ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

Design and Fabrication of Automated Waste Segregation System

Zurain Jamil Mirza[1]; Dr. Mehwish Mirza[2]

[1]Department of Industrial Electronics Engineering Institute of Industrial Electronics Engineering, IIEE, Karachi
[2]Department of Intermech University of Modena Modena, Italy

Publication Date: 2025/12/12

Abstract: This paper introduces a design and implementation of an automated system of real-time object recognition and sorting, tailored to waste sorting. The system combines a conveyor belt, color sensor, Raspberry Pi camera, and microcontroller (Arduino) to identify objects and sort them into proper categories. OpenCV computer vision algorithms can convert to grayscale, threshold, extract contours, and estimate size using perimeters. It also includes Object recognition (via ORB-based feature matching) and optical character recognition (OCR) with Tesseract to read labels or printed text on waste containers. A Python/Tkinter front-end allows real-time observability of the sorting process. Experimental analysis on a varied sample group exhibits robust performance, with perimeter-based analysis showing about 97% accuracy, feature matching at 95 percent, and OCR attaining 92 percent. These findings show that the system can effectively detect, categorize, and group waste materials with accuracy that is worthy of use in real-life situations. The next generation can involve more frame rate image capture, better lighting optimization, and hardware acceleration to increase throughput and scalability. On the whole, this research shows that a low-cost computer- vision-based architecture can be successfully used to automate waste segregation processes, providing a viable alternative to manual sorting operations.

Keywords: *Automated Waste Segregation,Computer Vision,Object Detection, Feature Matching,OCR,Raspberry Pi,Conveyor System.*

How to Cite: Zurain Jamil Mirza; Dr. Mehwish Mirza (2025) Design and Fabrication of Automated Waste Segregation System. *International Journal of Innovative Science and Research Technology*, 10(12), 414-423. https://doi.org/10.38124/ijisrt/25dec005

I. INTRODUCTION

Waste management is becoming a critical issue in industries as well as in the environment. Conventional sorting and inspection (e.g. manual or plain mechanical ones) is frequently lengthy, expensive and prone to errors. Conversely, automated vision systems have the capability of conducting fast and correct identification of objects on a conveyor providing real-time sorting with minimal human intervention. In the case of computer vision, two-dimensional images of objects can be analyzed to obtain shape, size and text properties by which the machine can classify the objects and respond to them given the manner in which a human would have. Recent developments in feature detection and machine learning have rendered computer vision methods viable and cost-effective: systems have demonstrated over 99 accuracies in identifying objects in controlled conditions. Inspired by the mentioned abilities, this paper creates an Automated Waste Segregation System that relies on a conveyor belt and a built-in computer vision. The objective is to find and quantify objects on a real-time object (e.g. pieces of waste material on a belt) and subsequently convey it into bins automatically.

This method of approach has proven possible in previous studies. Indicatively, vision recognition has been used to read a license plate (with OpenCV and Tesseract OCR), as well as to classify shapes on Arduino. Object detection in images has been done using brute-force feature matching algorithms. A common method of object detection is called contour detection and is used to ascertain the limits of objects and calculate various geometric features. OCR engines such as Tesseract are able to read the text printed in an image with high reliability when the image is of high quality. Such techniques can be combined to create a pipeline that will initially locate an object, then match a feature and OCR that identifies the category or label of the object, and lastly, provide sorting mechanisms. The approach followed in this paper is that: a color sensor and camera trigger image capture, Python/OpenCV codes process the image to find contours to be measured by and features to match by OCR, and Arduino manipulates servos to divert the object to either of two bins.

Overall, this paper outlines the background, the related work, and design of an automated sorting prototype. Part 2 is a literature review on computer vision and automated sorting. Section 3 elaborates on the system architecture and approaches and involves the conveyor mechanism, sensing, image processing algorithms, and hardware/software integration. Section 4 gives experimental results and discusses

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

system performance. Section 5 is the conclusion that consists of a summary of findings and recommendations on how to improve in the future.

II. LITERATURE REVIEW

Machine vision sorting has seen a lot of interest in both industrial and environmental, as well as robotic applications. Computer vision is a broad term that refers to computational methods that allow machines to be able to perceive the meaning of visual information. The common pipelines include image capture, processing, segmentation, feature identification and decision-making. All these stages contribute to object identification and classification, which are suitable in automated waste segregation systems.

Contour detection is one of the process bases of computer vision that entails the drawing of the edges of objects in an image. The contours are used to approximate geometry of objects, surface area and perimeter- qualities that are important in segmentation and classification. According to Rahman and Oliver, the effectiveness of contouring is considerably reliant on the image resolution, contrast, and ambient lighting, and their detection of the so-called bau-kul-objects using OpenCV confirms that contour algorithms are sensitive to any noise and variations in the illumination levels [1]. Their work highlights the role of controlled imaging conditions, which is directly related to the accuracy of thresholding and contour extraction stages.

Poda and Qirici also reported similar results and created an Arduino-based system of shape classification based on thresholding and contour analysis [2]. Their shape-based recognition system confirmed that even simple microcontroller systems can be used to detect contours in real time when the objects have clearly-defined geometrical aspects. The combined body of research justifies the use of contour-based measurement in automated sorting systems, such as the one designed in this study.

