Computer Vision-Based System for Classification and Sorting Color Objects

A T Fadhil\textsuperscript{1,2}, K A Abbar\textsuperscript{1,3}, and A M Qusay\textsuperscript{1,4}

\textsuperscript{1}Department of Production Engineering and Metallurgy, University of Technology, Baghdad-Iraq
\textsuperscript{2}Email: 70047@uotechnology.edu.iq.
\textsuperscript{3}Email: 70080@uotechnology.edu.iq.
\textsuperscript{4}Email: 70209@uotechnology.edu.iq.

Abstract. Interest in computer vision has grown considerably in industrial applications of robotics arms such as packaging, sorting, and etc. The real challenge is how to improve the current sorting system based on image processing in the four-points to identify, manipulate, select and sort objects depends on color and geometry. This paper presents a mechatronic sorting system with the application of image processing using a web camera by sensing existing parts and capturing them in real-time, then defining colors and analyzing geometric data. This information is processed for the handling mechanism. A robotic arm has 5 DOF is used in sorting the objects have different geometries (circle, rectangular, and triangle) and a different color (red, green, blue, brown, orange, and cyan). The project deals with an automated material handling system. It aims to classify color objects by color, size, and geometry, which comes on a conveyor belt by selecting and placing the parts in their pre-programmed location. Image processing is applied in EmguCV and visual studio software. An image is a matrix of pixels, image processing is the processes applied on image to convert it into digital form and implement some process on it to improve an image. The Canny edge detection algorithm is used for shape identification in image processing. A microcontroller with computer vision is used to automate the sorting process. Experimental results reveal the effectiveness, high accuracy and low cost of computer vision with a robotic arm in the sorting process according to color and shape.

Keywords: Industrial robotic; Sorting process; Microcontroller; Digital image processing.

1. Introduction

An industrial robotic arm is a device performs repetitive tasks with accuracy and speed such as sorting process and pick-place process. Recent developments in electronic and printed circuit board technology have opened up new prospects for industrial applications in the sorting and handling field and many other applications. To automation the sorting process, the robotic arm system needs to computer vision system or sensors or both of them [1, 2]. 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 some operations on it to extracting the data from this image such as determining the color and geometry of the objects [3, 4]. The controlling process is performed by the microcontroller for the robotics system components such as the sensors, drivers, and motors [5]. The current paper relates to a device and a method for classification and sorting of objects, using a mechatronics system, sensors working on a physical basis and the engineering characterization of each object.
2. Literature Review
Md. Hazrat Ali et al. [6] presented a vision system based robot Scorbot to sorting objects based on the color, shape, and size. A USB camera was mounted on the gripper and the image processing processes were performed using MATLAB software. Visual Basic was introduced to the connection between MATLAB and Scorbase software to pick and place objects.
M. M. Sofu et al. [7] proposed an automatic apple sorting process based on their weight, color, and size. Two cameras were mounted on the roller conveyor and captured four images to any apple to analyze the properties of apple using image processing. The load cells measured the weight of the apples and also estimated the weight of apple using the area of apple with error gram rate (5-6) %.
PLC used in controlled conveyors, bowls, and actuators.
Sangeetha G.R et al. [8] a developed stereo vision system based robotic arm to determine the target coordinates. The robotic arm with three degrees of freedom and an inverse Kinematic solution was used. The image processing algorithm was implemented using MATLAB and the accuracy of a robotic arm system achieved to (2 cm).
Priya Vinayak Garad [9] presented objects sorting according to shape using robotic arm and image processing. A webcam was used to capture the image and processed in image processing using MATLAB. The robotic arm with DC motors is controlled by a microcontroller.

3. System Specifications
The basic theme of this project is the object moving on a conveyor belt being sensed, selected and sorted by color and geometry. For this, a camera is used as sensor inputs, the camera is a web camera that installed above the conveyor belt and connected to the computer via USB. Figure 1 is shown the basic components of a robotic system.

