The International Journal of Engineering and Information Technology

journal homepage:www.ijeit.misuratau.edu.ly

Designing and Implementing a 3DOF Robotic Arm for Color Sorting and Object Identification Using Computer Vision Technology

Aburas, O.
Elmergib University Libya
oeaburas@elmergib.edu.ly

Hussein, M.
École de technologie supérieure ,
Université du Quebéc,Canada
Mahmoud.hussein.1@ens.etsmtl.ca

Elnageh, A.
College of Electronic Technology,
Tripoli, Libya
aalnajeh11@gmail.com

Ebshish, M.
Elmergib University Libya
mohammedebshish14@gmail.com

Algadi, M.
Elmergib University Libya
azousef@gmail.com

Abstract— **This paper focuses on designing a three-degree-of-freedom robotic arm for color sorting using computer vision technology. It involves the use of a robotic arm composed of three joints and a gripper for grasping and manipulating objects. Each joint is actuated by a servo motor that is controlled by an Arduino microcontroller to control the angles of the joints and achieve the desired motion. The microcontroller relies on a camera to detect and analyze colors. The camera captures an image of the objects present in the designated area and converts it into digital data to be processed later by a computer. Then, the well-known (OpenCV) and (NumPy) library in the Python programming language is used to process this data and identify the colors. Once the color is determined, the corresponding Python command is executed through the Arduino microcontroller. That allows the robotic arm to move towards the designated location based on the identified color. Consequently, the arm grasps the detected object and places it in the appropriate position according to its color. The effectiveness of the proposed approach has been verified in real-time experiments.**

Keywords—**Robotic Arm; Arduino; Color Sorting; Computer Vision; Python Programming.**

I. INTRODUCTION AND BACKGROUND

A. *INTRODUCTION*

Robots are programmable machines designed to perform tasks autonomously or under remote control. Mobile robots can move from one place to another, often equipped with wheels or tracks and various sensors for navigation and obstacle avoidance [1], [2]. On the other hand, robotic arm is a type of manipulator that mimics the function of a human arm. It typically consists of a series of interconnected links and joints, allowing it to move in multiple degrees of freedom. Robotic arms are widely used in industrial automation, medical surgery, and space exploration, among other applications. They are often equipped with end-effectors, such as grippers or tools to perform specific tasks [3], [4], [5].

Nowadays, robotic arms are extensively utilized in industrial settings. They can take over tasks that are repetitive or involve heavy lifting, which would traditionally require human labor. These robots can work effectively and provide high consistency and dependability. They contribute to the creation of precise and high-quality products [6]. The computer vision system is one of the important fields in automating robotics using image processing. Image processing is the process of converting an image into a digital form and applying operations to extract data from it, such as determining the color and geometry of objects [5], [7].

The development and integration of computer vision technology has played a crucial role in advancing the capabilities of robotic systems. Computer vision enables robots to perceive and analyze visual information from their surroundings, empowering them to make informed decisions and execute tasks with precision. This technology is particularly valuable in color sorting applications, where accurately identifying and segregating objects based on color is essential. In this paper, we present a comprehensive study on the design and implementation of a 3DOF robotic arm for color sorting, leveraging the capabilities of computer vision technology. The proposed system utilizes an Arduino microcontroller board, servo motors for actuation, and employs advanced image processing algorithms implemented using Python with the OpenCV and NumPy, libraries.

Received 30 Apr , 2024; revised 13 May, 2024; accepted 15 Mar 2024.
AVAILABLE ONLINE 08 AUG, 2024.

B. RELATED WORKS

In recent years, many researchers have shown an increasing interest in the development of robotic arm systems for color sorting applications. For example, ZeChen proposed a solution towards industry 4.0 by introducing color-based object sorting using a robot arm and real-time object detection [8]. The authors Tejaswini and Sporting presented a system for real-time object sorting based on 3D parameters through image processing and robotic arm technology [9]. Cong et al. designed and implemented a robot arm system for classification and sorting using machine vision technology in [10]. Lennon and Shivakumar introduced an innovative technique of color sorting and object identification for a robotic arm. Their proposed approach not only streamlines the sorting process but also enhances efficiency and accuracy in diverse applications [11].

