Design and Development of a Robotic Arm

Kartik Agrawal, Jayesh Thakre, Nimesh Suryavanshi, Vivek Kumar

bt24eci001@iiitn.ac.in,bt24eci002@iiitn.ac.in, bt24eci006@iiitn.ac.in, bt24eci023@iiitn.ac.in
Department of Electronics and Communication Engineering (IOT)

Indian Institute of Information Technology Nagpur
Maharastra, India

Abstract— **This paper presents the design and development of a robotic arm equipped with computer vision to identify and grasp objects based on their color. By leveraging an ESP32-CAM module for real-time video capture, the system transmits video feeds to a Python program over Wi-Fi, where the OpenCV library processes the image data to perform color segmentation and object identification. Control commands are sent to an Arduino Uno microcontroller, which coordinates the robotic arm's movements via servomotors. This low-cost and effective system integrates Python, Arduino, and custom 3D-printed components to demonstrate a robust solution for color-specific object handling in various practical and industrial applications. Potential uses include automated sorting, object handling, and basic assembly tasks, particularly in environments where the objects of interest can be identified by unique color features.**

Keywords— **Robotic arm, Computer vision, Colour recognition, OpenCV, Arduino, ESP32-CAM**

I. INTRODUCTION

The integration of autonomous robotic systems in industrial automation has significantly advanced object detection and manipulation technologies, making it possible to develop cost-effective solutions for specific tasks such as sorting, assembly, and handling. Robotic arms equipped with vision systems can now be designed to perform tasks that previously required human intervention, especially in applications where color-based identification is viable.

This project focuses on developing a robotic arm capable of identifying and grasping objects based on their color using computer vision. The ESP32-CAM module serves as the primary sensor, capturing live video that is processed in real-time using OpenCV in a Python environment. Based on the detected color and location of the object, control commands are sent to an Arduino Uno microcontroller, which coordinates the movement of four servos responsible for the arm's positioning and gripping actions. The system is designed to be modular and affordable, leveraging 3D-printed components for the arm structure and relying on widely available hardware and software tools.

II. SYSTEM DESIGN AND COMPONENTS

A. Hardware Components

1. **Arduino Uno:** The Arduino Uno is the primary microcontroller that coordinates the robotic arm's actions. It receives position and movement commands from the Python program and translates these commands into specific servo adjustments. The Arduino's reliability and compatibility with various sensors and actuators make it ideal for this project.

2. **ESP32-CAM Module:** The ESP32-CAM is a Wi-Fi-enabled camera module that streams real-time video to a Python script for image processing. The module provides sufficient image quality for color segmentation while being affordable and easy to integrate. The ESP32-CAM operates as a local server, transmitting video over Wi-Fi, which allows the system to function in environments with a remote processing unit.

3. **Servomotors (MG995):** The MG995 servomotors provide the mechanical movement for the arm's joints. Four servos control the robotic arm's shoulder, elbow, and wrist joints, as well as the gripper. The servos offer high torque and precise positioning, which are essential for achieving accurate and repeatable movements.

4. **Robotic Arm Structure:** The arm's structure is 3D-printed using durable and lightweight materials. This modular design allows for easy assembly and customization. The structure houses all components, including the Arduino, ESP32-CAM, and servos, ensuring that they are securely mounted and optimally positioned for operation.

5. **Wiring and Connectivity:** Jumper wires and USB cables connect the ESP32-CAM, Arduino, and servos, facilitating reliable data transmission and power distribution. The setup is simple yet robust, ensuring that each component receives a stable connection without interference.

B. Software Components

1. **Python and OpenCV:** The Python programming language, along with the OpenCV library, handles the video feed and processes it to detect specified colors. OpenCV performs HSV-based color segmentation, allowing flexibility in choosing target colors. Python provides a versatile environment to control the system's functions and send commands to the Arduino over serial communication**.**

2. **Arduino IDE:** The Arduino Integrated Development Environment (IDE) is used to program the Arduino microcontroller, enabling it to interpret commands from Python and convert them into precise motor movements. The Arduino IDE supports serial communication with Python and provides libraries to manage servo controls efficiently**.**

III. MECHANICAL DESIGNING

A. 3D Model of the Arm Components

- Software: Use CAD software like Autodesk Fusion 360 or SolidWorks to design the arm structure.
- Segments: Design the arm with multiple segments or links (typically 2-3 for a basic arm).
- Base: Provides stability and houses the electronics.
- Shoulder and Elbow Joints: These provide movement in two primary directions (up-down and forward-backward).
- Wrist and Gripper: The wrist offers rotation, and the gripper is designed to hold and release objects.
- 3D Printing: After designing, export STL files and use a 3D printer to print each component.
- Material: PLA or ABS is commonly used for strength and durability.

