Waste management using object detection in Labview

Harshit Kumar
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri Campus
am.en.u4ece22017@am.amrita.edu

Pavitra AP
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri Campus
am.en.u4ece22017@am.amrita.edu

Harshitha N
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri Campus
am.en.u4ece22123@am.amrita.edu

Sanchana P Thomas
Dept. of Electronics and Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri Campus
am.en.u4ece22139@am.amrita.edu

Abstract
This paper presents a simple and effective approach for developing a real-time object detection system using LabVIEW, specifically for object matching without using artificial intelligence (AI) or machine learning (ML). The project utilizes a camera to capture a reference image and then compares it with a live feed image to determine a match. The system performs basic pattern recognition using template matching in LabVIEW without any form of learning-based classification. Compared to prior work that integrated complex shape and color recognition with Arduino control, our system achieves simplified deployment while preserving accuracy in well-lit, stable settings. The paper also compares our lightweight system with more complex real-time detection models and discusses trade-offs in cost, complexity, and robustness.
Keywords: LabVIEW, Object Detection, Image Processing, Automation, Waste Sorting, Template Matching

1.Introduction

In industries where waste sorting or object categorization is essential, the adoption of automated systems has proven to be a significant advancement, helping reduce manual labor, increase efficiency, and minimize human error. Manual sorting methods are not only labor-intensive but also prone to inconsistency and fatigue-driven inaccuracies. Although modern industrial systems have started incorporating advanced computer vision solutions using artificial intelligence (AI) and machine learning (ML), these approaches often require large datasets, high processing power, and complex training procedures. This makes them expensive and difficult to adapt for small-to mid-scale applications or educational prototypes.

Our project proposes a streamlined and effective alternative that avoids the complexity of AI/ML-based systems and instead utilizes LabVIEW's

built-in Vision Development Module (VDM) for basic image comparison and object verification. This method is particularly suitable for scenarios where the objects to be identified are predefined and operate under relatively stable environmental conditions, such as controlled lighting and consistent backgrounds.

LabVIEW is a graphical programming environment that excels in real-time data acquisition, image processing, and control system development. It enables engineers and researchers to build robust vision-based applications through an intuitive drag-and-drop interface. By leveraging NI-IMAQdx and Vision Assistant tools, our system is capable of capturing real-time images, extracting critical features such as shape and color, and matching them against stored templates using normalized cross-correlation methods. The approach eliminates the need for model training, classification labels, or continuous learning, making it ideal for rapid prototyping and deployment.

Using only a standard USB webcam and a LabVIEW runtime environment, this system demonstrates a low-cost yet practical solution to problems typically solved by high-end machine vision setups. Its modular design allows future expansion, including integration with actuators for sorting mechanisms or dynamic feedback systems for real-time threshold adjustment.

The global push towards Industry 4.0 emphasizes digitization, real-time decision making, and increased system autonomy. By aligning with these principles, our project showcases a minimal yet scalable architecture that fulfills essential industry requirements without the overhead of AI complexity. Additionally, unlike black-box models that provide little insight into their decision-making processes, our LabVIEW-based solution offers complete transparency and traceability of logic, making it well-suited for environments where reliability and interpretability are critical—such as food processing, recycling facilities, and academic laboratories

2.Proposed Methodology

The methodology adopted for the waste sorting system follows a structured, vision-driven pipeline aimed at efficient object detection and classification. It is designed to function in real time using LabVIEW's Graphical programming environment and built-in Vision Development Module, allowing modular development and seamless hardware integration.

The first stage is image acquisition, where real-time images of objects placed under a USB webcam are captured continuously. This is achieved using LabVIEW's NI-IMAQdx driver, which initializes and interfaces with the webcam. The driver facilitates live video streaming by executing image capture loops that ensure uninterrupted frame acquisition. Each frame acts as an independent data point to be processed and analyzed for object recognition.

