Ball Tracking Robot using Image Processing Techniques

1st Abdallah Falah
Mechatronics Engineering Department
German University In Cairo
Cairo, Egypt
abdallah.falah@student.guc.edu.eg

2nd Waleed Atef
Mechatronics Engineering Department
German University In Cairo
Cairo, Egypt
waleed.abdelrahman@student.guc.edu.eg

3rd Omar Magdy
Mechatronics Engineering Department
German University In Cairo
Cairo, Egypt
omar.abdelwaly@student.guc.edu.eg

4th Mohamed EShall
Mechatronics Engineering Department
German University In Cairo
Cairo, Egypt
mohammad.eshall@student.guc.edu.eg

5th Fouad Elsheikh
Mechatronics Engineering Department
German University In Cairo)
Cairo, Egypt
fouad.elsheikh@student.guc.edu.eg

***Abstract*—This paper presents the development of an au tonomous Arduino-powered car capable of detecting, tracking, and approaching a ball using image processing techniques. The system utilizes a camera module to identify the ball in real time, leveraging a combination of traditional computer vision methods and deep learning approaches for accurate detection under varying conditions. The car is equipped with an embedded control system that processes the ball's position and adjusts its movement accordingly. A PID controller ensures smooth navigation, allowing the car to follow the ball efficiently while minimizing sudden movements. Once the car reaches a predefined proximity to the ball, it stops to prevent collisions. The integration of efficient motor control, real-time image processing, and adaptive tracking algorithms enables reliable performance in dynamic environments. This work demonstrates a practical approach to autonomous ball tracking, with potential applications in robotics, sports analysis, and interactive robotic systems.**

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

I. Introduction

In an age where autonomous systems are becoming increasingly integrated into everyday life, the ability for a robot to visually track and follow a moving target is both a fascinating and practical challenge. This project focuses on building a mobile robot that uses real-time image processing to detect, locate, and track a moving object—in this case, a colored ball—within a controlled environment. The objective is to enable the robot to maintain accurate tracking by continuously adjusting its heading and speed based on visual feedback.

The robot is equipped with a camera that captures live video frames, which are then processed to isolate the target object using color-based segmentation. By converting the images to the HSV color space and applying filtering techniques, the system effectively detects the ball and calculates its centroid, representing the ball's position in the frame. The difference between the ball's position and the center of the robot's vision is computed to determine both direction and distance to the target.

To ensure smooth and responsive tracking behavior, a Proportional-Integral-Derivative (PID) control algorithm is implemented. This control system processes the positional error and dynamically adjusts the robot's motor commands to reduce deviation from the desired path. Commands are transmitted to the robot's Arduino controller via a serial interface, enabling precise control of both linear and angular movement.

This integration of computer vision and control systems allows the robot to behave autonomously and respond intelligently to a moving target. The approach can also be extended to track humans or vehicles by attaching distinct markers—such as colored tags, barcodes, or ArUco markers—making it applicable to a wide range of real-world scenarios, from smart surveillance to assistive robotics.

II. Methodology

A. Hardware Fabrication and Setup

The ball-tracking robot system was developed using a Raspberry Pi and an Arduino microcontroller. The Raspberry Pi was responsible for image acquisition and processing, while the Arduino handled motor control based on commands received from the Raspberry Pi.

Hardware Components:

- Raspberry Pi 4 (image acquisition and processing)
- Arduino Uno (motor control execution)
- Raspeberrypi Cam v2
- L298N Motor Driver
- DC Motors
- Chassis with wheels
- Power Bank (mobile power supply)

Connections Overview:

- Raspeberrypi Cam v2 is connected to the Raspberry Pi for real-time video input.

- Serial communication via USB is established between the Raspberry Pi and Arduino for command transmission.
- The Arduino generates PWM signals to control motor speed and direction via the L298N motor driver.

Fig. 1. Hardware

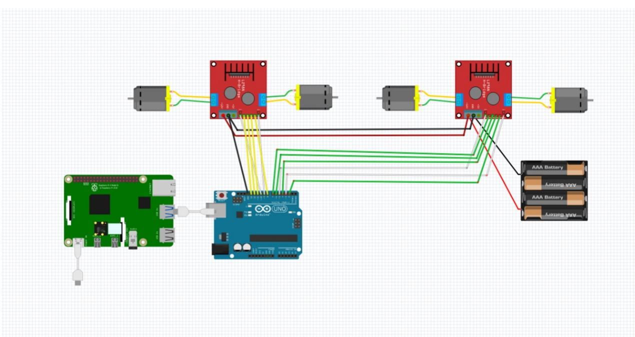

Fig. 2. Wiring Diagram

B. Image Processing Pipeline

1) G: eometric Transformation

- **Resizing:** Input frames are scaled to half their original size to reduce computation time.
- **Flipping:** Frames are horizontally flipped to match the camera's orientation with the physical motion of the robot.