These related works have contributed to advancements in the field of robotic arm-based color sorting systems, emphasizing real-time object detection, image processing, and computer vision technology. Motivated by the above discussions, this paper aims to enhance the object identification and color sorting processes and achieve higher precision and efficiency using computer vision technology integrated with a movable robotic arm. It can be implemented in various fields of industry such as the food industry, packaging, and automated manufacturing.

C. CONTRIBUTIONS

Some potential main contributions that could be included in the paper:

1. The main challenge in this research stems from the need to depend only on one camera, eliminating the utilization of other sensors that would prolong data processing. The decision to opt for this particular camera is driven by its cost-effectiveness and its compatibility with integration into any robotic system.

2. Designing a 3 DOF robotic arm with two links and three joints actuated by servo motors involves several steps including conceptualization, mechanical design, selection of servo motors, electronics design, and programming.

3. Deploying Python programming language in conjunction with the OpenCV and NumPy, libraries for object detection and recognition involves several steps, from setting up the environment to implementing the detection algorithms and deploying the solution.

The rest of the paper is organized as follows; section II represents the dynamic model and experimental work, section III presents computer vision technology, while section IV represents robot arm controlled by sorting the colors using computer vision.

II. ROBOTIC KINEMATICS AND EXPERIMENTAL WORK

A - KINEMATICS OF ROBOTIC ARM

The mechanical structure of a 3DOF (3 degrees freedom) robotic arm typically consists of three main components [12]: the base, the arm segments, and the end effector as shown in Figure 1.

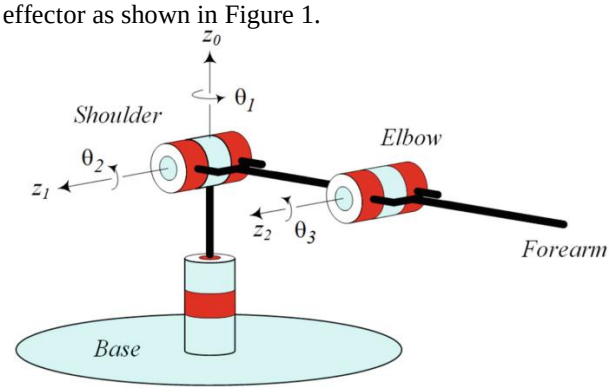

Figure 1: 3DOF robotic arm

1. Base: The base serves as the foundation for the robotic arm and provides stability and support. It often includes a motor or mechanism that allows the arm to rotate horizontally, providing the first degree of freedom.

2. Arm Segments: The robotic arm consists of two linked segments, which are connected by joints. These joints allow the arm to move in different directions, providing additional degrees of freedom. The joints are often actuated using servo motors or other actuators.

3. End Effector: The end effector is the tool or device attached to the end of the robotic arm that interacts with the environment. It can be a gripper, a tool for manipulation, or a sensor for measuring and interacting with objects. The end effector is responsible for performing the specific tasks for which the robotic arm is designed.

The mechanical structure is designed to provide flexibility and precision in movement, allowing the robotic arm to reach different points in its workspace and manipulate objects with accuracy. The arrangement of the segments and joints determines the range of motion and the types of tasks the robotic arm can perform. Additionally, the materials used in the construction of the arm are chosen to provide strength, durability, and lightweight characteristics to optimize the arm's performance.

Forward Kinematics and Inverse Kinematics are fundamental concepts in the field of robotics and mechanical engineering. These concepts are used to describe the relationship between the position and orientation of a robot's end-effector (such as a gripper) and the joint angles of the robot's manipulator [13].

In this paper the working principle depends on the inverse kinematics equations which determine the joint angles $(\theta_1, \theta_2, \theta_3)$ given the position and orientation of the end effector. The inverse kinematics can be given by the following equations [14]:

$$\theta_1 = tan^{-1}(x/y) \quad (1)$$

$$\theta_3 = tan^{-1}\left(\sqrt{1-c_3^2}/c_3\right) \quad (2)$$

$$c_3 = \left(\frac{y^2}{s_1^2} + z^2 - L_1^2 - L_2^2\right)/(2L_1L_2)$$

$$\theta_2 = tan^{-1}\left(\frac{-yL_2c_1s_3 + xL_2s_1s_3}{y(L_2c_1s_3 + L_1c_1) - x(L_2s_1s_3 + L_1s_1)}\right) \quad (3)$$

Where: θ_1 is the angle of the base, θ_2 and θ_3 are the angles of second and the last joint before the end-effector respectively, and L_1 & L_2 represents the lengths of the

respective links. x, y and z are the coordinates of the end effector from the base.

