Computer Vision-based Robotic Arm for Object Color, Shape, and Size Detection

Md. Abdullah-Al-Noman [1], Anika Nawer Eva [2], Tabassum Binth Yeahyea [3], Riasat Khan [4*]
[1,2,3,4] Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh
Email: [1] abdullha.noman@northsouth.edu, [2] anika.nawer@northsouth.edu, [3] tabassum.yeahyea@northsouth.edu, [4] riasat.khan@northsouth.edu
*Corresponding Author

***Abstract*—Various aspects of the human workplace have been influenced by robotics due to its precision and accessibility. Nowadays, industrial activities have become more automated, increasing efficiency while reducing the production time, human labor, and risks involved. With time, electronic technology has advanced, and the ultimate goal of such technological advances is to make robotic systems as human-like as possible. As a result of this blessing of technological advances, robots will perform jobs far more efficiently than humans in challenging situations. In this paper, an automatic computer vision-based robotic gripper has been built that can select and arrange objects to complete various tasks. This study utilizes the image processing methodology of the PixyCMU camera sensor to distinguish multiple objects according to their distinct colors (red, yellow, and green). Next, a preprogrammed command is generated in the robotic arm to pick the item employing Arduino Mega and four MG996R servo motors. Finally, the device releases the object according to its color behind the fixed positions of the robotic arm to a specific place. The proposed system can also detect objects' geometrical shapes (circle, triangle, square, rectangle, pentagon, and star) and sizes (large, medium, and small) by utilizing OpenCV image processing libraries in Python language. Empirical results demonstrate that the designed robotic arm detects colored objects with 80% accuracy. It performs an excellent size and shapes recognition precision in real-time with 100% accuracy.**

Keywords—Arduino Mega; Color Sorting; OpenCV; PixyCMU; Robotic Arm; Servo Motor; Shape Detection.

I. INTRODUCTION

With the continued advancement of science and technology, human living standards have improved worldwide [19]. It has also increased life anticipation and economic growth [15]. Consequently, with such development, peoples' work habits have changed drastically globally [16]. The uses of robots in several fields, such as industrial production, military combat, and many other applications in recent years, have received extensive attention [1, 20]. To minimize the labor cost, production time, and possible dangers in different types of risky tasks, the usage of robots are also encouraged [2]. For instance, robotic arms are now often employed in many cases to execute jobs that humans cannot or will not undertake [17]. Besides, to assist the elderly or the disabled, computer vision-based robotic arms have been utilized in recent times [3]. Computer vision is a recent advancement in the field of artificial intelligence (AI) field, and it is extensively utilized in industrial automation, detection, object identification, automated technology, and robotics [21, 22]. Picking and placing products flawlessly with consistent motions are essential in an industrial setting [4]. Additional control systems, joysticks or controllers, and even manual programming are now used to maneuver the sophisticated robotic arm [5]. A robotic system may offer a range of sensors such as temperature, radiation, color, weight, etc. [23]. However, there is an increasing tendency to replace all of these sensors with a single-lens camera and computer vision approaches to perform the tasks independently with enhanced precision [24]. In this technique, a novel control method is employed using computer vision to drive a robotic arm [25]. Thanks to computer vision and artificial intelligence techniques, a cost-efficient, user-friendly, and the highly efficient robotic arm can be built using such techniques [26].

Computer vision and artificial intelligence have been successfully utilized to operate complex robotic systems in many works. For instance, in a recent work [6], the authors designed an automated robotic arm-based assistance device by employing simple stereo matching and q-learning optimization techniques. The proposed device can perform five degrees of operation in a single instance and detect any object with the help of stereo vision. The system also keeps track of an object's parameters. The stereo camera can identify the RGB color. In this work, the Q-learning framework [29] is employed to control the position of the robot's arm. Finally, the paper showed experiments to identify an object's stereo vision and feature point. The downfall of the study is that it did not specify the exact distance between the objects.

In [7], Daniel Kruse et al. presents a new approach using a two-armed industrial robot for bimanual hybrid motion/force control and visual serving. This work aims to develop a telerobotic system that can manipulate a grasped item with both arms. The experimental testbed is a dual-arm industrial robot with 15 degrees of freedom, a camera, a torque/force sensor, and a rubber contact pad on each wrist. The operator orders are sent by gestures using a Microsoft Kinect sensor. Seven processes are running on three PCs, and they are connected via a local hub using the protocol called TCP/IP. In addition, the authors added global planning approaches to manage local equilibrium and more complex redundancy resolution.