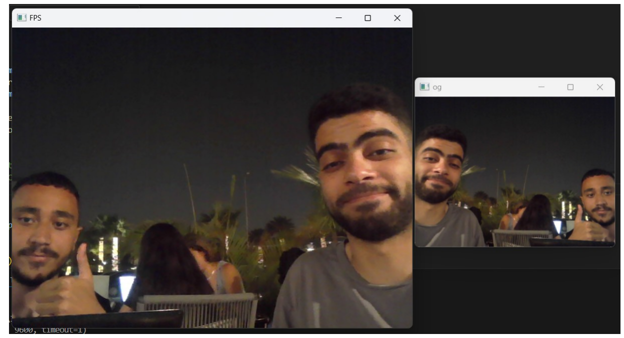

Fig. 3. Geometric Transformation

2) CLAHE (Contrast Limited Adaptive Histogram Equalization) : Histogram equalization is a computer vision technique that improves the contrast on an image, it does that by spreading out the intensity range of the image thus increasing the contrast. Adap tive Histogram Equalization dissects the image computes several histograms for each section and uses them to redistribute the lightness values of the image. CLAHE [4] has a contrast limiting procedure that is applied to each neighborhood of equalization, which prevents the over-amplification of noise that the adaptive histogram can give

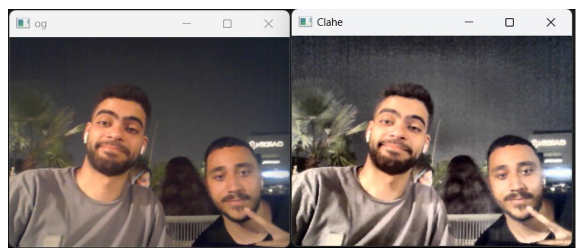

Fig. 4. CLAHE

3) Smoothing: A Gaussian Blur is applied to the enhanced image to suppress noise and stabilize subsequent feature detection steps.

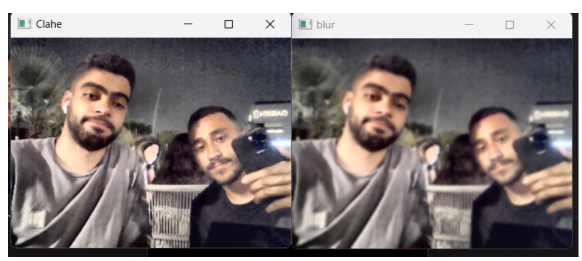

Fig. 5. Gaussian Blur

4) HSV Color Segmentation: HSVcolor space was used to segment the colors because it's the most commonly used method and easiest to understand, this is done by creating track bars for the upper and lower bounds of the HSV color space, H from 0 to 180, S form 0 to 255, and V from 0 to 255. Then the values that have been chosen is used in the cv2.inrange method to create a mask that is later used.

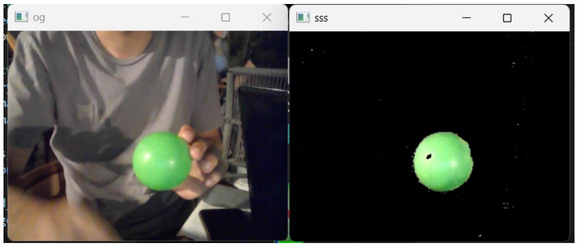

Fig. 6. HSV Color Segmentation

5) Morphological Operations: Morphological closing followed by opening is applied to the binary segmentation mask to eliminate small holes and remove isolated noise, improving the shape integrity of the detected object.

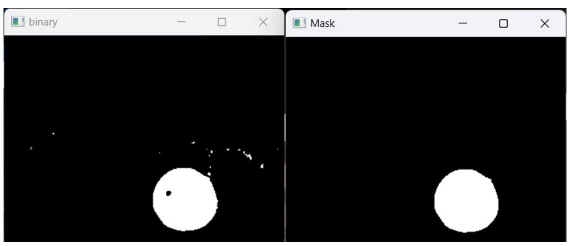

Fig. 7. Morphological Operations

6) Contour Detection and Centroid Estimation: After applying color-based segmentation and morphological operations to clean the binary mask, contours are extracted using OpenCV's findContours function. Among the detected contours, the one with the largest area (above a defined minimum) is assumed to be the target object (the ball). The centroid of this contour is computed using image moments, which represent the distribution of pixel intensities. The centroid coordinates (cx, cy) are calculated from the spatial moments, and are used to determine the horizontal offset of the ball from the center of the frame. This offset becomes the input for the PID controller to steer the robot accordingly.

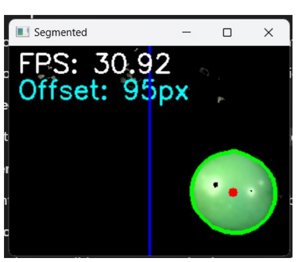

Fig. 8. Contour Detection And Centroid Estimation

C. Control System

The robot's control system is designed to track and follow a green-colored object using visual input from a PiCamera and communicate motor commands via a serial connection to an Arduino. This system utilizes image processing and a Proportional-Integral-Derivative (PID) control approach to ensure accurate alignment and distance regulation relative to the target.