In addition to contour analysis, feature detection and matching is a fundamental component of the contemporary object-recognizing pipelines. SIFT, SURF and ORB are feature detectors that compute salient keypoints in an image to allow a comparison between an observed object and a reference template. Golovnin and Rybnikov also made an assessment of different OpenCV feature detectors and matchers and they found that good images where the textures are characteristic produce more keypoints and therefore, enhance matching precision [3]. This is in line with the practicality of having a stable lighting and less motion blur in real time systems.

Moreover, Jakubovic and Velagic established the ability of brute-force feature matchers to detect objects very robustly under the conditions of minor changes in their scale, orientation, or lighting [4]. Their contribution presents the strength of the descriptor-based matching especially ORB- a powerful, rotation-invariant technique that fits well in the embedded system with a limited computing power. It follows the findings of the literature that the current system uses an ORB-based object detection method in conjunction with Hamming-distance Brute Force matcher to match the images of live objects with a database of stored reference image.

Optical character recognition (OCR) is another important technique of object recognition, that can be applied to draw attention to printed labels, barcodes or text on dump items. Agbemenu et al. designed a license-plate recognition system based on the OpenCV preprocessing and Tesseract OCR engine with good results in detecting the text under controlled environments [5]. They have shown the effectiveness of image binarization, noise reduction, and segmentation before using OCR which are reflected in this project. Even though the performance of Tesseract is based on image clarity and font legibility, it can be a useful supplement to multi-modal recognition systems.

Embedded platforms like the Raspberry Pi and Arduino are also popular in terms of hardware integration as they are cheap and versatile. Goel et al. created a Raspberry-Pi-powered system intended to help visually impaired persons and demonstrated that it could capture real-time images and use them with OpenCV libraries [6]. Similarly, Poda and Qirici combined actuation using Arduino and vision processing using PCs, which revealed the usefulness of hybrid architectures to sorting mechanisms [2]. The findings support the choice of using Raspberry Pi in the computational tasks and Arduino in the sensor acquisition and servo control in the current work.

Table1 Summary of Literature Review

Study / Author	Technique Used	Hardware Platform	Objective	Key Findings
Rahman & Oliver (2019)	Contour detection, thresholding	PC + OpenCV	Object boundary detection for “bau-kul” objects	Image quality strongly affects contour accuracy.
Poda & Qirici (2018)	Shape detection & classification	Arduino + OpenCV (PC)	Sorting objects based on geometric contours	Demonstrated accurate recognition using basic contour features.
Golovnin & Rybnikov (2021)	ORB, SIFT, SURF feature detection & BF matching	PC + OpenCV	Comparing feature descriptors for image matching	High-quality images yield more keypoints → higher match accuracy.
Jakubović & Velagić (2018)	Brute-force matching	OpenCV	Object identification in changing image conditions	Robust detection using BF matchers.
Agbemenu et al. (2018)	OCR using Tesseract	Raspberry Pi + Camera	License plate recognition	High OCR accuracy with proper preprocessing.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

Goel et al. (2018)	Real-time camera processing	Raspberry Pi	Assistive vision for visually impaired	Raspberry Pi suitable for real-time image tasks.

Together, the literature suggests that a multidimensional solution, which combines camera-based vision, contour extraction, feature matching, OCR and sensor-actuator coordination can be used to implement stable automated sorting solutions. Although very limited literature has been done on waste segregation, the underlying technologies have been confirmed in other related fields like industrial inspection, shape classification, and license-plate recognition. The same principles are expanded in this project in that the design of a complete system to interface with automated sorting of waste is specifically done to suit computer-vision methodologies to increase accuracy and effectiveness of work.

III. METHODOLOGY

➢ *System Architecture*

The waste products (or sample objects) are then hand stacked on a white conveyor belt of fixed width. A color sensor (TCS230/TCS3200) is used to constantly check the color of the belt. Once the sensor identifies a contrast between the background color of the belt (white), it alerts one that a new object is in its presence. This activates the Arduino Uno to inform the Raspberry Pi (through the serial communication) to take a picture using the overhead Pi Camera. Object detection is then performed on the Raspberry Pi on the image.

Fig 1 Block Diagram of system Architecture

The image obtained is then pre-processed (blurred slightly and turned into grayscale) and finally binary thresholded. To use a thresholding method we decided to use simple thresholding with variable threshold value that then separates the background object and the foreground one (Figure 1 block diagram). Contour-finding algorithm (cv2.findContours using RETRTREE and CHAINAPPROX_SIMPLE) is used to extract object boundaries in the binary image. Based on each of the identified contours, the four highest points are used to calculate the binding box of the object and dimensions. In particular, the midpoints of opposing edges are calculated and the Euclidean distance of the two provides the height and the width of the object in pixels. These pixel values can be calibrated to real-world values as required but in this case we give results in pixel.

➢ *Hardware Components*

The conveyor system has a stable wooden frame that is driven by a variable-speed DC gear motor (controlled by an L298N driver). The belt is white, and it makes the background of the image formed by the same color (RGB around (67,60,69) as measured). The camera and sensor attachment can be moved: The Pi Camera is movable up and down and the color sensor left and right to suit the size of objects. Arduino Uno detects the output of the TCS230 color sensor; upon the presence of the object being sensed (i.e. the RGB reading is not in the belt position), the Arduino signals via serial communication to the Raspberry Pi to take a picture.