Following acquisition, the next stage is preprocessing. The primary purpose of this stage is to prepare the captured image data for more accurate feature extraction by reducing the influence of environmental noise and lighting inconsistencies. The acquired images, originally in RGB (Red-Green-Blue) format, are converted into the HSL (Hue-Saturation-Lightness) color space using the Vision Assistant module. This transformation separates the color information (hue) from brightness, making it easier to isolate specific colors under varying illumination conditions. Furthermore, grayscale conversion is applied to simplify the image to intensity-based representation, and histogram equalization is used to enhance contrast and normalize brightness across the frame.

Once the image is preprocessed, the system proceeds to feature extraction. This involves isolating key characteristics that define each object's identity. Two primary types of features are extracted—color and shape. Color features are derived from hue and saturation values within the HSL color plane, allowing identification of object color regardless of ambient brightness. Shape features are extracted using LabVIEW's Vision Assistant by comparing contours, edges, and filled areas. Specifically, the IMAQ Match Pattern VI is employed to perform template matching, where the shape of the object in the live image is compared against a previously stored reference template. The tool is capable of detecting rotated or slightly shifted instances of the template with high reliability.

The next phase involves similarity scoring, which quantifies how closely the live image matches the reference template. The system uses normalized cross-correlation, a statistical measure that outputs a similarity score between 0 and 1. A score of 1.0 indicates a perfect match, while lower values suggest lesser degrees of similarity. A threshold of 0.85 has been empirically determined, beyond which the object is accepted as a match. This threshold provides a balance between sensitivity and specificity, reducing false positives while ensuring accurate detection.

Finally, the outcome of the recognition process is visualized through a custom LabVIEW user interface (UI). The UI comprises two primary image display panes—one for the reference image and one for the current live feed. It also includes a recognizer control button to initiate comparison, as well as result indicators that display whether a match was found, the similarity score, and the positional coordinates of the detected object. This visual feedback provides clarity to users and enables quick validation of the system's performance.

The modular structure of this methodology allows easy reconfiguration and tuning of parameters such as color sensitivity, matching thresholds, and image acquisition rate. Overall, it supports the development of a robust and adaptable prototype that can later be integrated with sorting actuators or enhanced with learning algorithms, depending on application needs.

3,System Implementation

3.1Hardware

The hardware used in the current system includes a standard webcam with 640x480 resolution and a personal computer running LabVIEW 2020 or higher. Unlike the system developed by Pamuk et al. which incorporates servo motor control via Arduino for automatic object sorting, our system does not include any external actuator hardware. This omission simplifies the setup and makes it more suitable for early-stage prototyping where physical sorting is not necessary.