B. EXPERIMENTAL WORK

The servo motors interface directly with the Arduino Uno microcontroller. Both the Arduino Uno board and the camera are interfaced with a laptop computer, enabling real-time monitoring and control. The servo motors precisely govern the robot arm's movements, while the power supply generator ensures a consistent power delivery. Through this integrated system, the robotic arm executes tasks with remarkable precision. The circuit is connected on a breadboard for efficient connectivity. Upon connection to the laptop computer, the robotic arm initiates operation, ready to fulfill its designated tasks.
This proposed system, as shown in Figure 2, costs approximately 200 dinars. This cost does not include the price of the computer.

In this work, careful consideration was given to selecting items of suitable and practical quality, as well as affordable in price. This means that elements and components were chosen to fulfill the main purpose while being reasonably priced.

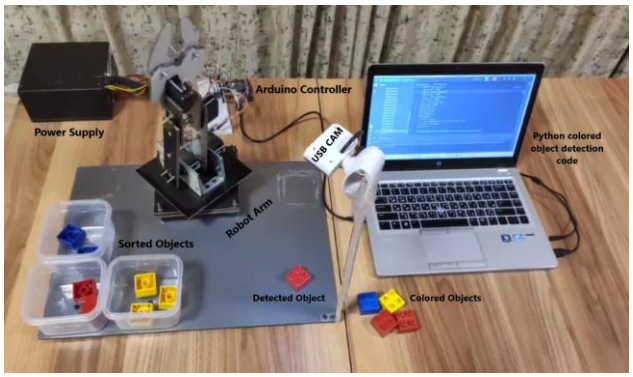

Figure 2: Final prototype of system.

III. COMPUTER VISION TECHNOLOGY

Computer vision technology is a branch of artificial intelligence that enables computers to interpret and understand the visual world. It focuses on developing algorithms and techniques that allow machines to extract, analyze, and interpret information from digital images or videos [15], [16].

A. IMAGE PROCESSING USED A CAMERA SENSOR

The main function of the camera sensor in this setup is to capture photos and videos and display them on computer for further image processing analysis. While any camera can technically serve the purpose here, opting for a high-quality camera is advisable to facilitate efficient data extraction from the captured images. The camera is connected to the computer via a USB port. Controlling the time period for opening and closing the attached camera, as well as capturing photos and videos, has been done through the Python programming language. Python was selected because it is the language commonly employed for such tasks, offering a balance of simplicity, flexibility, and robustness that aligns well with the work's requirements. Beside a specific code that is written to open and close the camera, it is also necessary to use Library (OpenCV) when opening the camera and capturing photos and videos using the Python programming language because one of the functions of this library is to capture images and videos from the camera, Figure 3 represents a captured image from a camera using Python code that can be written as following:

```
import cv2
camera = cv2.VidioCapture(0)
while True:
   ret , video = camera.read( )
   cv2.imshow(winname: 'mmm' , vedio)
   if cv2.waitKey(1) & 0xff == ord('q'):
      break
camera.release( )
cv2.destroyWindow( )
```

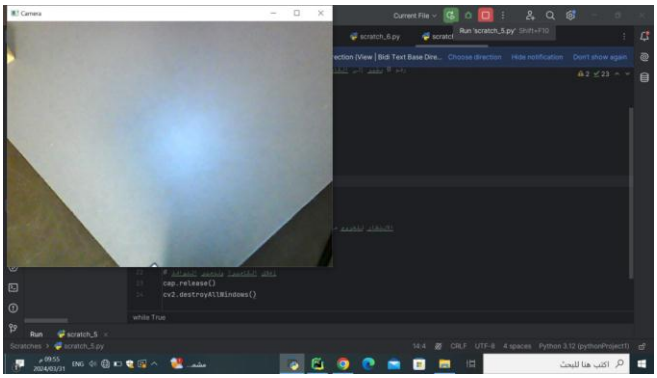

Figure 3: Image captured from camera using Python code.