In this recent work [8], the authors developed a fusion-based manipulation of sensory-motor and a robotic bionic arm, a control strategy of grasping for an automated hand-eye system. The proposed device was developed by vision serving, motion optimization, sEMG, and a hybrid force approach. The arm was designed by joining different motors with an Arduino microcontroller and was controlled with the help of a Point Grey Bumblebee stereo camera. The hand was working by a matrix calculation using epipolar geometry [30]. In this paper, different types of data like joint, relaxation, and grasping time graphs are collected with the help of 3 fingers force and 20 frames per second camera capturing the images.

In [9], the authors explore the concept of building a robot that can track colored objects. The brightly colored object has a basic design. A wheeled robot has been designed by utilizing a motor and wheels. A Pixy 2 based wheeled robot with sensors is utilized to detect objects with different colors. This robot moves in the same direction as the thing it's attached to is moving. When it comes to mobility, this robot has two wheels, i.e., one on each side (right and left). The Arduino Uno board's microprocessor is in charge of controlling the movement of the robot.

In this recent work [10], the authors designed a self-feeding assistive robotic arm of 7 degrees of freedom for people with severe disabilities. This system used a robotic arm simulator to obtain the motion of the robotic arm. In this process, inverse kinematics equations [31] were used to control the robot's arm position. The paper showed that the robot could successfully transfer the food to the appropriate location of its user. The authors conducted further experiments on adding a web camera that is attached to the end-effector of the robotic arm to monitor whether foods dripped from its edge or not.

The study in [11] focuses on implementing a computer vision-based robotic arm that will detect various objects by color sorting, grab that object, and place it in a specific place. The proposed system will also measure the length, width, and distance between objects and identify their object's position by using the computer vision technique. The arm will do the movement using a servo motor controlled by an Arduino Uno microcontroller.

This paper implements PixyCam CMU to build a computer vision-based robotic arm by implementing automatic color, shape, and size detection of objects. This study makes the following key contributions:

- An automatic color detection system is executed with a Pixycam CMU. The proposed robotic arm can pick up and place the color-sorted items in a precise location employing Arduino Mega and MG996R servo motors.
- The device can recognize items in terms of their various shapes and sizes, which has been executed by the OpenCV functions.
- The designed system's performance is assessed in terms of detection accuracy and real-time detection rate. Finally, the implemented device's characteristics and cost are compared to other existing works.

To the best of our knowledge, a computer vision-based robotic arm employing PixyCam CMU for object color, shape, and size recognition methodologies is implemented on a sophisticated embedded device such as the Arduino Mega 2560 and Pixycam CMU for the first time in this work.

The paper is organized into three sections. Section I will review the method and components used to design and implement the proposed system. The outcome of the real-time applications of the system and the challenges faced during the experiments are presented in Section II. Finally, Section III will discuss the conclusion and possible extension of the proposed system.

II. Proposed System

In this paper, an automatic robotic arm has been built that can sort various objects based on their colors, shape, and sizes. The proposed device can pick a selected object and place it in a specific place according to the desired distance. The subsequent paragraphs discuss the software and hardware elements utilized to design the proposed robotic device.

A. Software Tools

Arduino 1.8.15: Arduino is a cross-platform IDE that allows programmers to modify code and then upload the code to a board where it can be tested [27]. C and C++ are two conventional languages for Arduino. As a result, the program primarily targets developers and coders who work in those two languages. In this work, we use this software to build the code, and with the help of the software, we can control the robotic arm.

Pixy Mon 2.0.9: Pixy Mon is a program that lets us specify the output port, set up the Pixy device, and control color signatures [28]. We can connect a USB connection to Pixy's back and run Pixy Mon to observe what Pixy sees when connected to the Arduino or other microcontrollers. We use this software to detect the colors of the object with the help of Pixy cam.

Python 3.9: Python is a programming language that allows connecting the systems faster and more efficiently. We use this program to detect the geometrical shape of an object. With the help of easy Python language coding, we can load the image and calculate the geometrical shape of an object.

Anaconda: Anaconda is a Python and R programming language distribution that facilitates management systems to ensure installation in computer science. In this research, we use an Anaconda prompt to run the code and display the result for the shape detection of an object.

B. Hardware Tools

Arduino Mega 2560 Rev3: The Arduino Mega 2560 R3 is a microcontroller board. It is based on the ATmega2560. It contains 54 digital I/O pins, 4 UARTs (hardware serial ports), 16 analog inputs, a USB connection, a power jack, a reset button, an ICSP header, and a 16 MHz crystal oscillator [12]. The use of this Arduino microcontroller is to control the robotic arm through Arduino code.

