced Control Algorithms:* Algorithms like PID and fuzzy logic are increasingly being used for smoother navigation. While not implemented in this study, future work could incorporate such methods for improved precision.

III. Motivation for our Project

The increasing need for automation in robotics has driven innovations in autonomous systems that can perceive and act upon their environment. Object-following robots represent a key area of research due to their potential applications in industries such as logistics, healthcare, surveillance, and personal assistance. The motivation behind our project stems from :

A. Practical Applications:

- In warehouses, object-following robots can transport items without requiring pre-defined paths, improving flexibility.
- In healthcare, they can assist patients by autonomously fetching items.
- In smart homes, they can interactively follow users to provide convenience, such as carrying groceries or assisting with tasks.

B. Affordability and Accessibility:

Many existing solutions in robotics are prohibitively expensive due to their reliance on advanced sensors like LiDAR, proprietary algorithms, and high-powered processors. Our project aims to provide a cost-effective alternative by utilizing readily available hardware (ESP32, USB camera, and L239D motor driver) and open-source software technologies (Python, OpenCV, Arduino).

C. Scalability and Versatility:

By incorporating modular design principles, our system can be adapted for various tasks, such as multi-object detection, obstacle avoidance, and path planning, simply by adjusting the software.

D. Educational Impact:

The project is designed to serve as an educational tool for students and researchers to learn about integrating computer vision, IoT, and robotics at a low cost.

IV. Why Our Project is Better Than Others

A. Cost Efficiency Without Compromising Performance:

- Unlike many high-end robots that use expensive sensors like LiDAR, our system relies on a USB camera and computer vision algorithms to achieve precise object tracking.
- The use of an ESP32 microcontroller reduces costs while offering sufficient processing power and connectivity for real-time applications.

B. Open-Source and Customizable:

- Our software stack is based on widely used open-source libraries, including OpenCV and NumPy, ensuring accessibility for further development and research.
- The code can be easily modified to track objects of different colors, shapes, or sizes, making the system adaptable for diverse use cases.

C. Webserver Integration for Remote Control:

- The integration of a web server on the ESP32 allows the robot to be controlled and monitored remotely over Wi-Fi. This feature is often absent in similar low-cost systems.
- Real-time command execution ensures that the robot responds immediately to environmental changes, making it ideal for dynamic scenarios.

D. Efficient Communication:

- The project demonstrates seamless communication between the Python script and ESP32 through HTTP requests, achieving high reliability and ease of implementation.
- Unlike traditional serial communication, which is limited to short-range use, the Wi-Fi-based communication ensures scalability and remote operability.

E. Enhanced Image Processing:

- Our use of advanced image processing techniques, such as Gaussian blur, edge detection, and morphological operations, enhances accuracy in object detection even under challenging lighting conditions.
- The additional visualization features (distance transform, edge map) provide insights into the detection process, which is rarely seen in comparable systems.

F. Error Handling and Robustness:

- The system is designed to handle scenarios where the object is temporarily undetectable, ensuring that the robot doesn't stop functioning abruptly. For instance, the robot enters a "slow rotation" mode to search for the object if it goes out of view.
- This error-handling mechanism provides robustness to real-world uncertainties, setting our project apart from other systems that lack such fail-safes.