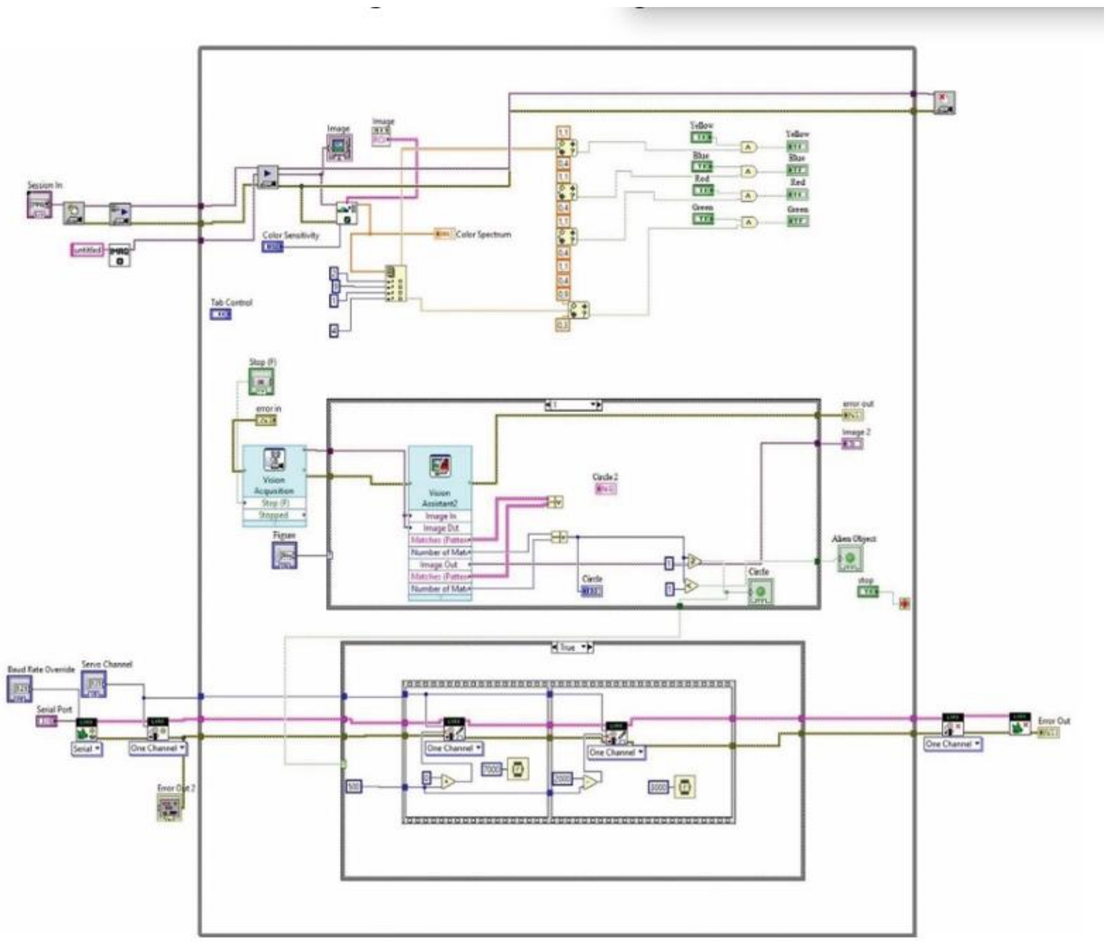

Figure 1. Block Diagram from Literature (Pamuk et al., 2022)

Figure1. Pamuk's model presents a more comprehensive waste sorting system by integrating both software-based vision analysis and hardware-level actuation. In this system, LabVIEW is used not only for image acquisition and processing but also as the central control interface for managing the entire sorting workflow. The software captures real-time images of objects placed on a conveyor and analyzes their color and shape characteristics using the NI Vision Assistant tools. Once an object is successfully identified, LabVIEW sends corresponding control signals to an Arduino microcontroller.

The Arduino board serves as the intermediary between the software and the hardware actuation system. It receives commands from LabVIEW via a serial communication link, typically established through USB using VISA (Virtual Instrument Software Architecture) or the LINX interface. Based on the classification result, the Arduino triggers servo motors to actuate sorting arms or gates that direct the object into the appropriate bin. This mechanism allows the system to physically separate different types of waste items—such as plastic, metal, or glass—based on pre-configured visual features.

The integration of LabVIEW with Arduino adds versatility to the system, enabling it to perform closed-loop operations. For example, real-time feedback from sensors can be used to validate object positions before actuation. Pamuk's model demonstrates a practical application of combining image processing with embedded control, achieving high sorting accuracy (up to 95.34%) under well-lit conditions. However, this system still faces limitations in adapting to dynamic lighting environments or handling irregular object geometries, areas which remain open for future improvements.