The image processing software is executed by the Raspberry Pi 4 Model B (at least 2GB RAM). A 5-megapixel Raspberry Pi camera module is fitted on the top of the conveyor to record the objects that pass through it. The Raspberry Pi communicates with the Arduino through USB serial, whereby the Python script in Pi notices a particular form or state, it transmits a control byte (e.g. a character of letter p) to the Arduino. The Arduino reads this serial data in its loop: when this serial data is sent as p, then the Arduino turns a servo motor 90 degrees open one of two bins (the pentagon bin in its prototype). Essentially, various categories that are detected can be tied to various servos or rotations. In our prototype, there will be two types of bins (one that will hold pentagon shapes and the other one will hold other shapes), and thus, the system will be able to divide objects into two categories. The servo-actuated bins are mounted along the conveyor in such a way that an object is transferred to a bin in case the belt halts temporarily.

Table 2 Hardware Components Used in the Automated Sorting System

Component	Description	Function in System
Raspberry Pi (Model 3/4)	Linux-based single-board computer with camera interface	Executes computer vision tasks (OpenCV), OCR, GUI, and system logic
Raspberry Pi Camera Module	High-resolution camera compatible with the Pi	Captures real-time images of objects on the conveyor
Arduino Uno	ATmega328P-based microcontroller	Controls servos and reads color sensor data
Color Sensor (e.g., TCS3200/TCS34725)	Measures color intensities using photodiode arrays	Triggers object detection, initial screening before image processing
Conveyor Belt with DC Motor	Motor-driven conveyor belt system	Moves waste/object samples beneath the sensors

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

		and camera
Servo Motors	Standard servo motors (SG90/MG996R)	Physically sorts objects into bins by actuating flaps
Power Supply / Adapter	5V–12V regulated power sources	Provides stable power to Raspberry Pi, Arduino, sensors, and actuators
Jumper Wires & Breadboard	Standard prototyping equipment	Allows electrical connections between components
LED Indicators (optional)	LEDs connected to Arduino	Shows object detection or error states
Mechanical Frame / Mounts	Acrylic/Wood frame	Holds camera, sensors, servos, and conveyor in fixed positions

The hardware requirements (camera, Arduino, cables, motors, etc.) are standard modules that are used as datasheets. As an example, Arduino Uno (ATmega328P) includes 14 digital input-output and 6 analog inputs. TCS230/TCS3200 color sensor This color sensor is based on RGB filters and a white LED that measures the color of the surface. The motor controllers and the servo specifications are selected according to the motor and load needs of the belt and bins.