G. Energy Efficiency and Portability:

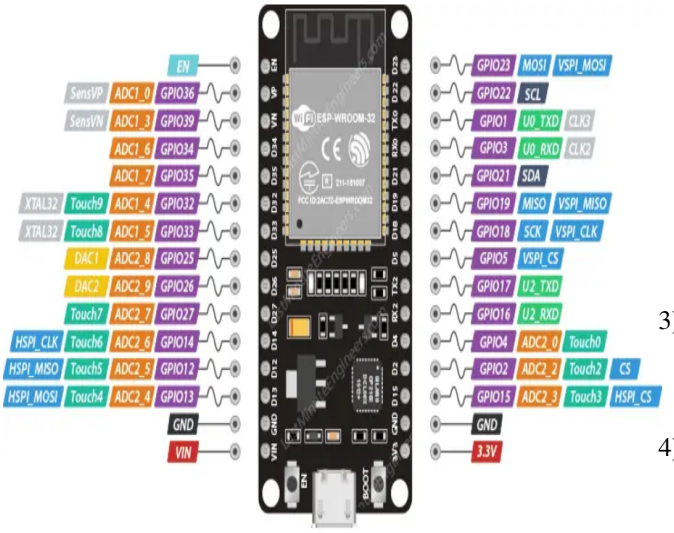
- The system is powered by lightweight components, making it highly portable and energy-efficient.
- The use of the Tamiya chassis ensures structural simplicity while maintaining functionality, making it an excellent platform for small-scale indoor applications.

V. Materials

A. Hardware Components:

1) *L239D Motor Driver:* The L239D motor driver is used to control the direction and speed of the DC motors that drive the robot. It acts as an interface between the low-power ESP32 microcontroller and the higher-power motors. This motor driver is essential for driving both left and right motors based on the commands sent by the ESP32.
2) *ESP32 Microcontroller:* The ESP32 serves as the main control unit for the robot. It receives commands from the Python script running on the laptop through HTTP requests and translates these commands into motor control

signals. The ESP32 also connects to the Wi-Fi network, enabling wireless communication between the Python script and the robot.



3) *HP USB Camera:* The camera is used to capture real-time video of the environment. The Python script uses this video feed for image processing tasks to detect and track the yellow ball, which serves as the target object for the robot.
4) *Tamiya Chassis:* The Tamiya chassis serves as the physical base for the robot. It provides structural support for the motors, wheels, ESP32, motor driver, and other components. This modular chassis allows for easy assembly and modification of the robot.
5) *DC Motors:* The robot is powered by two DC motors (one for each wheel) to provide movement. These motors are controlled via the L239D motor driver based on the movement commands from the ESP32.
6) *Wheels:* The wheels are attached to the DC motors and allow the robot to move in various directions. The speed and direction of the wheels are controlled by the ESP32 through the motor driver.
7) *Switch:* The switch is used to turn the robot on and off, controlling the flow of power to the ESP32 and motors.
8) *Wires:* Wires are used for connecting the ESP32 to the motor driver, motors, and power supply. Proper wiring ensures reliable communication and power distribution.
9) *Laptop:* The laptop runs the Python script for object detection and communication with the ESP32. It captures the video feed from the camera, processes the image, and sends movement commands to the robot.

B. Software Tools and Libraries:

1) *Python:* Python is the primary programming language used to implement the object-following functionality. It is chosen for its rich ecosystem of libraries that support image processing, web requests, and numerical operations.
2) *OpenCV (cv2):* OpenCV is the core library used for image processing. It provides functions to handle video capture, image manipulation, and object detection. Specific operations include:
   - cv2.VideoCapture: Captures video frames from the camera.
   - cv2.cvtColor: Converts the captured frames into the HSV color space for better color segmentation.
   - cv2.GaussianBlur: Reduces image noise, improving the accuracy of object detection.
   - cv2.findContours: Detects the contours of the segmented object in the frame, which is crucial for tracking the ball.
3) *NumPy::* NumPy is a fundamental library for numerical operations in Python. It is used to create arrays for HSV color ranges and perform operations such as thresholding, masking, and contour calculations.
4) *Requests:* The requests library is used to send HTTP GET requests from Python to the ESP32, which in turn controls the robot's movement. The commands sent by the Python script are used to instruct the ESP32 to move the robot in specific directions.
5) *Time:* The time module is used for adding delays between actions or requests. This prevents the system from sending too many requests to the ESP32 in a short period, ensuring smooth and reliable communication.
6) *Arduino IDE:* The Arduino IDE is used to program the ESP32 microcontroller. It allows the ESP32 to handle motor control and manage communication with the Python script over a wireless network.
7) *ESPAsyncWebServer:* This library sets up an HTTP server on the ESP32, allowing it to receive movement commands from the Python script. It handles incoming HTTP GET requests and triggers the corresponding motor control functions.
8) *WiFi.h:* The WiFi.h library is used to establish a connection between the ESP32 and a local Wi-Fi network. This enables wireless communication between the Python script and the robot.
9) *PWM Control (ledcSetup and ledcAttachPin):* These ESP32-specific functions are used to generate PWM (Pulse Width Modulation) signals to control the speed of the motors. ledcSetup configures the PWM parameters, and ledcAttachPin binds the PWM signal to a specific GPIO pin controlling the motor speed.