To determine the center of a video frame or image captured using the Python programming language, these commands has been used as shown in the following text as:

```
import cv2
cap = cv2.VedioCapture(1)
while True:
   ret , frame = cap.read( )
   if not ret:
      break
height , width, = frame.shape
center_x = width//2
center_y = hieght//2
cv2.circle(frame, (center_x, center_y), 10, (0, 255, 0), 2)
cv2.imshow('Camera', frame)
if cv2.waitKey(1) == 27:
   break
cap.release( )
cv2.destroyAllwindows( )
```

The last python code working on opening the camera and continuously reads frames from it. Then The code functionality determines the center of the camera frame by

calculating the width and height of the frame and dividing them by 2. The code draws a green circle at the center of each frame using the OpenCV method that called cv2.circle as shown in Figure 4. The modified frame with the circle is displayed on the screen using cv2.imshow.

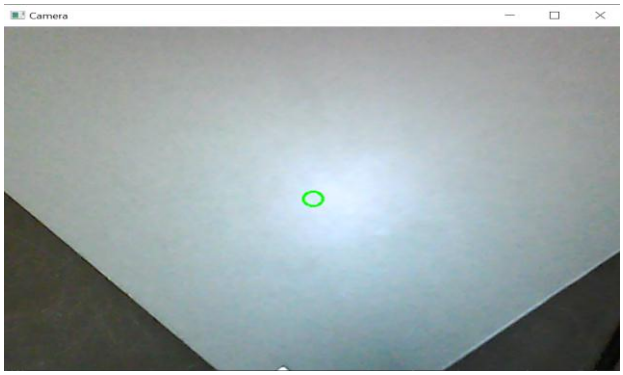

Figure 4: Center of the camera.

B. *SORT AND DETECTED COLORS USING THE PYTHON PROGRAMMING LANGUAGE:*

To sort colors using Python programming language, firstly capturing needed image from the camera in order to process it. This requires tools from a library that handles image and video processing, such as (OpenCV).

Additionally, it is needed to specify the color ranges for the image representation. This can be done using a library that is called (NumPy), where colors are separated, and the image is converted into a color space using RGB (red, green, blue) technology.

C. *RGB MODEL FOR SORTING AND DETERMINING COLORS:*

In this work, Python language was utilized which classifies colors in the photo center using RGB technology that uses three basic channels: red, green and blue to represent colors.

In the RGB model, the value for each channel is set between 0 and 255. Where 0 represents the absence of a certain amount of color, 255 represents the maximum possible existence of the color. On this basis, the designed Python program enables to determine the image center, then analyzing the color of an object in the center to its three main components (red, blue, green) which represents three basic object colors in a form of a matrix, which showing the colors probability value to each color inside matrix, in order to classify and determine object color.

The minimum and maximum values for each targeted object color are sorted and calculated with the help of Python program, it is possible to use the following Python code syntax to determine the basic main object colors as:

```
Import cv2
def capture_color( )
   cap = cv2.VedioCapture(1)
   color_values = [ ]
   while True:
      ret , frame = cap.read( )
      cv2.imshow( winname : 'Camera' , frame)
      color = frame(frame.shape[0] // 2, frame.shape[1] // 2]
      color_values.append(color.tolist())
      if cv2.waitKey(1) == 27:
         break
cv2.destroyAllwindows( )
cap.release( )
for color in color_values:
   print(color)
capture_color( )
```

The last code reads the object color values placed at the center of the camera frame and extracts these values into multiple arrays as shown in Figure 5. Each array represents the color values for the three primary colors, the minimum value array can be extracted by taking the minimum value for each of the three primary colors in the arrays extracted from the code. Similarly, the maximum value array can be extracted by taking the maximum value for each of the three primary colors.

D. *SORT THE OBJECTS COLOR AT THE CENTER OF A CAMERA USING PYTHON PROGRAMMING LANGUAGE:*

Firstly, the camera captures the image of an object as shown in Figure 5, so that the image is processed programmatically. The Python code determines the center of the image frame where the object was placed, and by sorting the object color value to its three primary colors (red, blue, and green) in order to identify its exact color.

Using arrays of upper and lower limits for each color, the actual color presented in the center of the image frame is determined.

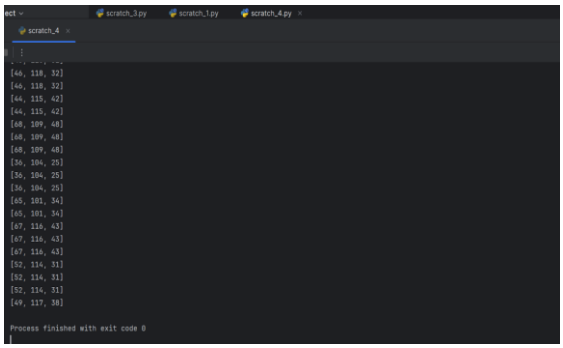

Figure 5: Color arrays extracted from the center of the image.