PixyCMU Cam5 V1: The Pixy CMUcam5 is an Arduino and Raspberry Pi compatible camera sensor [18]. This camera sensor successfully loads the image where many other image sensors have failed. It is often challenging to utilize this sensor with a simple Arduino board-type CPU without getting saturated. We use this component for the color detection of a specific object.

Tower Pro MG996R: The Tower Pro MG996R is a servo motor consisting of metal gear. It has a maximum of 11 kg/cm stall torque [13]. It can rotate from 0 (zero) to 180 degrees. This high-torque motor depends on the PWM wave's duty cycle given to its signal pin, just as other RC servos. In this work, we used this motor to build the arm, grab the object, and place it in a destination folder.

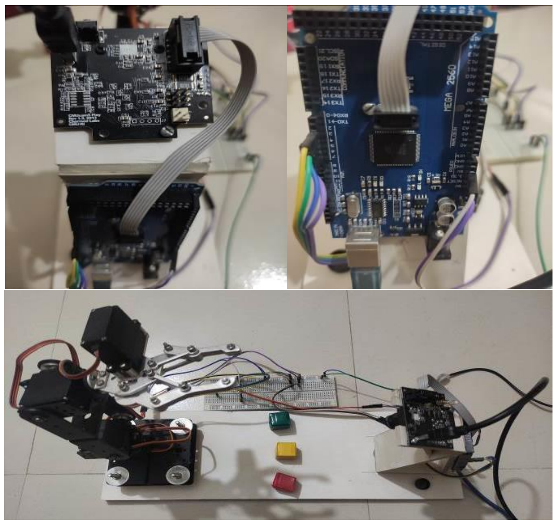

Fig. 1. Total hardware design of the proposed system.

Fig. 1 demonstrates the complete hardware design of the proposed system. Initially, the Arduino Mega microcontroller is connected with PixyCMU and the servo motor of MG996R. The four servo motors with the robotics kits (screws and metal angels to connect the servo motors) build the structure of the robotic arm. The system power supply comes from a 9-Volt DC adapter. With the help of this adapter, the robotic arm gets the power to perform the desired task smoothly. The total circuit connection is made on the breadboard. With the laptop connection, the robotic arm finally starts to do work. However, this proposed system in Fig. 1 approximately costs about 12,840 BDT ($150). It is worth mentioning that this total cost does not include the laptop’s price.

Fig. 2 demonstrates the working procedure of the proposed system for colored object detection, picking up and placing it in the desired place. According to Fig. 2, the robotic arm moves toward different items and selects the object according to its color, i.e., red object. The PixyCMU camera sensor does the color detection of the proposed system. The PixyCMU camera sensor employs a filtering technique implemented by distinct RBW colors and a region-growing approach to recognize different objects [32]. In this method, the color difference value in each direction is calculated as:

$$\Delta C(x,y) = \begin{cases} \widetilde{W}(x,y) - R(x,y): W\ interpolation \\ \tilde{R}(x,y) - W(x,y): R\ interpolation \end{cases} \quad (1)$$

where the pixel location indices are denoted as (x,y). $\widetilde{W}$ and $\tilde{R}$ denote the temporary detected colors in each direction. According to the region growing algorithm, an individual area is considered coherent if all the pixels are coherent according to the similarity principle [33]. Initially, we begin the process with the seed pixel and compare its adjacent pixels with the preset threshold as:

$$|Z_{max} - Z_{min}| \leq threshold \quad (2)$$

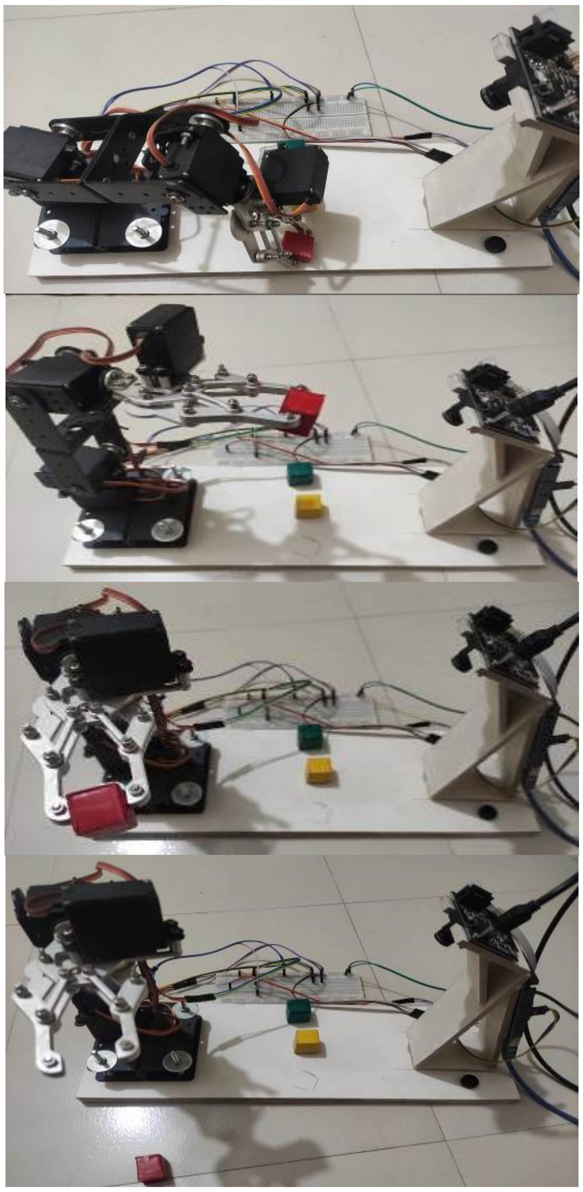

Fig. 2. Working process of the proposed system.

In (2), Z_{max} and Z_{min} illustrate the maximum and minimum pixel intensity values in the considered region. Then the arm bends down to catch the object and picks it up by its gripper. Next, the arm releases the object to a fixed place where the code specifics for the different colors. After releasing one object according to its color, the arm's function automatically detects more items if any other exists on the Pixy cam screen. If it exists, the robotic arm continues the process and releases the object to a different place. To quickly understand the color sorting process, we defined our program in color sorting detection with a color sorting release point.

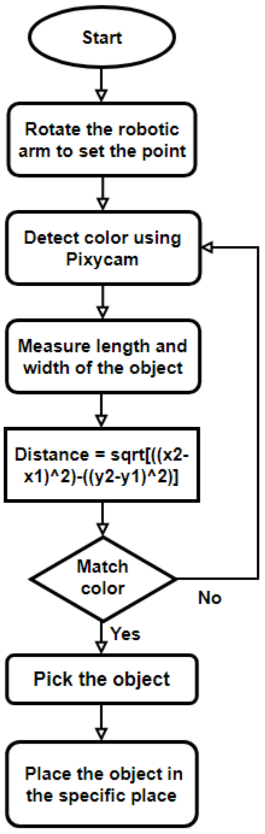
Start
Rotate the robotic arm to set the point
Detect color using Pixycam
Measure length and width of the object
Distance = sqrt[((x2-x1)^2)-((y2-y1)^2)]
Match color
No
Yes
Pick the object
Place the object in the specific place

Fig. 3. Working flow chart of the robotic arm's color sorting operation.

Fig. 3 illustrates the functional flowchart of the proposed automated device for the color sorting process. From the figure, we can understand how the project is working sequentially. Firstly, when the project is started, the robotic arm will wait for instructions from its users. Moreover, this will decide the PixyCMU camera's ability to detect the object's color. If the color matches the previously saved color in the Pixy cam's memory, it will receive a detection instruction from the Arduino Mega to the robotic arm's motor. Then the robotic arm will pick up the object and place it in the desired place according to the color-sorting method. Then the Pixy cam rechecks if any other colors object is present or not. If so, it redoes the same procedure and places the object in other fixed places for that specified color.

To measure the distance, we have used the equation of the Pythagorean theorem, which is expressed as:

$$d = \sqrt{((x_2 - x_1)^{\,2} + (y_2 - y_1)^{\,2})} \tag{3}$$

where x_1 and y_1 are the coordinates of the first point and x_2 and y_2 are the coordinates of the second point [14]. In this experiment, the x_1 and y_1 points are the positions of the Pixy cam and the x_2 and y_2 points are the positions of the specific object. Finally, we measured all the points and then implemented these points in (1) to find the distance of an object. After measuring the distance between these two points, the function displays the approximate distance in millimeters (mm).