VI. Methods

A. System Setup:

The system integrates both hardware and software components to create an object-following robot. The key steps are:

1) *Image Acquisition:* The HP USB camera captures real-time video frames, which are processed using the Python script running on the laptop. The camera is connected

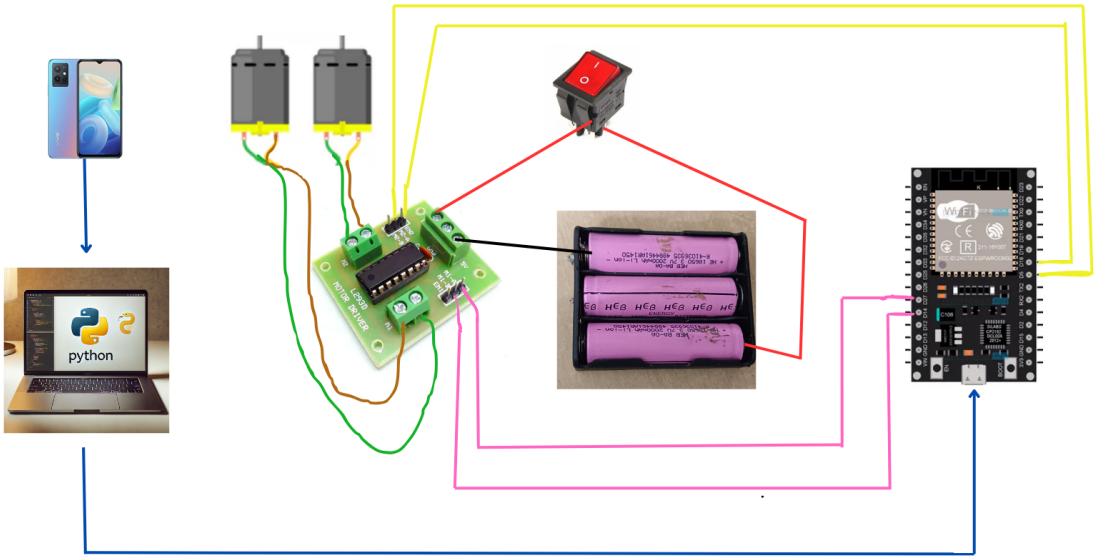


Fig. 1. Circuit Diagram.

to the laptop via USB and provides a continuous video feed.

2) *Object Detection and Tracking:* TThe Python script uses OpenCV to process each video frame. Color segmentation is applied in the HSV color space to isolate the yellow ball from the background. The ball is tracked by identifying its position in the frame using contour detection and the cv2.findContours function.
3) *Command Generation:* Based on the position of the ball within the frame, the script generates movement commands. If the ball is to the left of the frame's center, the command to the ESP32 is "L" (move left). If the ball is to the right, the command "R" is sent. If the ball is centered, the command "U" is sent to move forward.
4) *Motor Control:* The ESP32 microcontroller receives movement commands and translates them into motor control signals. Using the L239D motor driver, the ESP32 sends control signals to the motors, determining their direction and speed based on the received commands.