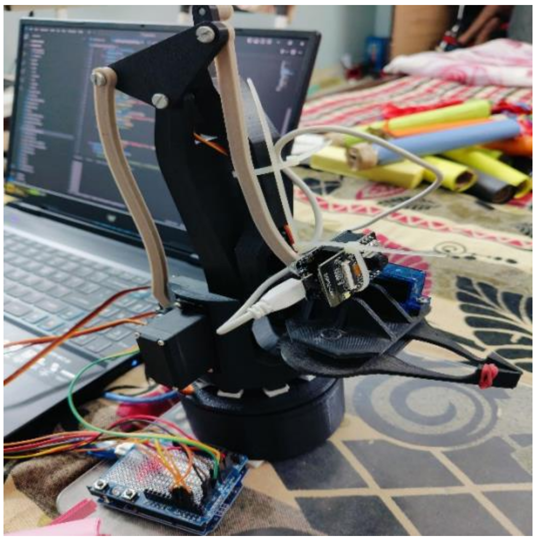

Fig 1. Robotic Arm Model

Fig 1. shows the exact model of Designing a color-recognizing, object-sorting robotic arm using an Arduino involves several stages, from defining specifications to final testing. The process begins by setting up requirements, such as using 3D-printed components, cost-effective parts, and achieving reliable color detection with a sufficient range of motion. Mechanical design includes modeling the arm with segments, joints, and a gripper, then printing components in PLA or ABS. These parts are assembled with servo motors to enable movement.

The electronics feature an Arduino microcontroller, a compatible camera sensor for color recogniton (e.g.ESP32 CAM) and a power management system for the servos. Programming the Arduino involves setting up inverse kinematics for precise arm control and defining thresholds for color recognition. The firmware enables the arm to read sensor input, make color-based decisions, and move objects to specific bins. Calibration and testing involve fine-tuning servo angles, improving color detection accuracy, and ensuring reliable performance. Optimizations include code efficiency and power management improvements. This robotic arm can serve educational, industrial, or home purposes, with the potential for future upgrades, such as IoT integration and additional features.

IV. METHODOLOGY

The methodology consists of three main stages: vision processing, robotic arm control, and ESP32-CAM configuration.

A. Vision Processing

Video Stream Initialization

The ESP32-CAM module captures real-time video and streams it to a Python script running on a computer. The module is configured to connect to a local Wi-Fi network, establishing a video feed accessible to the Python program.

Color Segmentation and Object Detection

The Python script uses OpenCV to apply color filtering based on the HSV color model, which is ideal for segmenting objects by color in varying lighting conditions. The program defines an HSV range corresponding to the target color, which can be adjusted for different applications.

Pseudo-code Example for Color Detection:

css

Copy code

```
Initialize video stream from ESP32-CAM.
Define target color in HSV range.
Capture each frame from video feed.
Convert frame from RGB to HSV format.
Apply color mask to filter out specified color.
Detect contours in masked image.
If contours meet size threshold:
Extract object coordinates.
Send coordinates to Arduino for positioning.
```

Contour Analysis

After applying the color filter, the script performs contour analysis on the filtered image to identify the shape and size of the object. Only contours of a specified minimum size are considered, which reduces false detections. Once a valid contour is detected, the object's position is calculated, and commands are sent to the Arduino to initiate arm movement.

B. Robotic Arm Control

The Arduino Uno microcontroller receives position and movement commands from the Python program and directs the servos to move the arm into the appropriate positions. The system supports a variety of movement commands such as "Left," "Right," "Center," and "Grasp," which guide the robotic arm toward the detected object.