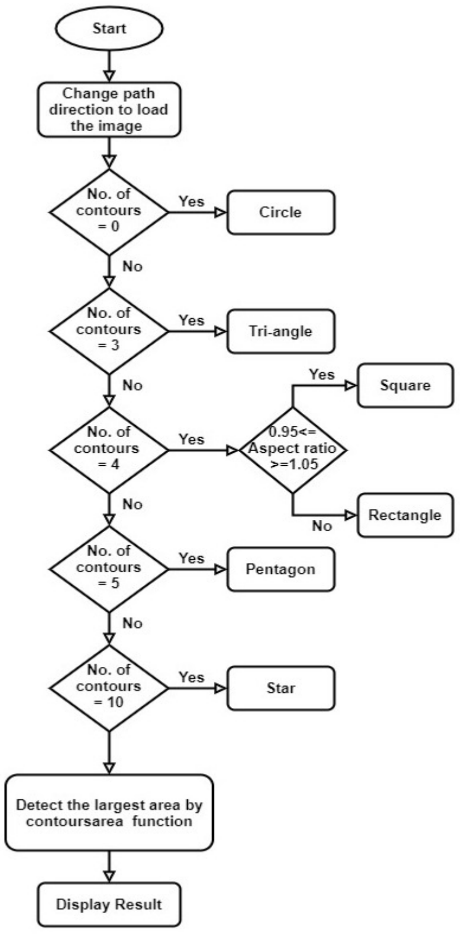

Fig. 4. Working flow chart of the geometrical shape detection.

To implement the object shape and size detection, we have used the OpenCV libraries in the Python programming language and run through the Anaconda prompt in this work. Fig. 4 demonstrates the shape detection part. First, we have to change the path direction of the folder of Python coding for shape detection. Next, we run the program, and the result will be displayed in the image. Here, the contour function is used for the shape detection part of this project. The contour function detects the curve joining points of all continuous points with the same color or intensity. If the object has three contours, it will be a triangle. In the same way, the star, pentagon, or other shapes are detected along with the contour point. The area's aspect ratio is calculated for the square and the rectangular regions to differentiate them from each other, as they have the same number of contours points of four. Otherwise, it will be a circle. We can detect the geometrical shape through the image processing system.

Finally, we have used the contour function in the OpenCV library to detect the object's size. To do that, we have to measure the area of the object's shape. Hence, we have used the contour function to determine the total area of the boundary of the object's shape. This specific function measures how much area it occupies and compares it with the other object's shape. Then the function automatically detects the largest area of different objects, and finally, it shows the largest and smallest spaces of the object's shape.

III. Results and Discussion

In this paper, a computer vision-based robotic arm has been designed. The proposed system is able to automatic color sorting, distance measurement, and shape and size detection of an object. To design the hardware device, four MG996R high-torque servo motors are connected with Arduino Mega to pick the object and place it in its desired place. A PixyCMU smart sensor is attached to the Arduino to detect the color of an object. OpenCV libraries in Python have been used for object shape and size identification. With the help of a PC connected to the proposed device, the robotic arm successfully detects the object's color, shape, and size picks the object, and places it in a specific place.

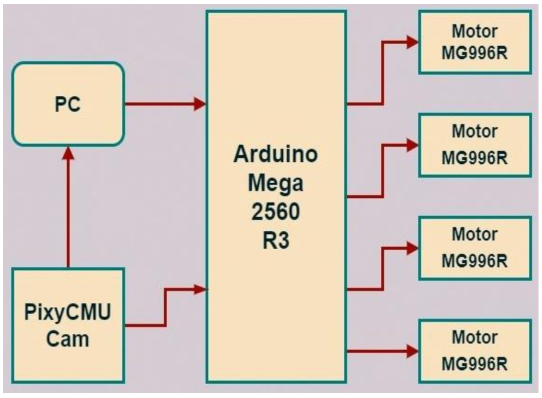

Fig. 5. Working flowchart of the proposed system.

Here in Fig. 5, we can observe the block diagram of the proposed robotic arm. It shows us the connection between the hardware components that have been used in this project. The wire connection between the Pixy cam, Arduino Mega board, and servo motor makes the total circuit connection of the robotic arm.