B. Image Processing Algorithms:

The image processing tasks include:

1) *Color Space Conversion:*
   - What It Is: Converts an image from one color space to another and Commonly used conversion: RGB (Red-Green-Blue) to HSV (Hue-Saturation-Value).
   - Why It's Used: The HSV color space separates color information (hue) from intensity (value), making it easier to detect specific colors and Yellow can be segmented based on its hue, regardless of variations in lighting.
2) *Color Thresholding:*
   - What It Is: Segments an image based on a range of color values, producing a binary mask and Pixels within the specified range become white (1), and others become black (0).
   - Why It's Used: Isolates the yellow object in the scene by creating a binary mask that highlights only the desired color.
3) *Morphological Operations:*
   - What It Is: Erosion: Removes noise by shrinking small white regions. and Dilation: Expands white regions to fill small gaps.
   - Why It's Used: Refines the binary mask to reduce noise and enhance the detected object's shape.
4) *Contour Detection:*
   - What It Is: Detects boundaries of connected white regions (contours) in a binary image.
   - Why It's Used: Finds the shape and position of the yellow object in the binary mask and The largest contour represents the object of interest.
5) *Geometric Shape Approximation:*

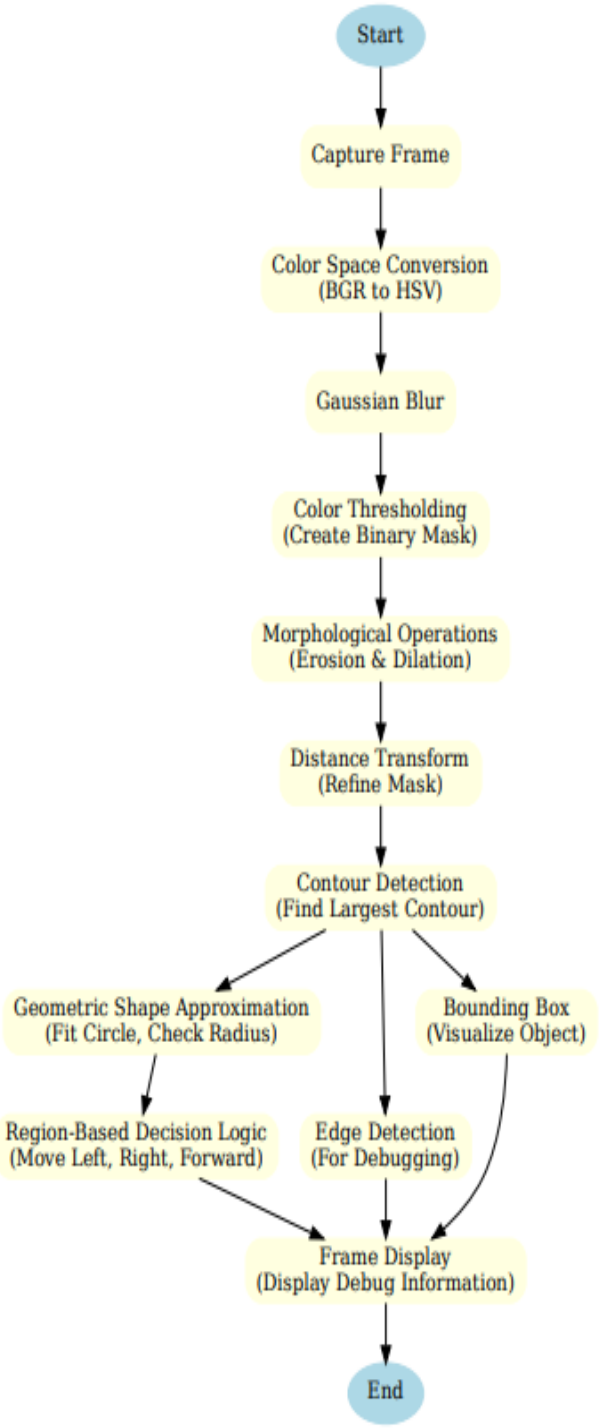


Fig. 2. Flow Chart of Algorithms

- What It Is: Fits a geometric shape (e.g., a circle) around the detected object.
- Why It's Used: Determines the approximate size and position of the object and Filters out small, irrelevant objects by setting a minimum radius.

