KT Journal of Mechanical Engineering

ISSN: 3048-9563
(Online)

Key Tech Services Publication, India

http://www.ktjme.com
https://doi.org/10.64188/3048956326102
Vol.: 3, No. 1, 2026, Pages: 10-17

A SOFTWARE-CENTRIC OBJECT DETECTION AND DECISION PIPELINE FOR VISION-GUIDED ROBOTIC SORTING USING YOLOv8

Yashraj Jadhav

[1] School of Mechanical & Materials Engineering, University College, Belfield, Dublin 4

Abstract

Vision-guided automation requires more than a camera and a robotic manipulator; it requires a reliable software pipeline that converts visual information into a structured decision for downstream robotic action. This paper presents a software-centric framework developed for object detection and decision support in a vision-guided robotic sorting application. The work focuses on the programming workflow rather than the mechanical design of the manipulator. The implemented approach combines Python-based development, computer vision, a YOLOv8 detection engine, and a dataset-driven machine-learning workflow. The software pipeline includes image acquisition, data preparation, model selection, training, validation, inference, object-label interpretation, and transfer of the detection outcome to the sorting logic. The project also considers the practical integration of the software layer with a simulated robotic environment developed using the 3DEXPERIENCE ecosystem and DELMIA. Representative objects such as an apple, banana, scissors, and mouse were used to demonstrate the recognition workflow. The study establishes a modular architecture in which the perception module can be modified independently of the robot motion and application logic. This separation improves maintainability, facilitates model replacement, and supports future integration with industrial automation systems. The contribution of the work is therefore a structured programming methodology for linking deep-learning-based object detection with robotic sorting decisions

1. Introduction

Received: 25.01.2026 Revised: 19.02.2026 Accepted:15.03.2026 Published Online: 10.04.2026

Keywords: Python, robotic manipulator, 3D experience, DELMIA, YOLOv8

*Corresponding author name and email: **Yashraj Jadhav, Yashraj.jadhav@ucdconnect.ie**

How to cite this article: Yashraj Jadhav, A software-centric object detection and decision pipeline for vision-guided robotic sorting using yolov8, KT Journal of Mechanical Engineering, 3 (1) 2026, 10-17. https://doi.org/10.64188/3048956326102

Industrial automation is increasingly shifting from fixed-rule machines toward systems capable of interpreting their environment and adapting their actions. In a conventional sorting process, an operator or a pre-programmed mechanism may be required to identify an object, determine its category, and initiate a corresponding handling operation. Such an arrangement becomes restrictive when object appearance, position, or product mix changes. Vision-based automation addresses this limitation by introducing a perception layer that can interpret image data and provide a machine-readable description of the observed scene [1–3]. The present work is based on a robotic sorting project in which a vision system is used to recognize objects and support automated manipulation. The earlier project work involved mechanical design, robotic simulation, kinematics, and application studies. The present paper deliberately concentrates on the programming and artificial-intelligence pipeline. The central question is how a software system can convert raw visual input into a structured object-detection result that can be consumed by a robotic decision layer.

The programming problem is significant because the success of a vision-guided robot depends not only on the neural-network model but also on the complete data and software lifecycle. Image collection, labeling, preprocessing, training, validation, model inference, confidence handling, and communication with the automation logic must operate as a coordinated pipeline. A model with high recognition capability can still produce unreliable automation if the data flow and decision logic are poorly designed. The work reported in this paper therefore proposes a modular software architecture using Python, TensorFlow-based machine-learning concepts, YOLOv8, and the COCO 2017 dataset as part of the development ecosystem. The objective is to demonstrate how a programmable perception module can be integrated with a robotic sorting workflow. Unlike a general study of robotic manipulators, the emphasis here is on the computational sequence from image acquisition to object classification and decision generation.