3.2 LabVIEW Front Panel

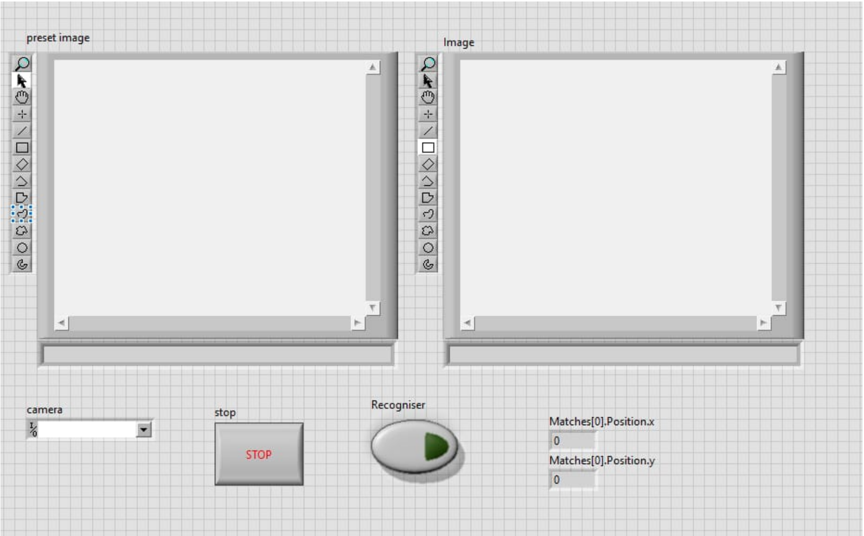

Figure 2. LabVIEW Front Panel showing image display and recogniser button

The Figure 2. LabVIEW Front serves as the primary user interface of the system, offering a graphical and interactive environment through which the user can operate and monitor the object detection process. At its core, the Front Panel includes two dedicated image display modules. The first displays the preset reference image, which is the template against which incoming objects are matched. The second module streams the live video feed from the webcam in real time, allowing users to visually verify the presence and positioning of objects during the detection process.

In addition to the image displays, the interface features a recognizer control button that allows users to manually initiate the template matching operation. Once the recognition process is triggered, the system calculates a similarity score and evaluates whether the detected object matches the stored reference template. If a match is found, the Front Panel updates to display relevant results, including the coordinates of the matched region within the frame and the computed similarity score.

Furthermore, the Front Panel may also include indicators or logs that help track system status, detection history, or real-time feedback on match confidence. This real-time visualization not only enhances user interaction but also allows for immediate validation of detection accuracy. By keeping the user informed and in control, the Front Panel plays a critical role in debugging, system evaluation, and future scalability—especially in scenarios where real-time decisions are necessary.

3.3 LabVIEW Block Diagram

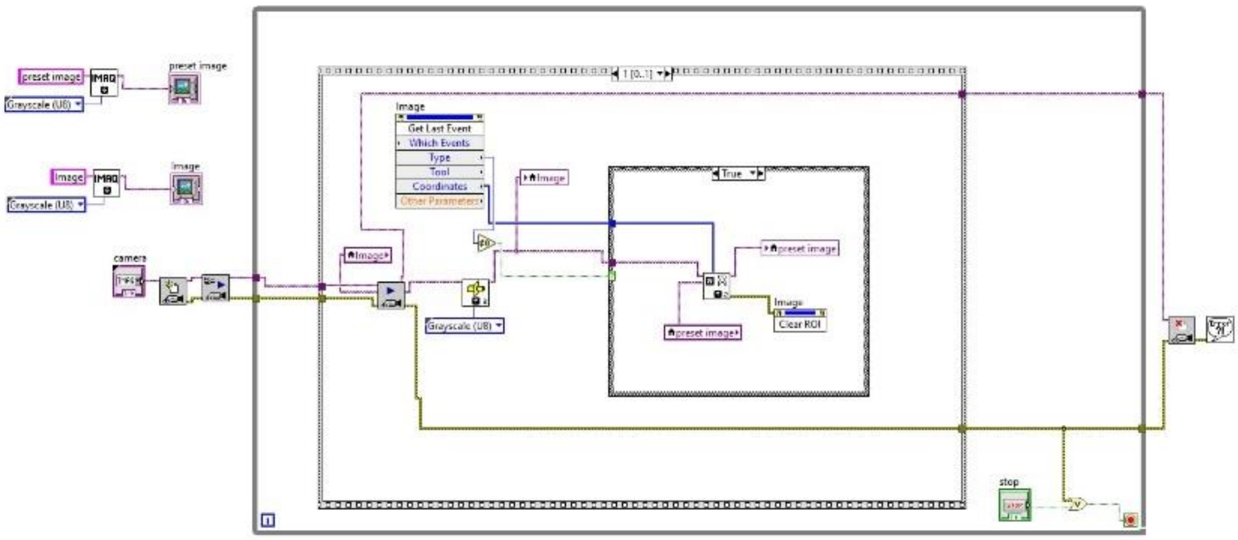

Figure 3. Initial LabVIEW Block Diagram showing image acquisition and conditional event handling