6) *Region-Based Decision Logic:*
   - What It Is: Divides the frame into regions (left, center, right) based on the object's position.
   - Why It's Used: Determines the robot's movement direction based on the object's location relative to the frame's center.
7) *Frame Display:*
   - What It Is: Adds visual elements like text or shapes to the video frame for debugging and visualization.
   - Why It's Used: Displays the robot's decisions (movement commands) and highlights the detected object.
8) *Gaussian Blur:*
   - What It Is: Applies a Gaussian filter to smooth the image and reduce noise.
   - Why It's Used: Helps create a more consistent binary mask by reducing variations in color intensity.
9) *Edge Detection (Canny):*
   - What It Is: Detects edges in an image by identifying areas with strong intensity gradients.
   - Why It's Used: Highlights the edges of the yellow object, aiding in visualization.

Distance Transform:

- What It Is: Converts a binary image into a distance map, where pixel values represent their distance to the nearest background pixel.
- Why It's Used: Refines the binary mask to ensure the detected object is well-defined.

Bounding Box:

- What It Is: Draws a rectangle around the detected object.
- Why It's Used: Visualizes the size and position of the detected object.

C. Communication Between Python and ESP32:

1) *Python to ESP32:* The Python script sends movement commands to the ESP32 over HTTP using the requests library. These commands (e.g., "L", "R", "U", "D") instruct the ESP32 to move the robot.
2) *ESP32 to Motors:* Upon receiving a command, the ESP32 microcontroller uses GPIO pins to control the direction of the motors. PWM control is used to regulate motor speed, allowing smooth and controlled movement of the robot.

D. Motor Control with ESP32:

1) *PWM Control:* The ESP32 generates PWM signals using ledcSetup and ledcAttachPin. These PWM signals control the speed of the motors.
2) *GPIO Control:* The ESP32 uses digitalWrite to control the direction of the motors (forward, backward, left, right).