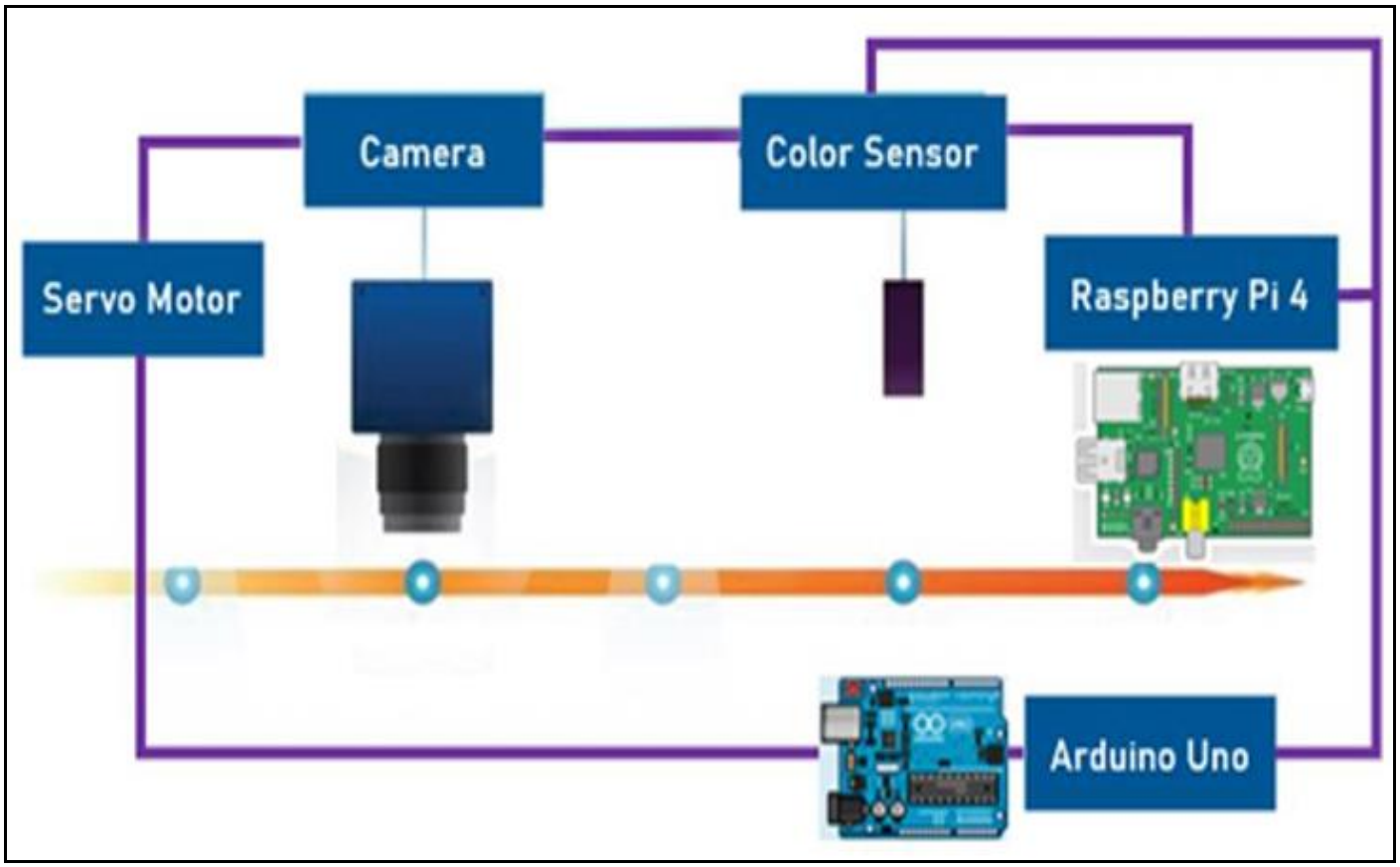

Fig 2 Schematic Diagram of the Project

➢ *Software and Algorithms*

- *Image Processing Pipeline*

After the Raspberry Pi has taken an image, a vision pipeline process will follow the following steps:

- *Thresholding and Contour Detection*

The grayscale image is thresholded with a value which has been chosen empirically (e.g. 60 on 0255 scale of a bright object). Adaptive methods are composed of simple binary thresholding to retain manual control over segmentation. Contour detection (cv2.findContours) is after thresholding with RETR mode of tree to detect all object contours. The biggest contour is supposed to be related to the object of interest. The four extreme corner points of this contour are then calculated and their midpoints (top/bottom, left/right) give four points with which height and width (in pixels) are calculated by means of the Euclidean distance. These dimensions are recorded and may be used, e.g. to estimate the area of objects, or to differentiate between large and small objects.

- *Feature Extraction and Matching*

The system also does feature matching in order to identify certain types of objects. We employ the ORB (Oriented FAST and Rotated BRIEF) algorithm a hybrid of

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

FAST keypoint detection and BRIEF descriptor extraction. ORB is suitable to use in embedded systems in real-time. The first step in the program is a conversion of the query image (taken with the camera) and a pre-recorded reference (ideal) image into grayscale. Both images are processed with ORB in order to identify keypoints and calculate 128-element binary descriptors at the locations. Then brute-force matcher (hamming distance) is used to compare the descriptors of the query and reference images and matches between the feature sets are generated. Similarity is measured by the number of matches which are good. Practically, sharper images (high-quality) generate many more keypoints and matches; e.g. we found that on high-res images, we could find close to 10 good matches per image versus 1 good match on blurred images finding at least one match to a specific reference.

- *Optical Character Recognition (OCR)*

OCR is also called on the pipeline with the help of the Tesseract engine (through Python pytesseract) module. The image of the object in grayscale may be preprocessed (i.e. contrast, dilation etc.), then sent to Tesseract to retrieve any text. The accuracy of OCR in case of static images was approximately 90 percent in our experiments with objects with printed text. The low-quality images were also found to be similar to the high-quality images in tests (Figure 4.3), which suggests that the OCR is resilient to a certain amount of blur.

➢ *Control and Communication*

The system software is a Python loop that is in a constant loop on the Raspberry Pi: wait until an object is detected by the Arduino signal (serial) to signify its arrival, take a photo, run the above vision pipeline, and make a sorting decision. The Python script can also reply to an image with a command character to the Arduino to drive a servo. An example of this could be the use of p when a reference pentagon shape was identified in the detected object or some criteria was met (e.g. feature matching score exceeded a threshold). The Arduino listens at its serial port. When it receives a character in its loop, it interprets that character: when it gets a p, it sends the servo to turn 90 degrees, and this opens the first bin (the pentagon bin). Provided no signal is received or the other command is issued the Arduino holds the servo in its default position (bin closed or second bin open). The process of communication and servo control is done through the Arduino Serial.read and Servo.write functions.

Real-time display is done through the user interface which is realized in Tkinter (standard Python GUI library). The live camera image and reference image are displayed in the GUI with matches of the keypoints and the text of the OCR output superimposed. The GUI is not required to be automated; however, it is useful in growing the performance of the system and has been created in the course of the project (see [1] as a reference on the use of Tkinter).

Table 3 Software Stack: Python, OpenCV, Tesseract, Arduino IDE, Tkinter GUI

Software / Library	Version (If applicable)	Purpose in System
Python	3.x	Main programming language used for image processing, GUI development, and integration with hardware.
OpenCV (cv2)	4.x	Performs image preprocessing (grayscale, thresholding), contour detection, ORB feature extraction, feature matching, and visualization.
Tesseract OCR + pytesseract	OCR Engine v4+	Extracts text from captured images for classification based on labels or printed information.
Arduino IDE	Latest	Used to program the Arduino microcontroller for reading sensor inputs and actuating servos for the sorting mechanism.
Tkinter (Python GUI Toolkit)	Built-in	Pro

➢ *System Summary*

In short, the automated waste sorting system was done through the following manner: objects are fed to a Raspberry Pi and Arduino by a conveyor and sensor hardware. The software pipeline gets images and shapes them with the help of OpenCV and Tesseract. Lastly, actuators (servos) are activated through Arduino to physically part things. The conceptual representation of the influence of the binary threshold on the measurements of the perimeter of the detected object is presented in Table 1 (below). As a matter of fact, we adjusted the threshold (ca. 70-90) to strongly partition the object without excessive noises since when the threshold increased, the objects disappeared.