1) Horizontal Alignment – PI COntrol: To align the robot horizontally with the detected target, a **PID controller** is employed based on the horizontal offset between the centroid of the object and the center of the frame. The control output is computed as:

$$x = K_p \cdot e_x + K_i \cdot \sum e_x + K_d \cdot \Delta e_x$$

Where:

- e_x is the current horizontal offset (error),
- K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively.

If the horizontal offset exceeds a specified threshold, the robot rotates left ('L') or right ('R') with a PWM speed proportional to the PID output.

2) Forward Motion Control: When the horizontal alignment is within the threshold, a constant PWM activates forward motion based on the **contour area**, which indirectly represents the distance to the target. The control logic attempts to maintain the target's area within a specified range:

- If the area is smaller than the desired maximum, a forward ('F') command is issued.
- If the area exceeds the threshold, the robot stops ('S').

3) Command Throttling: To prevent overwhelming the serial communication channel or the Arduino, commands are throttled and sent only every N frames, controlled by a commandinterval counter.

D. Algorithm

Algorithm 1: Green Object Tracking and Control with PiCamera and Arduino

```
Data: Camera input, Serial connection to Arduino
Result: Green object tracking and control commands sent to Arduino
Initialize serial connection to Arduino
Initialize CLAHE (Contrast Limited Adaptive Histogram Equalization)
Configure and start Picamera2
Initialize PID parameters and thresholds
Initialize HSV bounds for green color
while True do
    Capture frame from camera
    Convert frame from RGB to BGR
    Increment frame count
    if 1 second elapsed then
        Calculate FPS
        Reset frame count and timer
    end
    Flip and resize frame
    Equalize Y channel of image using CLAHE
    Apply Gaussian blur
    Convert frame to HSV
    Threshold HSV image for green range
    Apply mask to obtain segmented image
    Convert segmented image to grayscale and binarize
    Apply morphological open and close
    Find contours in binary mask
    Filter contours based on minimum area
    Set default command to stop (S000)
    if valid contours exist then
        Select largest contour
        Compute contour area and draw it
        Compute centroid (cx, cy)
        Calculate horizontal offset from image center
        Apply PID control for horizontal alignment
        if offset ¿ threshold then
            Calculate PWM based on PID output
            Set direction (L or R) based on offset sign
            Encode and set command
        end
        else
            Apply PID for distance control using area
            if area in valid range then
                Set forward command with PWM
            end
            else
                Stop command
            end
        end
    end
    else
        Stop command
    end
    if time to send command then
        Send command to Arduino via serial
        Print sent command
    end
    Display processed frames (segmented, mask, original)
    if key 'q' pressed then
        Send stop command and break loop
    end
end
Stop camera and close serial connection
Destroy all OpenCV windows
```

III. Results And Limitations

A. Performance Comparison: Raspberry Pi vs Laptop

A performance comparison was conducted to evaluate the real-time efficiency of the system across two platforms: a Raspberry Pi and a standard laptop. • The Raspberry Pi achieved a lower frame rate and control response due to limited computational power of the pi. • Both platforms demonstrated comparable image pro cessing performance, confirming the system's real-time suitability.

B. Overall System Results

The system maintained an average processing speed of approximately 15 FPS. The PI controller with a Kp gain of 0.3 and Ki of 0.01 for horizontal alignment and distance regulation demonstrated stable and smooth motor commands, minimizing overshoot and oscillations, The horizontal offset PI controller adjusted the steering direction to center the object in the frame, while the distance control using a constant PWM forward motion based on the object's perceived size. The communication with an Arduino microcontroller over serial allowed real-time transmission of directional commands. The command throttling mechanism, which sent control signals every 5 frames, prevented overwhelming the serial interface and ensured consistent motor behavior.

C. Limitations

Limitations included sensitivity to green shades in the environment and the fixed HSV thresholds, which may require tuning for different scenarios. Also, fast object movements with limited FPS and Computatuional power made the system lag and overshoot.

IV. Conclusion

This project successfully demonstrated the design and implementation of an autonomous ball-tracking robot using real-time image processing and embedded control. By integrating a Raspberry Pi for image analysis and an Arduino for motor control, the system efficiently detected and followed a colored ball in a controlled environment. The combination of HSV color segmentation, morphological processing, and contour detection allowed for accurate target localization, while the PID control mechanism ensured smooth and responsive movement.

Although the system faced limitations in processing speed and sensitivity to lighting and color variations, it maintained stable tracking performance under typical conditions. The modular nature of the design allows for future enhancements, such as adaptive thresholding, improved object recognition techniques, and application to more complex dynamic environments. This work lays a practical foundation for expanding vision-based autonomous systems in fields like robotics, surveillance, and human-computer interaction.