In OpenCV library, the color channels are ordered as blue (B), green (G), and red (R), which is the reverse order compared to the usual RGB representation. Therefore, when working with color representations in OpenCV arrays, it is important to remember the reversed order of the color channels. For example, for the red color, the typical values for the lower and upper bounds would be as follows: Lower red array = [0 0 170], upper red array = [50 50 255], and for green, the typical values for the lower and upper bounds would be as follows: lower green array = [0 170 0], upper green array = [50 255 50], and for blue, the typical values for the lower and upper bounds would be as follows: lower blue array = [170 0 0], upper blue array = [255 50 50].

These are the default values for the three primary colors (red, green, blue). In ideal conditions, the color values should range between these values.

An experiment has been done on object color detection in both normal and abnormal light conditions in order to check whether the usb camera able to recognize and detect object color or not in different light conditions.

Figure 6, Figure 7 and Figure 8 show a red, blue, green objects in front of usb camera under both ideal and non-ideal light condition to investigate the ability of the system to distinguish the color of each object that located in front of camera.

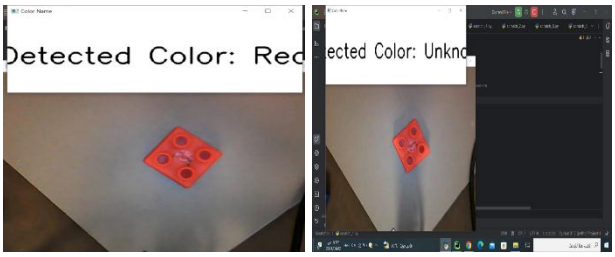

Figure 6: Detection of red under ideal and non-ideal lighting conditions.

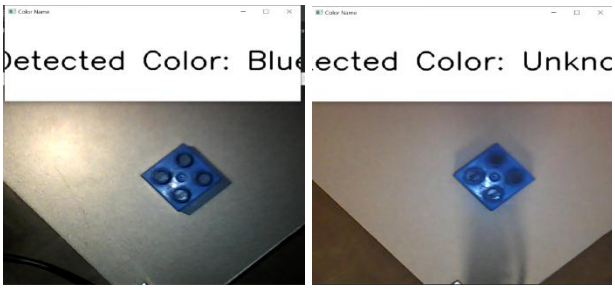

Figure 7: Detection of blue under ideal and non-ideal lighting conditions.

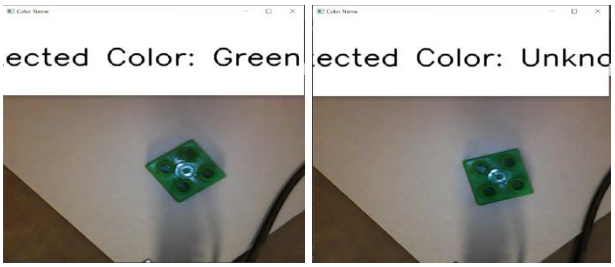

Figure 8: Detection of green under ideal and non-ideal lighting conditions.

Table (1) illustrates the color detection rates when taking color values under ideal lighting conditions.

Task	Red	Green	Blue
task 1	✔	✔	✔
task 2	✔	✖	✔
task 3	✔	✔	✖
task 4	✖	✔	✔
task 5	✔	✖	✖
task 6	✖	✔	✖
task 7	✔	✖	✔
task 8	✔	✔	✖
task 9	✖	✔	✔
task 10	✖	✔	✖
Success	6	7	5
Failure	4	3	5
Success (%)	60%	70%	50%
Average=60%			

When the lighting conditions change, the accuracy of color detection varies. See Figure 6, 7, and 8. Table (1) shows that the color detection accuracy percentage is 60%, which is considered weak, as the error rate reaches 40%, which is a significant error percentage. It also demonstrates that the best color detection accuracy is achieved under normal lighting conditions.

Since the error rate is high when the lighting conditions change, it is necessary to adjust the default color values and make them as suitable as possible for all lighting conditions. In other words, it is necessary to modify the values and make them adaptable to changes in lighting conditions.