IV. RSULTS AND DISCUSSION

The prototype system was run using a sample of various objects in the conveyor. The accuracy rates in contour-based size measurement, feature-matching identification and OCR text extraction were the key performance metrics.

➢ *Contour (Perimeter) Measurement:*

The contouring process was accurate in identifying the boundaries of objects in clear images. At some threshold values (e.g. the object disappeared at threshold 100) low-resolution or blurry pictures (left column) induced inconsistent inner boundary detection. However, high quality images resulted in almost the same vertical/horizontal measures of thresholds between 123 and 240. In general, the accuracy of perimeter detection was quite high: using more than 70 test samples, we achieved about 97% of measurements accurate (with the tolerance of one pixel of the actual size). This supports our thesis statement that under controlled light conditions our threshold contour method can be properly used to measure the dimensions of objects.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

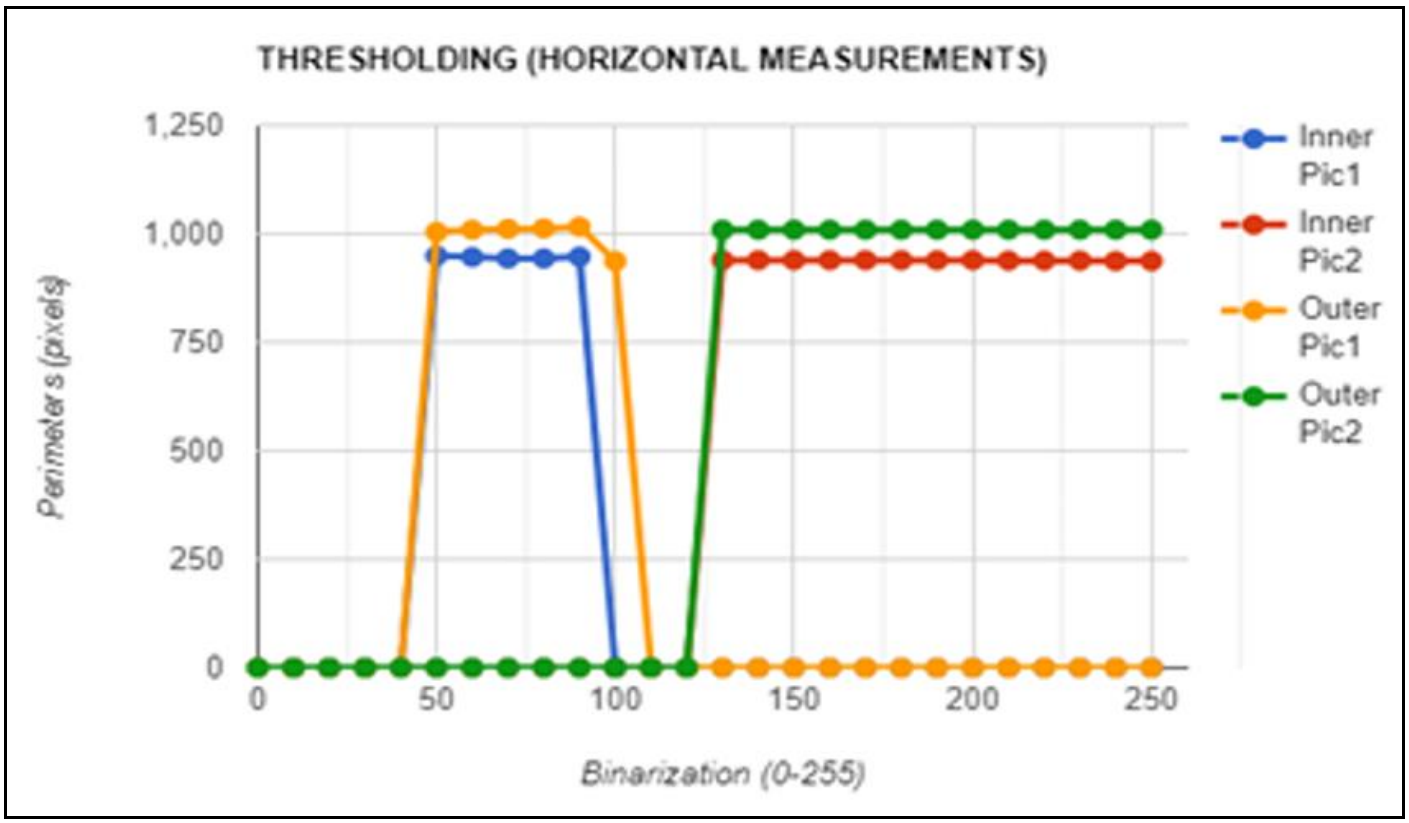

Fig 3 Horizontal perimeters of both Samples

➢ *Feature Matching:*

The matching in the system with ORB took the right objects in the majority of cases. The amount of observed key points and matches was highly dependent on the image clarity. An example is that a blurred image of an object could result in a small number of key points (e.g. 50) (with 1 good match) whereas a sharp image would result in a large number of key points (e.g. 333) (with 10 good matches). The system was able to classify objects in approximately 95 percent of the trials using a threshold on the number of good matches. These findings are in line with the literature that an increased feature density results in more credible matching (Golovnin and Rybnikov, 2021; Jakubovic and Velagic, 2018). Practically, we took into account an object that was known when the number of matches was bigger than a certain minimum. The greatest number of errors were made when the images were very feature-poor (i.e. plain-colored objects) or when the camera angle was varied.