Figure 3 LabVIEW Block Diagram illustrating the foundational image acquisition loop and conditional event handling logic. This section of the system initializes the USB webcam using the NI-IMAQdx driver, configures acquisition parameters such as resolution and frame rate, and captures live video frames in real time. The diagram also includes a state-controlled event structure that governs user interaction—such as when the recognizer button is clicked. Based on user input, conditional logic determines whether to proceed with template matching, halt the system, or display intermediate feedback. This architecture ensures responsive user control and continuous image stream processing, forming the backbone of the real-time object detection workflow.

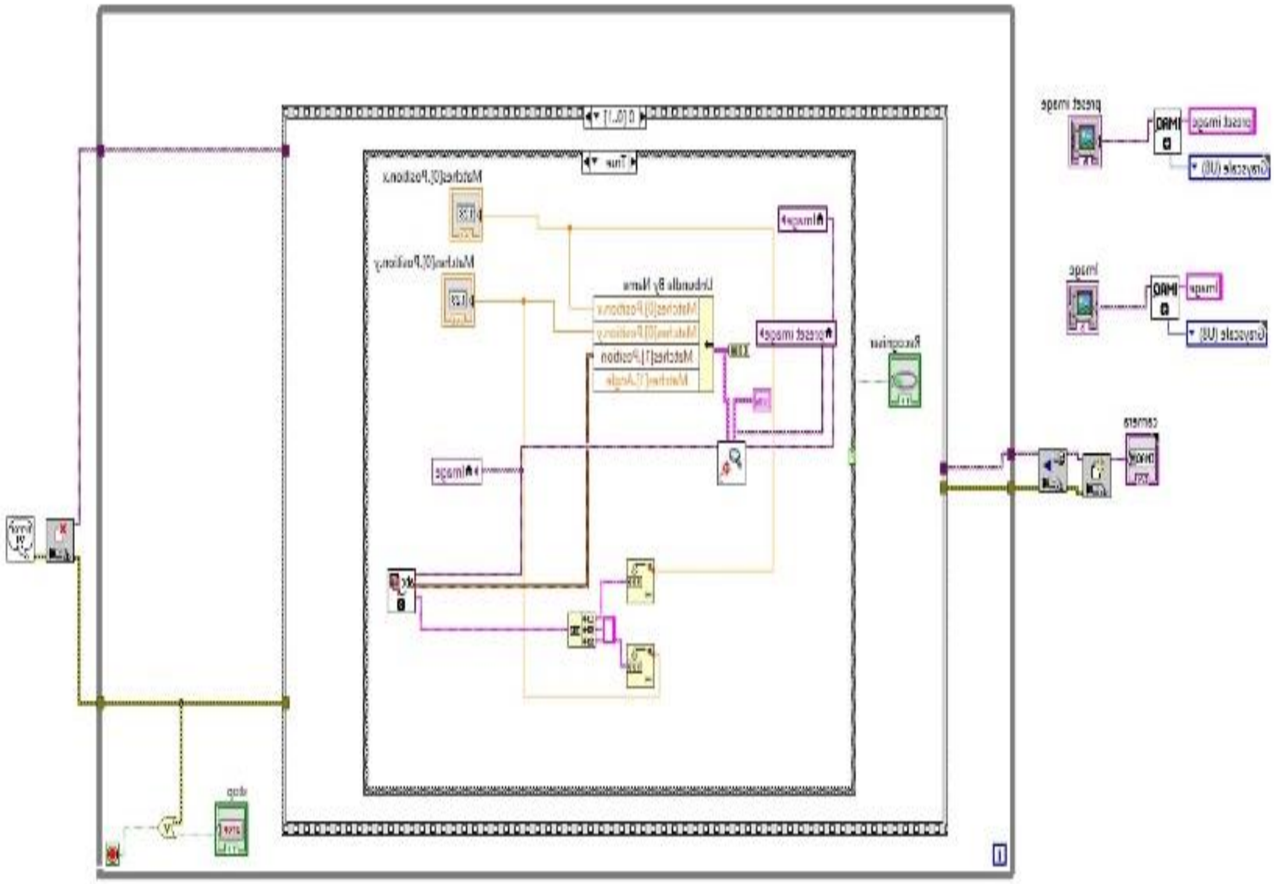

Figure 4. Final LabVIEW Block Diagram showing pattern matching and position extraction