Table (2) shows that the color detection accuracy is 93.33%, which is considered good, with an error rate of 6.67%, which is a small error percentage.

Table 2 demonstrates the color detection rates when taking color values under varying lighting conditions to be as adaptable as possible to all lighting variations.

Task	Red	green	Blue
task 1	✔	✔	✔
task 2	✔	✖	✔
task 3	✔	✔	✔

task 4	✖	✔	✔
task 5	✔	✔	✔
task 6	✔	✔	✔
task 7	✔	✔	✔
task 8	✔	✔	✔
task 9	✔	✔	✔
task 10	✔	✔	✔
Success	9	**9**	**10**
Failure	1	**1**	**0**
Success (%)	90%	**90%**	**100%**
Average = 93.33%			

IV. ROBOT ARM CONTROLLED BY SORTING THE COLORS USING COMPUTER VISION

A. Robot Arm Controlled by Sorting the Colors using a Photo:

The movement of the robotic arm is controlled using color sorting through computer vision. In this method, an image is captured using a camera and then processed using computer vision techniques in Python programming language.

An object is placed at the center of the frame indicated by the camera. Upon capturing the image, the camera's center is determined, and the color values at the center of the image are extracted. Color values for specific colors are stored in Python programming language. Then, the color values of the desired colors are compared with the color values at the center of the image. If the color values at the center of the image match one of the stored color values, the color of the camera's center is identified as that color. If the color at the center of the image does not match any of the stored colors, it is considered as another color.

After completing this stage, the color of the camera frame center will be determined, either as one of the pre-stored colors or as another color.

Once the color is identified, a controller (an Arduino Uno here) is used to control the servo motor angles, which in turn moves the arm to the desired position, based on the detected color, the controller determines specific angles for the servo motors. Depending on the angles of the motors, the arm will move to a specific point. In this work, three colors were used, namely red, blue, and green, as they are the primary colors. The color values for these colors were determined based on multiple tests conducted under different lighting conditions to find the best possible values for these colors and to maximize the likelihood of accurately sorting them. Each color has a minimum and maximum value array. The color values for these colors were determined as follows: lower red array = [0 0 40], upper red array = [120 120 255], lower green array = [0 20 0], upper green array = [45 90 30], lower blue array = [50 10 0], upper blue array = [255 80 30].

To ensure the accuracy of these color values, a color calibration test can be performed at the testing location. Upon capturing the image, determining the camera's center, and extracting the color values at the center, these values will be compared with the color values of the three colors. If there is a match with one of these values, the corresponding color will be identified. If there is no match, the color will be identified as "another color."

There are four possible outcomes when comparing colors: (red, green, blue, or another color), based on these four cases, the arm will move accordingly. If the object in the center of the image is red, the arm will grasp the object and place it in the designated area for red. If it is green, the arm will grasp the object and place it in the designated area for green. Similarly, if it is blue, the arm will grasp the object and place it in the designated area for blue, If the detected color is different from that, the arm will not move, meaning it will remain in its initial position.

The angles for the servo motors to reach the camera's center have been predetermined. These angles will be used to approach and capture the object located at the camera's center. Additionally, the necessary angles to reach the designated positions for the three colors have also been predetermined. After capturing the object and based on its color, it will be placed in the corresponding designated area. Afterwards, the arm will return to its initial position, can calibrate the designated position for placing the object at the center of the camera and determine the specific positions based on their colors using the user interface as previously explained to obtain the required angles for reaching these positions. Figure 9 shows the block diagram to control the robot arm by sorting the colors by taking a photo.

B. Control the Robot Arm by Sorting the Colors while Taking a Video:

The movement of the robotic arm is controlled using color sorting through computer vision. In this method, a video is captured using a camera and then processed using computer vision techniques in Python programming language.

The steps for controlling the robotic arm for color sorting by capturing a video are similar to the steps for controlling the robotic arm by capturing an image, with some differences. In this method, a video will be captured instead of a single image. The process of determining the center of the frame remains the same in both cases. As for the color values, they are the same for the three primary

colors (red, green, and blue) in both scenarios.