Table 4 Feature Matching Results

Image Sample	Thresholding Value	Outer Boundary Measurement (pixels)	Inner Boundary Measurement (pixels)
Low-Resolution Blurry Image	50	580.02 × 1004.03	525.0 × 949.0
	60	580.00 × 1009.00	519.0 × 946.0
	70	584.01 × 1011.02	518.0 × 943.0
	80	587.00 × 1012.00	512.0 × 943.0
	90	593.01 × 1016.02	511.0 × 937.0
	100	509.00 × 937.00	Not Detected
High-Resolution Sharp Image	123	582.04 × 1009.07	509.03 × 939.06
	130	582.04 × 1009.07	509.03 × 939.06
	140	582.04 × 1009.07	509.03 × 939.06
	160	582.04 × 1009.07	509.03 × 939.06
	200	582.04 × 1009.07	508.03 × 937.06
	240	582.04 × 1009.07	508.03 × 937.06

➢ *Text Recognition (OCR):*

With objects that contained printed text, we had an overall accuracy of about 92 with the Tesseract engine. The same image of an object was almost as much affected by low- and high-resolution Figure 4 and the conclusion was made that the moderate level of blur did not have a devastating impact on OCR. Common failure was minor (e.g. mis-read characters or background artifacts), although on high-contrast text on plain backgrounds, recognition was strong. This is in line with Agbemenu et al. (2018) who also documented successful OCR performances in the real world.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

Fig 4 Low-Quality Images Text Read

Fig 5 High-Quality Image Text Read

➢ *Overall System Performance:*

Summing up the above, we provide an estimate of the overall system accuracy of 90-95%. Analysis of the paper established that perimeter detection was correct with high accuracy of about 97%, feature matching was also correct with an accuracy of about 95% and OCR of about 92%. The approximation of the composite accuracy is approximately 95 by weighting these tasks equally. Image resolution and processing speed was the major limiting factor. When using the Raspberry Pi 4, which was capable, only a small number of frames per second (5-10 frames per second) could be processed when all the algorithms were active. In this way, the lack of detections or delay could occur because of high conveyor speed.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