![Figure 1. Robotic System with Basic Components.](image)

The camera takes a quick snapshot and feeds on the computer’s color processing. In the PC Visual Studio (C#) is used to handle the color, depending on the signal being given to the Arduino controller. The microcontroller will, in turn, control the servomotors. These motors will control the movement of the robotic arm, by controlling the angular movement. So the automatic arm will be fully controlled by servomotors. The gripper of the robotic arm will choose the object and place it depending on its color and geometry.
This is a completely automatic process no manual support needed. The microcontroller used here is Arduino which is a good platform for robotics application. Using both the software and hardware, object continuous sorting can be done in real-time.
Hardware specifications: Power supply (600 W), 5 DOF Robotic Arm, Conveyor belt, Relay, Microcontroller (Arduino Mega2560), USB Webcam, IR sensor, Ultrasonic sensor, and LCD (20x4).
Software specifications: Visual Studio (c#), EmguCV, and Arduino IDE.

3.1 Robotic Arm
The 5 DOF robotic arm with six servo motors as shown in Figure 2 is used and Table 1 illustrated the DH parameters.
3.2 Objects Manufactured
Three geometries with (circle, rectangular, and triangle) cross-section were manufactured from wood for weight considerations as shown in Figure 3 and the three containers were manufactured using an acrylic sheet as shown in Figure 4 which used to sort the objects based on geometry and color.

![Figure 2. Robotic Arm.](image)

![Table 1. DH Parameters of the Robotic Arm.](image)

| Link | $a_l$ (mm) | $\alpha_l$ (degree) | $d_l$ (mm) | $\theta_l$ (degree) |
|------|------------|---------------------|------------|---------------------|
| 1    | 0          | 90                  | 105        | $\theta_1$          |
| 2    | 105        | 0                   | 0          | $\theta_2$          |
| 3    | 100        | 0                   | 0          | $\theta_3$          |
| 4    | 0          | 90                  | 0          | $\theta_4$          |
| 5    | 0          | 0                   | 150        | $\theta_5$          |

![Figure 3. Objects Geometries (all dimensions in mm).](image)
3.3 Arduino Mega 2560
Arduino Mega 2560 is used as a microcontroller to control the motion of a robotic arm, an IR sensor, an Ultrasonic HC-SR04 sensor, a Relay, and the LCD. The Arduino Mega 2560 is shown in Figure 5 and have the technical specifications are the following:
- Operating Voltage: 5V.
- Digital I/O Pins: 54.
- Analog Input Pins: 16.
- Flash Memory: 256.
- Clock Speed: 16MHz.

3.4 IR Sensor
It is consists of two infrared sensors which work as transmitter and receiver of infrared as shown in Figure 6 and the working area of this sensor to detect the objects is 2-30 cm and the angle of detection is 35º.
3.5 Ultrasonic Sensor

The principle work of ultrasonic sensor is sending the ultrasonic waves (40 kHz) and when these waves collided with the object these waves are reflected the sensor and the distance is measured using the Equation 1.

\[ D = \frac{S \times T}{2} \]  

(1)

Where:
- D: the distance between the sensor and the object.
- S: the speed of the sound wave (340 m/sec).
- T: the time taken by the wave to travel the distance D.

The non-contact Ultrasonic HC-SR04 sensor shown in Figure 7 is used in this work to measure the height of the objects and the technical specifications of this sensor are:
- Operating voltage: 5V.
- Measuring Distance: 2-400 cm.
- Accuracy: 3 mm.
- Measuring angle covered: 15º.

![Figure 7. Ultrasonic (HC-SR04) Sensor.](image)

3.6 Camera

The Havit HV-N5081 webcam with resolution 640×480 pixels as shown in Figure 8 is used which it is installed above the belt to capture images for the objects.