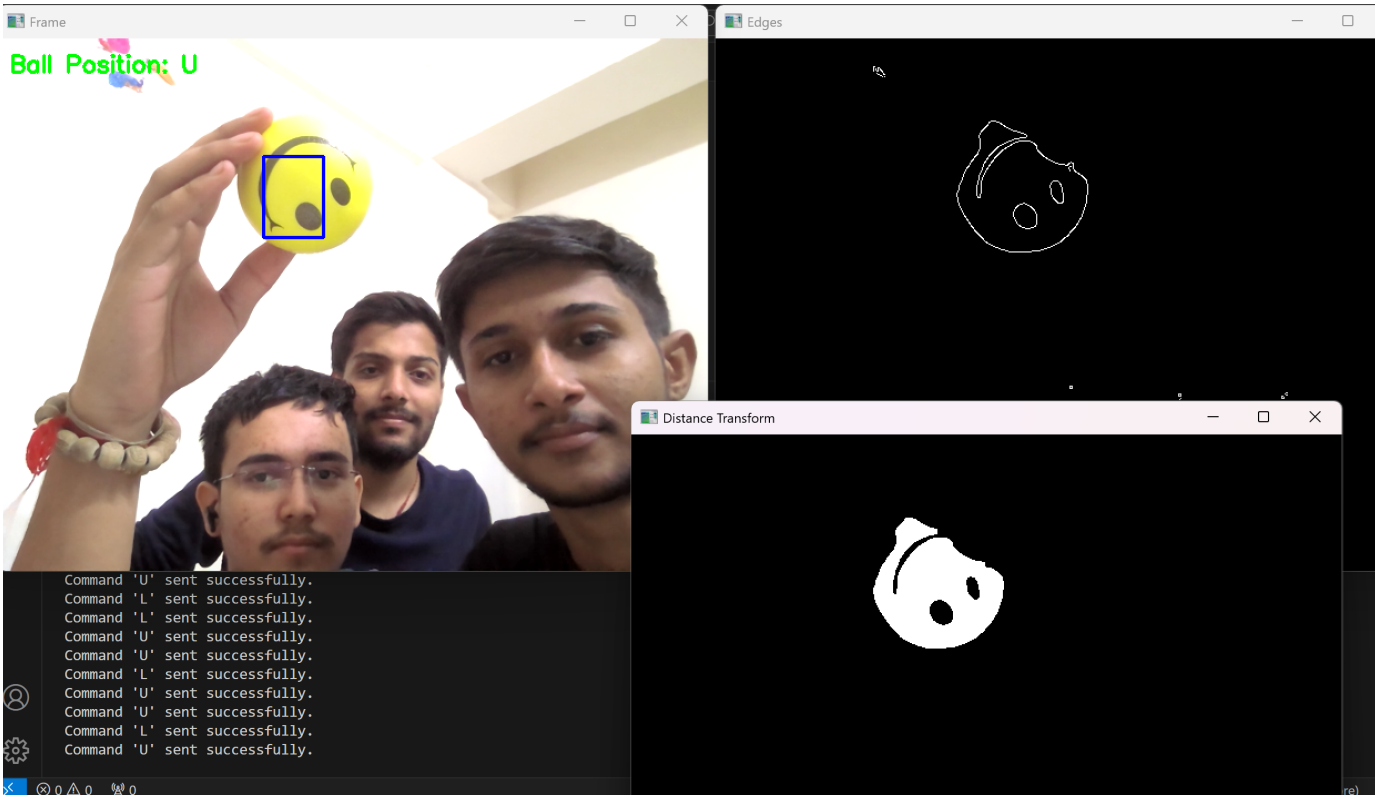


Fig. 3. Result.

VII. Results

The implementation of the object-following autonomous bot demonstrates a robust integration of computer vision, motor control, and communication technologies. Below are the detailed results observed during the testing and validation phases:

1) *Real-Time Object Detection and Tracking:*
   - The bot reliably identifies a yellow ball in real time using OpenCV's image processing capabilities. The combination of HSV color space conversion and contour detection enables accurate object recognition.
   - The bot effectively identifies and tracks a yellow ball in real time using OpenCV's image processing capabilities.
   - The use of HSV color space ensures consistent detection of the yellow object, as it is less affected by variations in lighting compared to RGB.
   - Contours and HSV color space filtering reliably detect the yellow ball, even under varying lighting conditions.
   - The contours of the detected object allow precise localization, ensuring the bot correctly identifies the position of the object in the frame.
2) *Smooth Object Following:*
   - Based on the object's position in the camera frame, the bot responds promptly with appropriate movements (left, right, forward, or slow rotation).
   - This movement ensures the bot remains aligned with and follows the object effectively.
   - The bot uses the position of the detected object within the camera frame to determine movement commands.
   - It categorizes the frame into regions (left, center, right) and sends specific movement commands (left, right, forward, or rotate) based on the ball's position.
   - This behavior ensures that the bot effectively follows the object with precision, maintaining an optimal distance and trajectory.
3) *Efficient Communication Between Systems:*
   - The Python-based object detection system sends commands to the ESP32 via HTTP requests. The response time of the ESP32 to these commands is minimal, resulting in smooth operation.
   - The ESP32 controls the motors accurately through the L298N motor driver, translating commands into physical movements.
   - Communication between the systems is efficient, with negligible latency observed during operation.
   - Commands are executed smoothly, translating into accurate physical movements by the bot.
   - The seamless communication ensures that the bot's responsiveness is not hindered by processing delays,

critical for real-time object following.