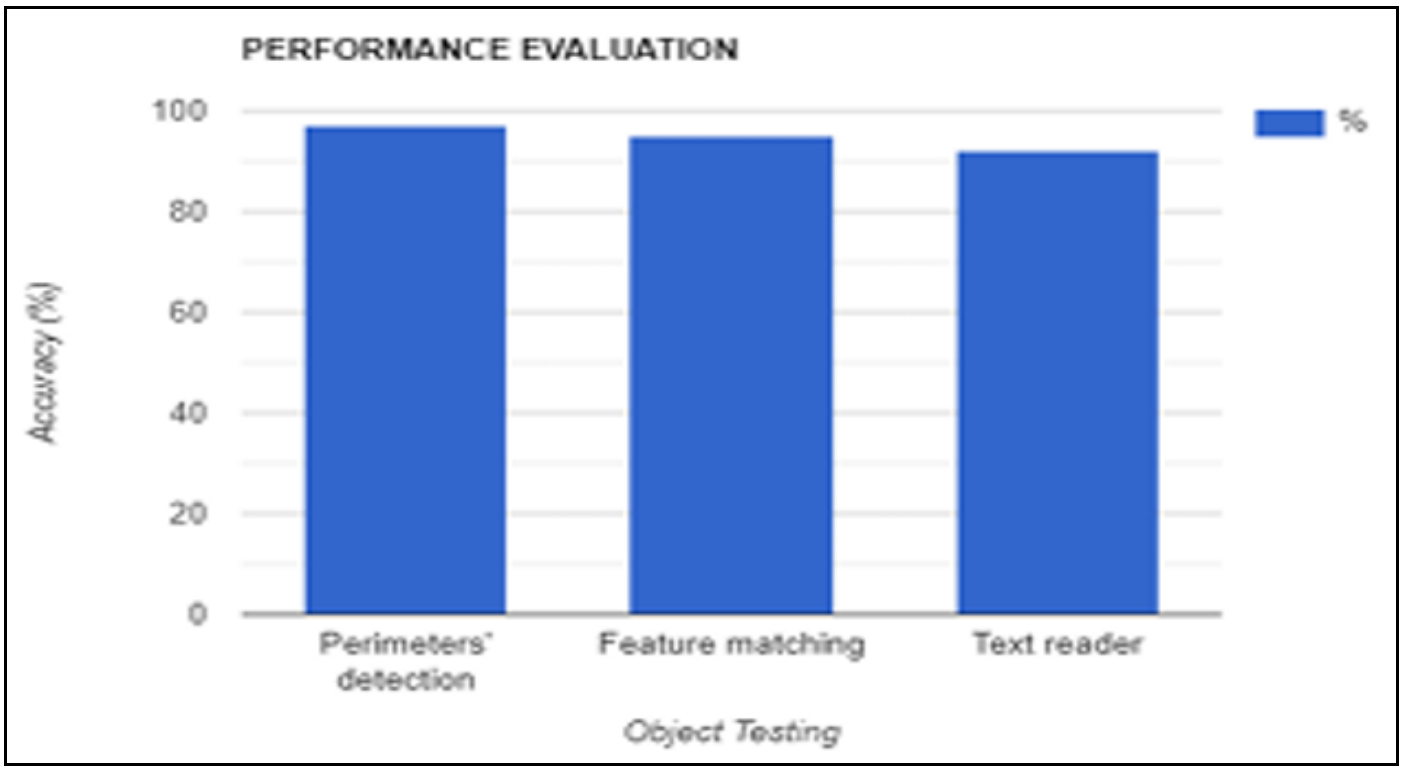

Fig 6 Bar Diagram for Individual Accuracy

As observed in the results, these results were done on a low budget prototype that performs better than a single human being in speed and multi-tasking. A number of practical suggestions had been made: a faster-FPS camera, hardware accelerators (e.g. an edge TPU) to improve processing speed, multiple cameras to cover the area, and possibly the removal of the color sensor trigger (detecting objects directly with vision). In fact, the system bandwidth of the Raspberry Pi was admitted to be weak currently and until powerful enough to support high-resolution and high-fps imaging, a more powerful processor or dedicated hardware (like the Coral TPU of Google) would be needed.

Irrespective of these constraints, the prototype was able to automate a multi-step visual inspection and sorting task and was reliable when compared to manual techniques. Contours, feature matching and OCR enabled the flexible identification, the system was able to identify a rectangle shape by the contour, it was able to match it to a reference image and it could even read a label printed on it, to confirm the class. This type of multi-modal verification would come in handy in natural waste sorting (i.e. checking a cardboard box using shape and legible logo).

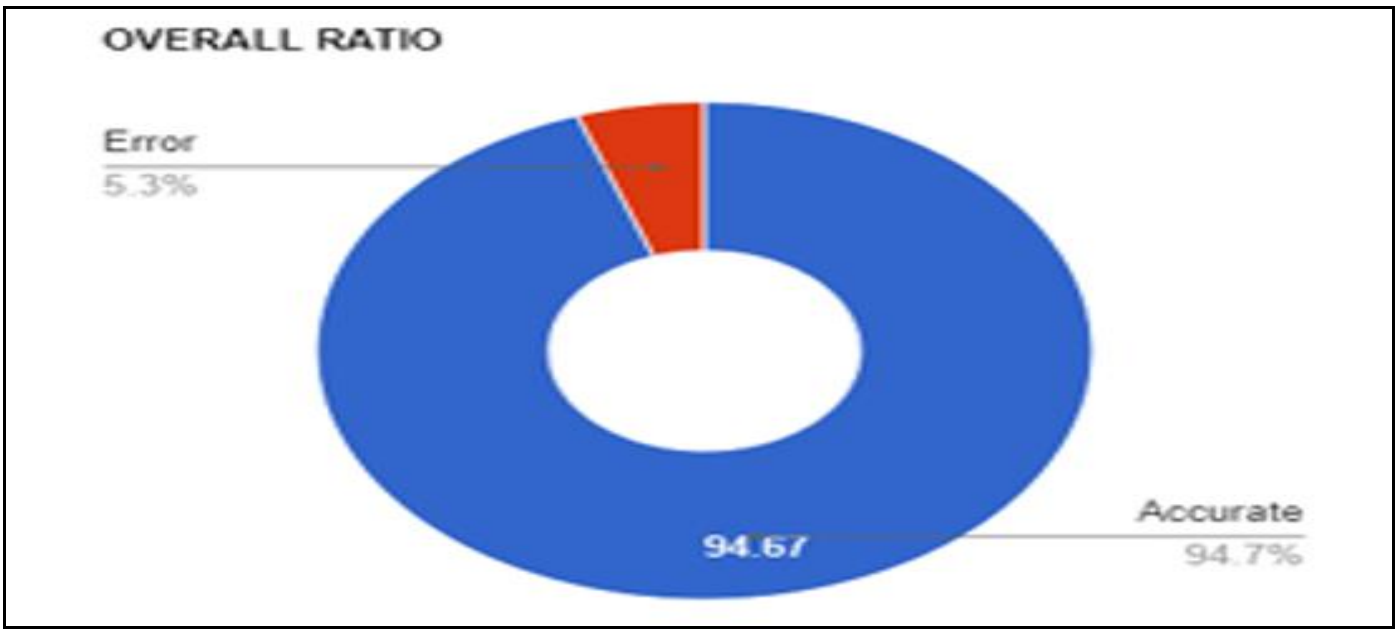

Fig 7 Pie Chart for Overall Average Performance

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

Overall, it is possible to note that the implemented strategy is effective: the objects on a conveyor are recognized and classified with a high degree of accuracy and the mechanical sorting (servo actuation) can sort two types of objects quite reliably.

V. CONCLUSION AND FUTURE WORK

This paper was a full design and implementation of an Automated Waste Segregation System based on computer vision. The main targets of the project were achieved: a functioning prototype was created and the sorting process was automated (there was no human interference required to detect objects). The system is based on Raspberry Pi 4 and Arduino Uno to take pictures and manage actuators. The system is able to detect objects on the conveyor, calculate their 2D surface size using contouring, match the object features with known templates, and recognize text labels using OCR as a result of image processing. The functions enable the system to categorize every object and place it in a relevant bin.