TABLE I. COLOR MATCHING AND SORTING ACCURACY

Task	Red	Yellow	Green
Task 1	✓	✓	✓
Task 2	✓	●	✓
Task 3	✓	✓	●
Task 4	●	✓	✓
Task 5	✓	✓	✓
Task 6	✓	✓	●
Task 7	✓	●	✓
Task 8	✓	✓	✓
Task 9	✓	✓	✓
Task 10	✓	✓	●
Success	9	8	7
Failure	1	2	3
Success (%)	90%	80%	70%
Average = 80%			

We have repeatedly tested the color sorting part of the proposed system to check the accuracy of the color detection of the PixyCMU camera sensor. From Table I, we can observe that the PixyCMU cam has successfully detected the colored items (red, yellow, and green) with 80% accuracy though it is light-sensitive. This process is repeated ten times for each of the colored objects. The color detection accuracy of the device varies with the change in lighting conditions and the distance of the objects with the Pixy sensor. However, it has given the best performance to our project in the normal light. After the accurate color detection of the item, the robotic arm successfully picks the object and places it in the color sorting way. Finally, the robotic arm holds the items effectively, and it can pick and place an object with 90% accuracy.

TABLE II. DISTANCE MEASUREMENT ACCURACY RATE

Task	Attempts	Success	Failure	Success (%)
Red	10	9	1	90%
Yellow	10	8	2	80%
Green	10	7	3	70%
Overall	30	24	6	80%

Table II presents the accuracy rate for distance measurement of the implemented device after picking and placing an object. The color-sorted item is placed in a specific place after the robotic arm picks up an object. We use the rules of 2D distance measurement and try to test over 30 times manually. The result table shows the accuracy rate of objects arrangements for the device is 80% because when the robot arm fails to detect the color of an object, it remains in the same position. Consequently, that is counted as a failure turn, and it happens six times throughout the total of 30 detections.

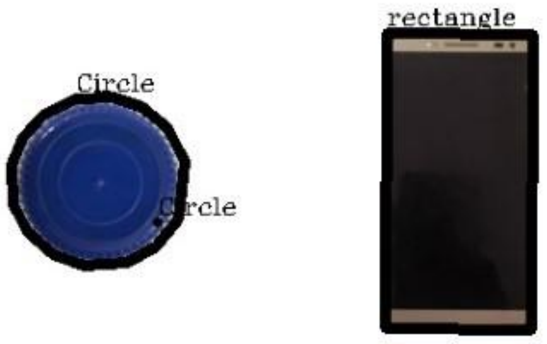

Fig. 6. Sample object shape detection output of the proposed system.

TABLE III. SHAPE DETECTION ACCURACY RESULTS

Shape	Attempts	Success	Failure	Success (%)
Circle	10	10	0	100%
Triangle	10	10	0	100%
Rectangle	10	10	0	100%
Square	10	10	0	100%
Pentagon	10	10	0	100%
Overall	50	50	0	100%

Next, Fig. 6 illustrates a sample object shape detection output of the proposed system. Table III demonstrates the accuracy results for geometrical shape detection of an object. Here, we test the object shape identification tasks ten times for various shaped articles, i.e., circle, triangle, square, rectangle, and pentagon, and each time all the shapes are successfully detected. We have used the contour function to detect the geometrical shape. According to Table III, the accuracy rate for shape detection is 100%.

TABLE IV. SIZE DETECTION ACCURACY RATE

Size	Attempts	Success	Failure	Success (%)
Large	10	10	0	100%
Medium	10	10	0	100%
Small	10	10	0	100%
Overall	30	30	0	100%

Table IV presents the accuracy results of the size detection part of this work. The average accuracy rate for size detection is 100%. The OpenCV image processing function successfully measures the boundary area of different shapes and detects the largest and smallest shaped objects.

TABLE V. COMPARISON TABLE OF HARDWARE COMPONENTS

Ref.	Microprocessor	Camera Sensor	Servo Motor	Connect to Arm
[6]	Neural network	CCD	N/A	N/A
[7]	DX100 controller	Sony XCD-X710CR	15	Gesture control
[8]	Harmonic Transmission Drives	Bumblebee Vision System	5	SEMG Collector
[9]	Arduino Uno R3	Pixy Cam CMU 2	3	USB
[10]	Arduino Mega 2560	N/A	7	N/A
[11]	Arduino Uno	Webcam	4	USB
This work	Arduino Mega 2560 R3	Pixy Cam CMU	4	USB

Table V depicts various hardware components utilized in similar works of computer vision-based robotic systems. Diverse types of microprocessor systems have been employed in many works, e.g., Arduino, Raspberry Pi, DX100 controller, etc. According to Table V, this proposed device implements automatic object color, shape, and size detection with comparable low-cost, simple equipment.