Pseudo-code Example for Arm Control:

vbnet

Copy code

```
If object detected:
Calculate object's relative position to arm.
If object is to the left of arm:
   Send "Move Left" command to Arduino.
If object is centered:
   Send "Grasp" command to Arduino.
Position servos to align gripper with object.
   Close gripper to secure object.
```

C. ESP32-CAM Configuration

The ESP32-CAM module is set up to function as a local video server, transmitting video frames to a Python processing unit over Wi-Fi. The configuration includes setting up Wi-Fi credentials, initializing the camera feed, and adjusting resolution settings to ensure smooth transmission and high enough image quality for OpenCV to perform accurate color segmentation.

V. RESULTS AND DISCUSSION

Testing confirmed the robotic arm's capability to accurately detect and grasp objects based on color. The system was evaluated across multiple lighting conditions, and adjustments to the HSV color range were necessary to maintain accuracy. Under optimal lighting, the color recognition achieved high precision, enabling the arm to respond accurately to objects within the specified color range.

Some delays in processing were observed due to the video transmission over Wi-Fi, but the arm consistently executed positioning commands with minimal deviation. Adjustments to the servos' positioning parameters and the color segmentation thresholds were required to improve accuracy in certain situations.

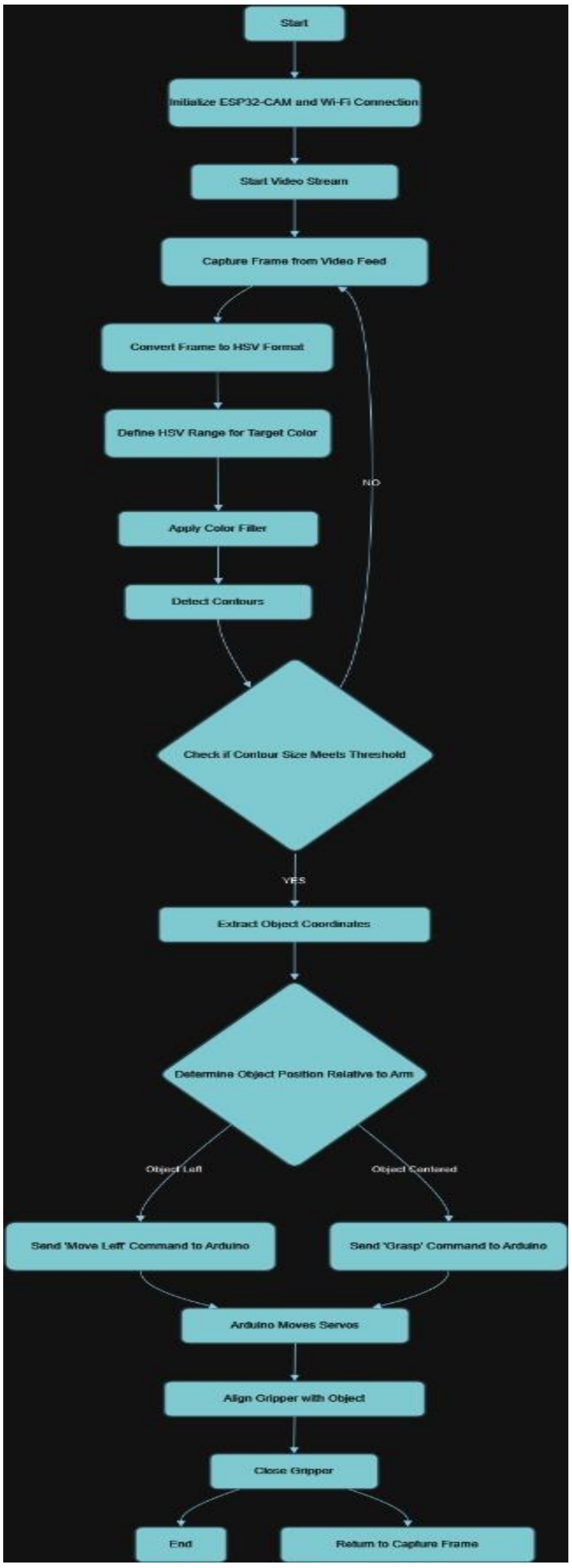
Start
Initialize ESP32-CAM and Wi-Fi Connection
Start Video Stream
Capture Frame from Video Feed
Convert Frame to HSV Format
Define HSV Range for Target Color
Apply Color Filter
Detect Contours
NO
Check if Contour Size Meets Threshold
YES
Extract Object Coordinates
Determine Object Position Relative to Arm
Object Left
Object Centered
Send 'Move Left' Command to Arduino
Send 'Grasp' Command to Arduino
Arduino Moves Servos
Align Gripper with Object
Close Gripper
End
Return to Capture Frame