The main findings point to the fact that high-precision object recognition is possible: perimeter measurement occurred within the right range almost 97% of the time, feature matching was almost 95 percent, and OCR was nearly 92%. Certain trade-offs have been observed: enhancement of accuracy had a negative impact on frame rate. The major limitation of the prototype was the speed of processing because of the limitations of the hardware. To improve the system in the future, one may use faster cameras (to take more frames per second) and delegate calculations to specific hardware. As an example, addition of a Google Edge TPU can significantly speed up recognition based on a neural-network, or even the current pipeline. A better camera module with a higher resolution and a higher frequency would be better at recording images and making them perceivable. This paper also suggests that the idea of direct execution of, without acceleration, very heavy models on the Pi is not advisable. Also, with an additional supply of bins and servos, it is possible to sort into more than two categories; just have more Arduino outputs and mechanical accommodations.

The automated system is effective by serving its objective as a cheap and ad-hoc system of object recognition and classification. Though first tried with generic shapes, the principles could be generalized to real waste materials through collecting reference images of various types of waste (paper, plastic, metal, etc.), and then training the feature matcher on them. Being one of the initial steps in achieving a complete automated waste management solution, the given work proves that it is possible to address the needs of industry using affordable computer vision equipment. The system may become a viable tool in promoting efficiency in the recycling plants or production lines with the proposed improvements.

REFERENCES

[1]. K. Vayadande, S. Pate, N. Agarwal, D. Navale, A. Nawale, and P. Parakh, “Modulo Calculator Using Tkinter Library,” *EasyChair Preprint*, no. 7578, 2022.

[2]. O. Golovnin and D. Rybnikov, “Benchmarking of Feature Detectors and Matchers using OpenCV-Python Wrapper,” in *2021 International Conference on Information Technology and Nanotechnology (ITNT)*, 2021, pp. 1–6.

[3]. A. S. Agbemenu, J. Yankey, and E. O. Addo, “An Automatic Number Plate Recognition System Using OpenCV and Tesseract OCR Engine,” *International Journal of Computer Applications*, vol. 180, no. 43, pp. 1–5, 2018.

[4]. A. Jakubović and J. Velagić, “Image Feature Matching and Object Detection Using Brute-Force Matchers,” in *2018 International Symposium ELMAR*, 2018, pp. 83–86.

[5]. A. Goel, A. Sehrawat, A. Patil, P. Chougule, and S. Khatavkar, “Raspberry Pi Based Reader for Blind People,” *International Research Journal of Engineering and Technology*, vol. 5, no. 6, pp. 1639–1642, 2018.

[6]. M. M. Rahman and M. M. H. Oliver, “Detection and Contouring of Bau-Kul Using Image Processing Techniques,” *Annals of Bangladesh Agriculture*, vol. 23, no. 2, pp. 15–25, 2019.

[7]. X. Poda and O. Qirici, “Shape Detection and Classification Using OpenCV and Arduino Uno,” *RTA-CSIT*, vol. 2280, pp. 128–136, 2018.

[8]. “Image Processing – an Overview,” *ScienceDirect Topics*. [Online]. Available: https://www.sciencedirect.com/topics/engineering/image-processing

[9]. “Computer Vision: What It Is and Why It Matters,” *SAS*. [Online]. Available: https://www.sas.com/en_us/insights/analytics/computer-vision.html

[10]. “Feature Selection Using Statistical Tests,” *Analytics Vidhya*, Jun. 27, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/feature-selection-using-statistical-tests/

[11]. “Tesseract OCR in Python with Pytesseract & OpenCV,” *Nanonets*, Aug. 09, 2022. [Online]. Available: https://nanonets.com/blog/ocr-with-tesseract/

[12]. “Raspberry Pi 4 Model B Specifications,” *Raspberry Pi Official*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[13]. “5MP Raspberry Pi Camera Module v1.3 – Daraz,” [Online]. Available: https://www.daraz.pk/products/5mp-raspberry-pi-camera-module-v13-i203164330.html

[14]. [“Raspberry Pi Camera Module 5M (China),” [Online]. Available: https://bdspeedytech.com/index.php?route=product/product&product_id=1878

[15]. “Arduino Uno Specification,” *Tomson Electronics*. [Online]. Available:

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/25dec005

https://www.tomsonelectronics.com/blogs/news/arduino-uno-specification
[16]. "Odseven 50cm Raspberry Pi Camera Ribbon Cable," [Online]. Available: https://xuanyao.en.made-in-china.com/product/EdXQJbYvJfWL/
[17]. "L298N Motor Driver Module," *Instructables*. [Online]. Available: https://www.instructables.com/L298N-MOTOR-DRIVER-MODULE/
[18]. "Arduino Modules – L298N Dual H-Bridge Motor Controller," *Instructables*. [Online]. Available: https://www.instructables.com/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/
[19]. "24VDC Low RPM High Torque DC Planetary Gear Motor," *Made-in-China*. [Online]. Available: https://www.made-in-china.com/video-channel/sgmadamotor_PeumkWZvbYHU_24VDC-Low-Rpm-High-Torque-DC-Planetary-Gear-Motor.html
[20]. "MG996R Servo Motor Datasheet," *Components101*, Apr. 3, 2019. [Online]. Available: https://components101.com/motors/mg996r-servo-motor-datasheet
[21]. "Arduino Color Sensor TCS230/TCS3200," *Random Nerd Tutorials*, Apr. 25, 2017. [Online]. Available: https://randomnerdtutorials.com/arduino-color-sensor-tcs230-tcs3200/