TABLE VI. COMPARISON TABLE FOR SOFTWARE TOOLS

Ref.	Color Detect	Shape Detect	Distance Measure	Position
[6]	Stereo camera	N/A	Stereo matching	QSO inverse Kinect
[7]	ALVAR tags	VTT ALVAR	ALVAR tags	Kinect
[8]	Gray mode	N/A	Hall effect sensor	Kinect
[9]	Pixy sensor	N/A	Pre-defined	Kinect
[10]	N/A	N/A	N/A	Inverse Kinect
[11]	HSV	OpenCV contours	N/A	Kinect
Proposed	Pixy sensor	OpenCV contours	Pythagorean theorem	Preprogrammed positions

Table VI demonstrates the comparison table for software tools compared to this paper and other related works. This study performs object color, shape and size recognition, and distance measurement with more straightforward techniques.

IV. CONCLUSIONS

This paper exhibits the design and the development of a computer vision-based automated system for sorting colors as well as detection of shape and size. It can detect the color of an object utilizing a PixyCMU camera sensor and pick and place the item in a specific place employing Arduino Mega and servo motors controlled robotic arm. The performance of the proposed intelligent device has been tested through the detection of random colored objects. Also, we successfully measured the distance for the robotic arm to pick up the object. Moreover, the robotic arm detects an object's geometrical shape and size using OpenCV image processing functions implemented in Python. The proposed system is expected to reduce labor costs and increase productivity and efficiency in industries.

In the future, a better camera sensor can be used to detect a 3D object. Raspberry Pi or Jetson Nano can be employed to control the proposed device more efficiently. More advanced deep learning and neural network approaches can be utilized for object color, shape, and size detection in the future.

REFERENCES

[1] F. Rubio, F. Valero, and C. Llopis-Albert, “A review of mobile robots: Concepts, methods, theoretical framework, and applications,” International Journal of Advanced Robotic Systems, vol. 16, no. 2, pp. 1–22, 2019.

[2] J. Arents and M. Greitans, “Smart industrial robot control trends, challenges and opportunities within manufacturing,” Applied Sciences, vol. 12, no. 2, p. 937, 2022.

[3] O. Glaufe, O. Gladstone, E. Ciro, C. A. T. Carvalho, and L. José, “Development of robotic arm control system using computational vision,” IEEE Latin America Transactions, vol. 17, pp. 1259–1267, 2019.

[4] A. Kelly, B. Nagy, D. Stager, and R. Unnikrishnan, “Field and service applications - An infrastructure-free automated guided vehicle based on computer vision - An Effort to Make an Industrial Robot Vehicle that Can Operate without Supporting Infrastructure,” IEEE Robotics & Automation Magazine, vol. 14, pp. 24–34, 2007.

[5] C.-Y. Tsai, C.-C. Wong, C.-J. Yu, C.-C. Liu and T.-Y. Liu, “A hybrid switched reactive-based visual servo control of 5-DOF robot manipulators for pick-and-place tasks,” IEEE Systems Journal, vol. 9, pp. 119–130, 2015.

[6] Y.-Z. Hsieh, “Robotic arm assistance system based on simple stereo matching and Q-learning optimization,” IEEE Sensors Journal, vol. 20, pp. 10945–10954, 2020.

[7] D. Kruse, J. T. Wen, and R. J. Radke, “A sensor-based dual-arm tele-robotic system,” IEEE Transactions on Automation Science and Engineering, vol. 12, pp. 4–18, 2015.

[8] Y. Hu, “Development of sensory-motor fusion-based manipulation and grasping control for a robotic hand-eye system,” IEEE Transaction on Systems, Man and Cybernetics: Systems, vol. 47, pp. 1169–1180, 2017.

[9] S. D. Perkasa, P. Megantoro and H. A. Winarno, “Implementation of a camera sensor pixy 2 CMUcam5 to a two wheeled robot to follow colored object,” Journal of Robotics and Control, vol. 2, pp. 496–501, 2021.

[10] S. Gushi, Y. Shimabukuro, and H. Higa, “A self-feeding assistive robotic arm for people with physical disabilities of the extremities,” International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), pp. 61–64, 2020.

[11] M. Intisar, M. M. Khan, M. R. Islam, and M. Masud, “Computer vision based robotic arm controlled using interactive GUI,” Intelligent Automation & Soft Computing, vol. 27, pp. 533–550, 2021.

[12] H. M. I. Salehin, Q. R. A. Joy, F. T. Z. Aparna, A. T. Ridwan and R. Khan, “Development of an IoT based smart baby monitoring system with face recognition,” IEEE World AI IoT Congress (AIIoT), pp. 292–296, 2021.