![Figure 8. Webcam.](image)

4. Image Processing

The color image is captured using a USB webcam and the following steps are used in image processing to detecting the object color and geometry:
Step 1: Image Acquire
An image is represented digitally after acquired by webcam. This image consists of three dimensions \( W \times H \times D \), where \( W \) and \( H \) represent the width and height respectively, and the \( D \) dimension represents the values stored in pixel.

Step 2: Color Detection
It is performed using InRange function in EmguCV library with Visual Studio C#. In this work, six colors (Blue, Green, Red, Brown, Orange, and Cyan) are a test and the ranges of these colors illustrated in Figure 9.

```csharp
// Get The Red Color
Mat OutputImage1 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 0)), new ScalarArray(new MCvScalar(25, 25, 255)), OutputImage1);
// Get The Blue Color
Mat OutputImage2 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 0)), new ScalarArray(new MCvScalar(255, 25, 50)), OutputImage2);
// Get The Green Color
Mat OutputImage3 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 0)), new ScalarArray(new MCvScalar(50, 255, 50)), OutputImage3);
// Get The Brown Color
Mat OutputImage4 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 50)), new ScalarArray(new MCvScalar(45, 82, 160)), OutputImage4);
// Get The Orange Color
Mat OutputImage5 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 80)), new ScalarArray(new MCvScalar(100, 100, 255)), OutputImage5);
// Get The Cyan Color
Mat OutputImage6 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(130, 80, 130)), new ScalarArray(new MCvScalar(255, 153, 255)), OutputImage6);
```

**Figure 9.** The Range of the Colors.

Step 3: Image Thresholding and Filtering
To perform the image processing and detect the geometry and dimensions of the object must convert the color image to binary image and this process called Thresholding. Each pixel has intensity value between 0 (black) to 255 (white) and the threshold value used in this work was (120) and an intensity value less than this value converts to 0 (black) and value of intensity larger than this value convert to 255 (white). And Smooth Gaussian filtering is used to reduce noise in an image.

Step 4: Object Detection
After performing the above steps, the image converts to a binary image consisting of a two-dimensional matrix. Each pixel becomes to have a single value that is either zero (Black color) or one (White color) and for objects edge detection in EmguCV, a Canny edge detection algorithm is applied. This algorithm works by searching at the image horizontally and then moving to the next row and the area of the object can calculate by summing the number of pixels has one value (white color) as shown in Figure 10.

```csharp
// Get the Red Color
Mat OutputImage1 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 0)), new ScalarArray(new MCvScalar(25, 25, 255)), OutputImage1);
// Get the Blue Color
Mat OutputImage2 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 0)), new ScalarArray(new MCvScalar(255, 25, 50)), OutputImage2);
// Get the Green Color
Mat OutputImage3 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 0)), new ScalarArray(new MCvScalar(50, 255, 50)), OutputImage3);
// Get the Brown Color
Mat OutputImage4 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 50)), new ScalarArray(new MCvScalar(45, 82, 160)), OutputImage4);
// Get the Orange Color
Mat OutputImage5 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(0, 0, 80)), new ScalarArray(new MCvScalar(100, 100, 255)), OutputImage5);
// Get the Cyan Color
Mat OutputImage6 = new Mat();
CvInvokeInRange(frame, new ScalarArray(new MCvScalar(130, 80, 130)), new ScalarArray(new MCvScalar(255, 153, 255)), OutputImage6);
```

**Figure 10.** Objects Detection.

The image processing steps are shown in Figure 11.
4.1 Camera Calibration

All measurements are performed on the image using image processing measured by pixel, and the dimensions of objects captured are measured in millimeter, hence, calibration was used to convert these dimensions from pixel to millimeter unit. The camera has two angles of view with x-axis ($\phi_x$) and y-axis ($\phi_y$), the distance between the lens and image is known by the depth of the image. According to Figure 12 and the angles ($\phi_x$) and ($\phi_y$), the depth can be obtained:

**Figure 12.** The View Angle of the Camera (a) x-axis (b) y-axis.
Where:

- $P_x$: The number of Pixels in columns (640 pixels of camera used).
- $P_y$: The number of Pixels in rows (480 pixels of camera used).
- $\phi_x$: Angle along the x-axis.
- $\phi_y$: Angle along the y-axis.
- $d_x = d_y$: The depth of the image.

\[
d_x = \frac{P_x}{\tan\left(\frac{\phi_x}{2}\right)} \tag{2}
\]
\[
d_y = \frac{P_y}{\tan\left(\frac{\phi_y}{2}\right)} \tag{3}
\]

Figure 13 illustrated the method of object detection by the camera in its coordinate system. Hence, the center position of the area to object in camera coordinate is obtained using the parameters ($x$ and $y$) relative to camera view window reference. The values of parameters are measured in pixel.

- $x = d_x \tan \alpha$ \hspace{1cm} \tag{4}
- $y = d_y \tan \beta$ \hspace{1cm} \tag{5}

In this work, the ultrasonic sensor was used to measure the object's height and then obtained the image depth to calculate the dimensions in the pixel unit. Therefore the scale factor in this work depends on the object height:

- At object height (65 mm).

\[
Scale\ factor\ (S) = \frac{40\ mm}{196\ pixel} = 0.204 \left(\frac{mm}{pixel}\right) \tag{6}
\]
4.2 Centroid of Object
The object in the image consists of the pixels and to calculate the centroid of it using the image processing, the image is converted to the binary image and then determined the moment of the object which is the average weight of the object pixels. Equations 7, 8, and 9 are used to calculate the centroid for the geometries of the objects used in this work:

- For Rectangular geometry.
  \[ \text{Centroid} (x, y) = \frac{1}{n} \sum_{i=1}^{n} V_{n(x,y)} = \frac{V_1(x,y) + V_2(x,y) + V_3(x,y) + V_4(x,y)}{4} \]  
  (7)

- For Triangular geometry.
  \[ \text{Centroid} (x, y) = \frac{1}{n} \sum_{i=1}^{n} V_{n(x,y)} = \frac{V_1(x,y) + V_2(x,y) + V_3(x,y)}{3} \]  
  (8)

- For Circular geometry.
  \[ \text{Centroid} (x, y) = \frac{\text{Area}}{\pi} = \text{Radius} \]  
  (9)

5. Methodology
Many actions performed to automate the sorting process and make it work real-time in detection the object color and geometry. Figures 14 and 15 are illustrated the components of the proposed system and the control board respectively and the steps of the robotic system to perform the sorting process showed in Figure 16.

![Figure 14. The Robotic System.](image1)

![Figure 15. The Control Board.](image2)
Figure 16. The sequence of the Sorting Process.
6. Results and Discussion
One of the main aims in this work was the process of transferring data between the Microcontroller and the Visual Studio (used to image processing) to make the robotic system work real-time and it was implemented by a special protocol was programmed by the Visual Studio C#.

The sorting process was performed for different objects geometries (Rectangular, circular, and triangular) with different colors (red, green, blue, brown, orange, and cyan) using the image processing in EmguCV and Arduino IDE (microcontroller). Figure 17 is shown the strategy of sorting the objects and place them in a specified place.
From Figure 18, the results extracted from the image processing were the color and the area of the objects were listed in Table 2. And the error between the actual area and the extracted area were measured.