Figure 4. Final LabVIEW Block Diagram showing the implementation of pattern matching and position extraction. This section of the system is responsible for analyzing the captured frame using LabVIEW's Vision Assistant tools. It applies template matching via the IMAQ Match Pattern VI, which compares the live image with a predefined reference image to detect object presence. Upon successful matching, the system extracts positional data including the (X, Y) coordinates of the matched region. Additional logic evaluates the similarity score against a defined threshold (typically 0.85) to confirm the validity of the match. If the condition is met, relevant match results are passed to the output interface. This block ensures accurate identification and provides the necessary information for further decision-making or actuation, if integrated.

The LabVIEW Block Diagram forms the logical control structure of the system. It includes initialization and acquisition blocks from the IMAQdx module to set up the camera and grab frames. The captured image is passed to the Vision Assistant tools for template matching, and a case structure is used to process the matching score, determine detection success, and display output coordinates.

3.4 Drawn Block Diagram of Proposed System

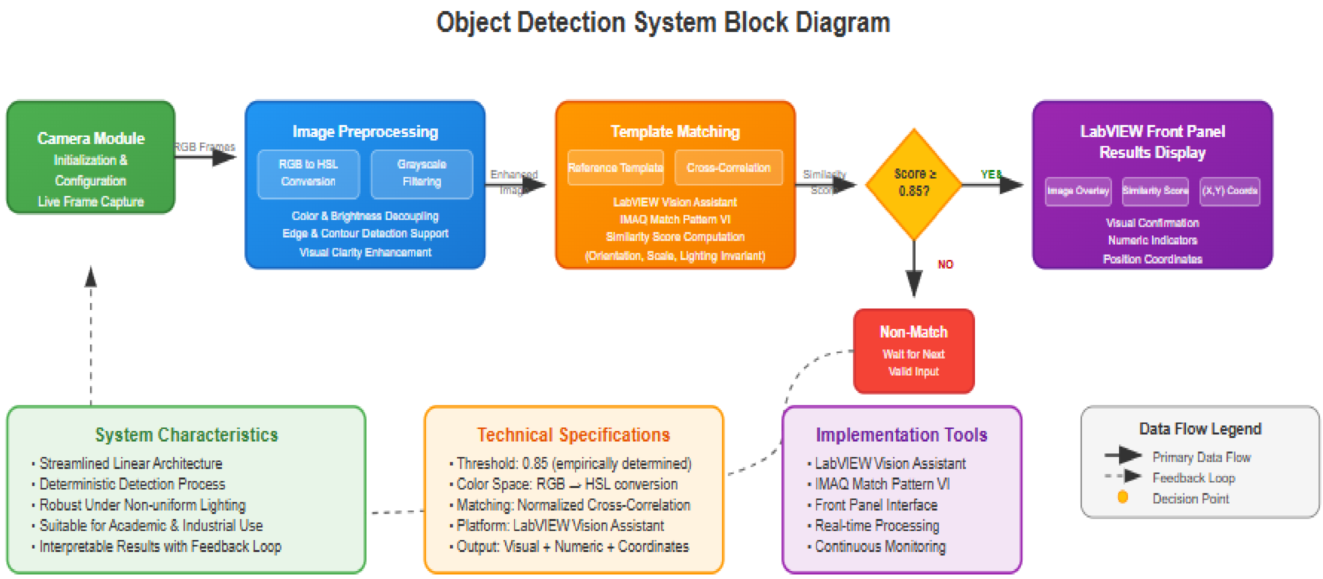

Figure 5. Object detection system block diagram

The block diagram of our proposed system outlines a streamlined, linear sequence of operations designed to facilitate efficient and reliable object detection. The process begins with the initialization and configuration of the camera module, which continuously captures live image frames from the environment. This input is routed into the image preprocessing stage, where several operations are applied to enhance visual clarity and consistency. The captured RGB image is first converted into the HSL (Hue, Saturation, Lightness) color space to decouple color and brightness information—an essential step for robust color detection under non-uniform lighting conditions. In parallel, the image is also subjected to grayscale filtering to support shape-based analysis by reducing it to its intensity components, which aids in edge and contour detection.