[13] A. A. Rafiq, W. N. Rohman, S. D. Riyanto, “Development of a simple and low-cost smartphone gimbal with MPU-6050 sensor,” Journal of Robotics and Control, vol. 1, pp. 136–140, 2020.

[14] L. Huang, G. Wu, W. Tang, and Y. Wu, “Obstacle distance measurement under varying illumination conditions based on

monocular vision using a cable inspection robot," IEEE Access, vol. 9, pp. 55955–55973, 2021.

[15] N. D. Rao and J. Min, "Living standards: material prerequisites for human wellbeing," Social Indicators Research, vol. 138, pp. 225–244, 2018.

[16] H. Kreinin and E. Aigner, "From "Decent work and economic growth" to "Sustainable work and economic degrowth": A new framework for SDG 8," Empirica, pp. 1–31, 2021.

[17] C. A. G. Gutiérrez, J. R. Reséndiz, J. D. M. Santibáñez and G. M. Bobadilla, "A model and simulation of a five-degree-of-freedom robotic arm for mechatronic courses," IEEE Latin America Transactions, vol. 12, pp. 78–86, 2014.

[18] M. F. Ahmad, H. J. Rong, S. S. N. Alhady, W. Rahiman and W. A. F. W. Othman, "Color tracking technique by using pixy CMUcam5 for wheelchair luggage follower," International Conference on Control System, Computing and Engineering (ICCSCE), pp. 186–191, 2017.

[19] J. H. Marburger, "Science, technology and innovation in a 21st century context," Policy Sciences, vol. 44, pp. 209-213, 2011.

[20] R. N. Darmanin and M. K. Bugeja, "A review on multi-robot systems categorised by application domain," Mediterranean Conference on Control and Automation (MED), 2017, pp. 701–706.

[21] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," computational intelligence and neuroscience, vol. 2018, pp. 1–13, 2018.

[22] K. L. Masita, A. N. Hasan and T. Shongwe, "Deep learning in object detection: A review," International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), pp. 1–11, 2020.

[23] J. H. Ryu, M. Irfan, and A. Reyaz, "A Review on Sensor Network Issues and Robotics," Journal of Sensors, vol. 2015, pp. 1–14, 2015.

[24] A. Sophokleous, P. Christodoulou, L. Doitsidis and S. A. Chatzichristofis, "Computer Vision Meets Educational Robotics," Electronics, vol. 10, pp. 1–24, 2021.

[25] L. Pérez, I. Rodríguez, N. Rodríguez, R. Usamentiaga and D. F. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," Sensors, vol. 16, pp. 1–26, 2016.

[26] K. B. Jang, C. H. Baek, and T. H. Woo, "Risk analysis of nuclear power plant (NPP) operations by artificial intelligence (AI) in robot," Journal of Robotics and Control, vol. 3, pp. 153–159, 2022.

[27] S. A. Salim, M. R. Amin, M. S. Rahman, M. Y. Arafat, and R. Khan, "An IoT-based smart agriculture system with locust prevention and data prediction," International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE), pp. 201–206, 2021.

[28] S. D. Perkasa, P. Megantoro and H. A. Winarno, "Implementation of a Camera Sensor Pixy 2 CMUcam5 to A Two Wheeled Robot to Follow Colored Object," Journal of Robotics and Control, vol. 2, pp. 496–501, 2021.

[29] B. Jang, M. Kim, G. Harerimana and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," IEEE Access, vol. 7, pp. 133653–133667, 2019.

[30] V. Larsson, M. Pollefeys and M. Oskarsson, "Orthographic-Perspective Epipolar Geometry," International Conference on Computer Vision (ICCV), pp. 5550–5558, 2021.

[31] A. R. A. Tahtawi, M. Agni and T. D. Hendrawati, "Small-scale Robot Arm Design with Pick and Place Mission Based on Inverse Kinematics," Journal of Robotics and Control, vol. 2, pp. 469–475, 2021.

[32] C. H. Park, J. Kim and M. G. Kang, "Color interpolation algorithm for an RWB color filter array including double-exposed white channel," EURASIP Journal on Advances in Signal Processing, vol. 58, pp. 1–12, 2016.

[33] O. Gómez, J. A. Gonzalez and E. F. Morales, "Image segmentation using automatic seeded region growing and instance-based learning," Iberoamericann Congress on Pattern Recognition, pp. 192–201, 2007.