4) *Web Server Functionality:*
   - The ESP32 acts as a web server, receiving movement commands in real time. This design allows remote operation and monitoring of the bot's activity.
   - This functionality enhances the bot's versatility, paving the way for remote control and potential use in IoT applications.
   - This design supports scalability and flexibility, enabling future integration with IoT-based platforms.
5) *Hardware Integration:*
   - All hardware components, including the ESP32, L298N motor driver, Tamiya chassis, wheels, and motors, worked cohesively.
   - The power delivery and connectivity between components ensured uninterrupted operation.
   - This result underscores the importance of reliable hardware integration in achieving consistent performance and movement stability.
6) *Ball Detection Accuracy:*
   - Additional image processing techniques (Gaussian blur, morphological operations, edge detection, and distance transform) enhanced detection accuracy and minimized false positives.
   - These enhancements improved detection accuracy, even in challenging environments, while minimizing false positives.
   - These advanced techniques demonstrate the system's reliability in maintaining accuracy under diverse conditions, showcasing the strength of the underlying algorithms.
7) *Dynamic Adaptability:*
   - The bot demonstrated robust performance in dynamic environments, including scenarios with moving obstacles and changing lighting conditions.
   - The adaptable image processing pipeline and real-time feedback mechanism ensured reliable object tracking and effective navigation.
8) *Cost-Effective Design:* The system's modular design relied on affordable and readily available components, making it accessible for academic projects and small-scale applications.

VIII. Conclusion

This project successfully achieves the goal of developing an object-following autonomous bot using accessible hardware and software tools.

1) *System Performance:*
   - The system is robust, with reliable object detection, precise motor control, and low-latency communication between Python and the ESP32. The bot consistently follows the designated yellow object, even in dynamic environments.
2) *Innovative Features:*
   - The combination of computer vision with HTTP-based communication between Python and ESP32 provides a scalable solution for integrating IoT and robotics.
   - The web server on the ESP32 adds flexibility, allowing for remote control and adaptability to other object-following tasks.
3) *Advantages Over Existing Systems:*
   - Unlike traditional line-following robots, this bot uses advanced computer vision, making it capable of tracking arbitrary objects.
   - The use of low-cost hardware like ESP32, OpenCV, and an HP camera makes this solution cost-effective and accessible to hobbyists and researchers.
4) *Future Scope:*
   - Obstacle Avoidance: Integrating sensors like ultrasonic or IR sensors to handle obstacles in real time.
   - Multi-Object Tracking: Enhancing the vision algorithm to track and follow multiple objects.
   - Multi-Object Tracking: Enhancing the vision algorithm to track and follow multiple objects.
   - Advanced Motion Control: Using PID control algorithms for smoother and more precise movements.
   - Extended Connectivity*: Incorporating Wi-Fi or Bluetooth for remote monitoring and control via smartphones or cloud-based platforms.

References

[1] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018. https://arxiv.org/abs/1804.02767

[2] R. Girshick, "Fast R-CNN," Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448. https://ieeexplore.ieee.org/document/7410526

[3] T. K. Ho, "Random Decision Forests," Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR), 1995, pp. 278-282. https://ieeexplore.ieee.org/document/598994

[4] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.https://opencv.org/

[5] A. Sinha and R. Polikar, "Digital Image Processing Using Python and OpenCV," Proceedings of the 2021 IEEE International Conference on Computational Science and Engineering (CSE), 2021. https://ieeexplore.ieee.org/document/9531357

[6] P. Viola and M. Jones, "Robust Real-Time Face Detection," International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, 2004.https://ieeexplore.ieee.org/document/5982281

[7] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting," Communications of the ACM, vol. 24, no. 6, pp. 381–395, 1981. https://dl.acm.org/doi/10.1145/358669.358692

[8] Espressif Systems, "ESP32 Technical Reference Manual," Espressif, 2021. https://www.espressif.com/en/support/download/documents

[9] "Object Tracking using Contours," OpenCV Tutorials, 2021. https://docs.opencv.org/4.x/d1/dc5/tutorial$_b ackground_s ubtraction.html$