Figure 17. The strategy of Sorting the Objects in Containers.
Figure 18. Image processing outputs.
Table 2. The error in the area.

| Sample No. | Color | Geometry | Actual Area (mm²) | Measured Area (mm²) | Absolute Error % |
|------------|-------|----------|-------------------|---------------------|-------------------|
| 1          | blue  | circular | 1256.637          | 1250.36             | 0.499             |
| 2          | blue  | rectangular | 600            | 609.758             | 1.626             |
| 3          | blue  | triangular | 450            | 460.377             | 2.306             |
| 4          | green | circular | 1256.637          | 1255.63             | 0.08              |
| 5          | green | rectangular | 600            | 606.678             | 1.113             |
| 6          | green | triangular | 450            | 461.938             | 2.653             |
| 7          | red   | circular | 1256.637          | 1265.9              | 0.737             |
| 8          | red   | rectangular | 600            | 607.677             | 1.279             |
| 9          | red   | triangular | 450            | 461.771             | 2.616             |
| 10         | orange | circular | 1256.637          | 1240.3              | 1.3               |
| 11         | brown | rectangular | 600            | 611.755             | 1.959             |
| 12         | cyan  | triangular | 450            | 458.608             | 1.913             |

The results indicated the minimum error between actual area and detected area was (0.08%) with the circular geometry of the objects and the maximum error was (2.653%) with triangular geometry of the objects. Figure 19 is shown the sorting process of the objects based on the objects geometries and their colors in the desired container.

Figure 19. The Sorting Process.
7. Conclusion

The image processing algorithm was used in this work gave the desired results in terms of detecting the color of the objects as well as detecting the geometries shape of the objects and extracting its dimensions. Based on that, the robotic system was used for real-time working to sort the objects with accuracy 100%. The image processing algorithm can be extended to use to detecting more different geometries of objects but the main limitation in the robotic system was the picking process of the objects because the robotic arm used have 5 DOF. Suitable lighting is required to achieve good image capturing which led to giving good results in an image processing program. The project worked successfully. There are two main steps in the sensing part, revealing objects and recognizing. The system successfully performs the handling station task, which is the selection and location. Thus a cost-effective mechatronic system can be designed using the simplest concepts and effective results can be observed.

References

[1] Raza K, Khan A T, and Abbas N, 2018 “Kinematic analysis and geometrical improvement of an industrial robotic arm”, Journal of King Saud University – Engineering Sciences, Vol. 30, pp. 218-223.
[2] Kurka P R G and Salazar A A D, 2019 “Applications of image processing in robotics and instrumentation”, Mechanical Systems and Signal Processing, Vol. 124, pp. 142-169.
[3] Wang Z, Li H, and Zhang X, 2019 “Construction waste recycling robot for nails and screws: Computer vision technology and neural network approach”, Automation in Construction, Vol. 978, pp. 220-228.
[4] Tiwari M, Lamba S S, and Gupta B, 2018 “An image processing and computer vision framework for efficient robotic sketching”, International Conference on Robotics and Smart Manufacturing, Procedia Computer Science, Vol. 133, pp. 284-289, Kazakhstan.
[5] Cunha J, Ferreira R, and Lau N, 2015 “Computer vision and robotic manipulation for automated feeding of cork Drillers”, Materials and Design, Vol. 82, pp. 290-296.
[6] Ali H M, K Aizat, K Yerkhan, T Zhandos, and O Anuar, 2018 “Vision-based Robot Manipulator for Industrial Applications”, International Conference on Robotics and Smart Manufacturing, Procedia Computer Science, Vol. 133, pp. 205–212, Kazakhstan.
[7] Sofu M M, Er O, Kayacan M C, and Cetisli B, 2016 “Design of an automatic apple sorting system using machine vision”, Computers and Electronics in Agriculture, Vol. 127, pp. 395-405.
[8] G R Sangeetha, Kumar N, Nishank P R Hari, and S Sasikumar, 2018 “Implementation of a Stereo vision based system for visual feedback control of Robotic Arm for space manipulations”, International Conference on Robotics and Smart Manufacturing, Procedia Computer Science, Vol. 133, pp. 1066–1073, Kazakhstan.
[9] Garad P V, 2017 “Object Sorting Robot Based On the Shape”, International Journal of Advance Research, Ideas and Innovations in Technology, Vol. 3, pp. 129-134.