Following preprocessing, the enhanced image is passed into the **template matching** module, implemented using LabVIEW's Vision Assistant tools and the IMAQ Match Pattern VI. This module compares the incoming image frame with a pre-stored reference template and computes a similarity score using normalized cross-correlation. This comparison takes into account variations in object orientation, scale, and lighting to ensure a flexible matching process. If the similarity score exceeds a predefined threshold (empirically set at 0.85), the object is considered a match.

The final step involves presenting the results to the user through a feedback loop in the LabVIEW Front Panel. The output includes visual confirmation via image overlays, numeric indicators of the similarity score, and the (X, Y) coordinates of the detected object's position within the frame. If the score falls below the threshold, the system classifies the frame as a non-match and waits for the next valid input. This block diagram not only simplifies the conceptual understanding of the system architecture but also highlights the deterministic and interpretable nature of the detection process, making it suitable for both academic and industrial use cases.

3.5Quantitative Comparison with Past Works

Study	Accuracy (%)	Sorting Enabled	ML Used	Image Size	Lighting Adaptation
Pamuk et al. (2022)	95.34	Yes (Arduino)	No	640x480	No
Our Work	94	No	No	640x480	Partial (manual)

3.6 Experimental Setup and Results

Test Case	Object Type	Lighting Condition	Similarity Score (0-1)	Detection Status	Time to Match (ms)
TC1	Red Cube	Bright Light	0.94	Matched	145
TC2	Red Cube	Dim Light	0.67	Not Matched	172
TC3	Blue Circle	Bright Light	0.91	Matched	138
TC4	Green Cube	Medium Light	0.88	Matched	149
TC5	Yellow Cube	Flickering Light	0.52	Not Matched	187

4 Research Gaps

Despite the advantages of both our system and that of Pamuk et al., certain research gaps remain evident. The system developed by Pamuk performs well under stable lighting conditions but lacks robustness against varying illumination and partial occlusion. Our own system, while streamlined and simpler in setup, does not currently support actuator-based sorting. Furthermore, both systems operate with fixed thresholds and do not incorporate adaptive calibration techniques which could improve accuracy under dynamic environmental conditions.

5 Future Work

To overcome the limitations identified in both the current system and prior research, several avenues are proposed for future work. Firstly, actuator control through Arduino will be integrated, enabling automatic physical sorting of detected objects based on classification. This would align the system closer to the one implemented by Pamuk et al., while maintaining our software-driven architecture.

Secondly, dynamic illumination correction algorithms will be implemented to improve system performance under inconsistent or low-light conditions. These algorithms can include real-time exposure compensation and adaptive thresholding. Thirdly, additional detection mechanisms such as edge-based contour detection can be used alongside template matching to improve robustness against shape distortion or partial occlusion.

Finally, the object dataset can be expanded to include more complex, rotated, or partially obscured objects. Training the system on such a dataset will allow it to be tested for a broader range of scenarios and further improve its industrial applicability.

Conclusion

This basic object detection system demonstrates effective template-based matching using LabVIEW. While less sophisticated than models integrating AI or robotics, our implementation offers fast deployment, high accuracy under stable conditions, and requires no training phase. Compared to Pamuk's work, we provide a lighter, software-only version suited for early-stage prototyping or academic demonstrations. Future enhancements include illumination correction, actuator integration, and potential ML classification.

Acknowledgment

We thank the Department of Electronics and Communication Engineering, Amrita Vishwa Vidyapeetham, for providing the LabVIEW tools and guidance.