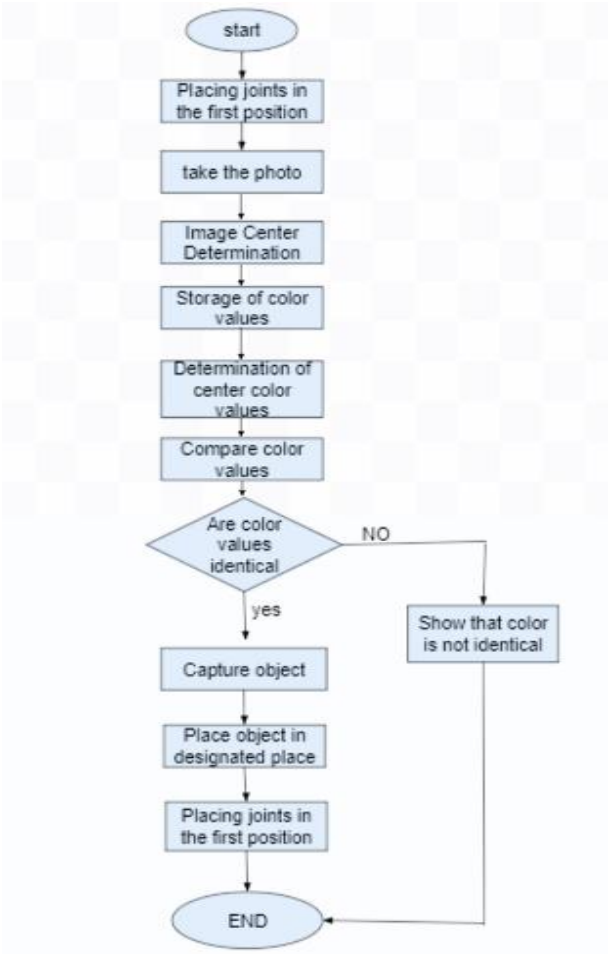

Figure 9: Flowchart to control the robot arm by sorting the colors by taking a photo.

During video capturing, the color values at the center are continuously analyzed until they align with the stored color values for the three predefined colors. This process involves reading the color at the center and comparing it against the predefined color values. Once a match is found, the identified color is then determined. However, if it doesn't match any of the three colors, another color value will be read and compared until a match is found, once the color is identified, the subsequent steps will follow the same process as capturing an image.

By using the microcontroller board (Arduino Uno), the angles required for the servo motors of the robotic arm can be determined to move the arm and capture the object, placing it in the appropriate position. Additionally, there are three color cases (red, blue, and green) when they are matched. There are no other cases because the color values will be compared until one of these three colors is obtained. If none of these colors are achieved, the video will continue to operate until one of these colors is obtained. Once one of these colors is reached, the color will be identified, and the video will be stopped.

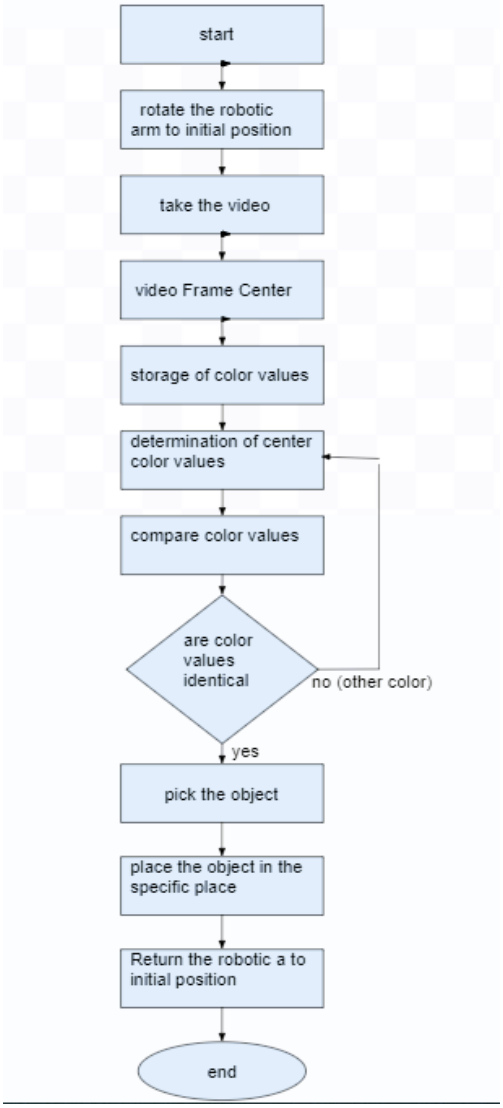

Figure 10: Flowchart to control the robot arm by sorting the colors by taking a video.