Fig 2. Flowchart of processing Robotic Arm

VI. LIMITATIONS

Despite its effective performance, the robotic arm system has several limitations:

Limited Field of View (FOV) :
The ESP32-CAM's 30-degree field of view restricts the arm's ability to detect objects outside this narrow range, making it less suitable for dynamic environments with objects distributed across a wider area.
Suggested Improvement: Future iterations could include multiple cameras or a camera with a wider FOV to enhance detection coverage.

Rotation Constraints :
The robotic arm is limited to 180 degrees of rotation, preventing it from accessing objects located behind it. This limits its range of operation in tasks that require a full 360-degree reach.
Suggested Improvement: Adding a rotating base could extend the arm's operational range, allowing it to handle objects located in a 360-degree workspace.

Lighting Sensitivity :
The HSV-based color segmentation is affected by changes in ambient lighting, leading to decreased accuracy in low or fluctuating light conditions.
Suggested Improvement: Implementing adaptive lighting control or using machine learning algorithms could improve color recognition robustness under varied lighting conditions.

Processing Delay :
Real-time video processing incurs slight delays due to Wi-Fi transmission and image analysis. This delay may hinder responsiveness in applications requiring immediate feedback.
Suggested Improvement: A faster microcontroller or optimized processing code could reduce response times, and a wired connection may improve real-time performance.

VII. CONCLUSION

This robotic arm system demonstrates a practical and cost-effective approach to color-based object manipulation, with potential applications in industrial sorting and basic assembly tasks. The system combines Arduino, ESP32-CAM, and OpenCV technologies to deliver autonomous functionality suitable for targeted applications. With further enhancements in multi-color recognition, object shape detection, and adaptability to varied lighting, this robotic arm

References

[1]. Kurbanhusen Mustafa, S., Yang, G., Huat Yeo, S., Lin, W. and Chen, M., 2008. Self-calibration of a biologically inspired 7 DOF cable-driven robotic arm. Mechatronics, IEEE/ASME Transactions on, 13(1), pp.66 75.

[2]. Kim, H.S., Min, J.K. and Song, J.B., 2016. Multiple-Degree-of-Freedom Counterbalance robot Arm Based on Slider-Crank Mechanism and Bevel Gear Units. IEEE Transactions on Robotics, 32(1), pp.230-235.

[3]. Kim, H.J., Tanaka, Y., Kawamura, A., Kawamura, S. and Nishioka, Y., 2015, August. Development of an inflatable robotic arm system controlled by a joystick. In Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on (pp. 664-669). IEEE.

[4]. Mohammed, A.A. and Sunar, M., 2015, May. Kinematics modeling of a 4-DOF robotic arm. In Control, Automation and Robotics (ICCAR), 2015 International Conference on (pp. 87-91). IEEE.

[5] .Siciliano, B., 2009. Robotics. London: Springer

[6]. Zhang, Z., 2015. Wearable sensor technologies applied for post-stroke rehabilitation (Doctoral dissertation, RMIT University).

[7]. Qassem, M.A., Abuhadrous, I. and Elaydi, H., 2010, March. Modeling and Simulation of 5 DOF educational robot arm. In Advanced Computer Control (ICACC), 2010 2nd International Conference on (Vol. 5, pp. 569 -574). IEEE.

[8]. Bhuyan, A.I. and Mallick, T.C., 2014, October. Gyro-accelerometer-based control of a robotic Arm using AVR microcontroller. In Strategic Technology (IFOST), 2014 9th International Forum on (pp. 409-413). IEEE.

[9]. Anon, 2016. The Ninth International Symposium. [online] Robotics Research. Available at: http://Robotics Research: The Ninth International Symposium [Accessed 4 Apr. 2016] [10] Condit, R. and Jones, D.W., 2004. Stepping motors fundamentals. Mi crochip Application Note: AN07,[Online]. Available: www. microchip. Co