V. CONCLUSION

In this study, a 3DOF robotic arm for color sorting and object identification using computer vision was successfully designed and implemented. The final prototype of our work utilized servo motors and an Arduino microcontroller, along with image processing and a camera sensor for color detection. The color sorting was achieved using the RGB model, and the robotic arm was controlled using computer vision techniques utilizing Python programming language with the OpenCV and NumPy libraries.

The implementation of the 3DOF robotic arm for color sorting has proven to be effective. The utilization of computer vision techniques, along with the integration of image processing and color detection, has contributed to the accurate sorting of objects based on their color. Additionally, the use of servo motors and an Arduino microcontroller has provided precise control of the robotic arm.

Overall, the successful implementation of the 3DOF robotic arm for color sorting using computer vision demonstrates its potential for automation and efficiency in various industries, such as manufacturing and logistics.

The ability to accurately and autonomously sort objects based on their color has the potential to streamline processes and increase productivity. Furthermore, the integration of computer vision and robotics showcases the advancements in technology and automation, paving the way for future innovation and development.

VI. ACKNOWLEGMENT

The authors would like to thank Ausama H. Ahmed for his help in proofreading the manuscript.

REFERENCES

[1] Ausama Hadi Ahmed and Othman E. Aburas. "A Guide to Implement a Two Wheeled Robot Using Pole-Placement on Arduino." The International Journal of Engineering and Information Technology, Misurata University ,VOL.6, NO.2,2020.

[2] Ausama H. Ahmed and Othman E. Aburas, et al. "Modeling and Control of Two Wheels Robot Using Linear Quadratic Regulators." 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA. IEEE, 2021.

[3] Abdullah-Al-Noman, Md, et al. "Computer vision-based robotic arm for object color, shape, and size detection." Journal of Robotics and Control (JRC) 3.2 (2022): 180-186.

[4] Intisar, Muhatasim, et al. "Computer Vision Based Robotic Arm Controlled Using Interactive GUI." Intelligent Automation & Soft Computing 27.2 (2021).

[5] Fadhil, A. T., K. A. Abbar, and A. M. Qusay. "Computer Vision-Based System for Classification and Sorting Color Objects." IOP Conference Series: Materials Science and Engineering. Vol. 745. No. 1. IOP Publishing, 2020.

[6] Vo, Cong Duy, Duy Anh Dang, and Phuong Hoai Le. "Development of Multi-Robotic Arm System for Sorting System Using Computer Vision." Journal of Robotics and Control (JRC) 3.5 (2022): 690-698.

[7] Chakole, Saurabh, and Neema Ukani. "Low-cost vision system for pick and place application using camera and ABB industrial robot." 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2020.

[8] ZeChern, Ong. "Towards Industry 4.0: Color-based object sorting using a robot arm and real-time object detection." Industrial Management Advances 1.1 (2023).

[9] Tejaswini, S., M. P. Spoorthi, and B. S. Sandeep. "Real time object sorting with 3D parameter using image processing and robotic ARM system." SN Computer Science 2.3 (2021): 162.

[10] Cong, Vo Duy, et al. "Design and development of robot arm system for classification and sorting using machine vision." FME Transactions 50.1 (2022): 181-181.

[11] Fernandes, Lennon, and B. R. Shivakumar. "Identification and Sorting of Objects based on Shape and Colour using robotic arm." 2020 Fourth International Conference on Inventive Systems and Control (ICISC). IEEE, 2020.

[12] Farman, Madiha, et al. "Design of a three degrees of freedom robotic arm." International Journal of Computer Applications 179.37 (2018): 12-17.

[13] Manolescu, Denis, and Emanuele Lindo Secco. "Design of a 3-DOF Robotic Arm and implementation of DH Forward Kinematics." Congress on Intelligent Systems. Singapore: Springer Nature Singapore, 2022.

[14] Atique, Md Moin Uddin, and Md Atiqur Rahman Ahad. "Inverse Kinematics solution for a 3DOF robotic structure using Denavit-Hartenberg Convention." 2014 international conference on informatics, electronics & vision (ICIEV). IEEE, 2014.

[15] Bengtson, Stefan Hein, et al. "A review of computer vision for semi-autonomous control of assistive robotic manipulators (ARMs)." Disability and Rehabilitation: Assistive Technology 15.7 (2020): 731-745.