1.1 OBJECTIVES OF THE STUDY

- To develop a structured software pipeline for image-based object detection in a robotic sorting application.
- To examine the role of dataset preparation, preprocessing, model training, and inference in an automated perception system.
- To integrate YOLOv8-based detection with a decision layer capable of generating sorting information.
- To establish a modular interface between the perception software and a simulated robotic automation environment.
- To identify practical programming challenges associated with real-time vision-guided automation.

2. Literature Review

Vision-based robotic systems combine image acquisition, visual interpretation, planning, and actuation. Early machine-vision systems commonly depended on manually designed features, thresholding, geometric rules, and colour segmentation. Such techniques can be effective in controlled environments but may become sensitive to illumination, background variation, object orientation, and changes in the appearance of products [4,5]. Deep-learning methods have changed the design of object-recognition systems by allowing discriminative features to be learned directly from image data. Convolutional neural networks (CNNs) have been widely used for image classification and detection because their hierarchical feature extraction can represent edges, textures, shapes, and higher-level object patterns [6,7]. Object detection differs from simple image classification because the system must identify both the object category and its spatial location in the image. For robotic applications, detection must be connected to a control or decision process. A detected object may be represented by a class label, confidence score, and bounding-box coordinates. This structured output can then be used by application logic to determine whether an object should be picked, rejected, stored, or sent to a particular destination [8,9]. The separation between perception and action is important because it allows the detection model to be improved without redesigning the complete robotic mechanism. YOLO-family detectors are particularly relevant to real-time applications because detection is formulated as a unified prediction process. The YOLOv8 engine provides a practical development route for integrating object detection into Python applications. In the present project, the software layer is considered as an independent computational module that supplies object information to a robotic sorting workflow. A further requirement is reproducibility. A practical machine-learning pipeline should define the dataset, preprocessing sequence, model configuration, training process, validation procedure, and deployment method. The project material also identifies the use of Python, Visual Studio Code, TensorFlow, YOLOv8, and the COCO 2017 dataset as part of the software ecosystem. These tools enable a workflow that can be extended to new object classes and application-specific datasets.

2.1 Research Gap and Contribution

Much of the conventional discussion of robotic sorting focuses on mechanical architecture, manipulator degrees of freedom, gripper design, or industrial application. The specific software engineering problem of converting an object-detection model into a modular robotic decision pipeline receives comparatively less attention in undergraduate engineering project studies. The contribution of the present work is a programming-focused formulation of the perception-to-decision chain. The paper emphasizes data flow, modularity, model lifecycle, and the interface between computer vision and robotic automation rather than repeating the mechanical design study of the associated project.

3. Methodology

The proposed methodology is organized as a sequence of computational stages. The stages are designed so that each module has a clear function and can be tested independently. The complete

workflow consists of data acquisition, dataset preparation, image preprocessing, model selection, training, evaluation, inference, decision generation, and integration with the robotic application.

3.1 Software Architecture

The software architecture is divided into five functional layers: (i) image acquisition, (ii) perception and object detection, (iii) interpretation of detection results, (iv) sorting decision logic, and (v) robotic application interface. The image acquisition layer provides input frames. The perception layer executes the trained YOLOv8 model. The interpretation layer extracts the detected class and associated confidence information. The decision layer maps the detected object to an application-specific destination. The interface layer provides the information required by the simulated or physical robotic operation. This modular structure avoids coupling the machine-learning model directly to the complete robot program. Consequently, a model can be retrained or replaced while retaining the application logic. Similarly, the same detection module can be reused for multiple sorting scenarios by changing the mapping between detected classes and destinations.

3.2 Dataset Preparation

The first stage of the programming workflow is the preparation of a labeled image dataset. Each image must be associated with the object class that the system is expected to recognize. The project workflow considered representative object classes including apple, banana, scissors, and mouse. The COCO 2017 dataset was included in the development ecosystem as a widely used source of annotated object imagery, while application-specific data can be added for improved adaptation to the target environment. The dataset must contain sufficient variation in object scale, orientation, position, illumination, and background. A dataset containing only nearly identical images may produce a model that performs well during development but poorly when exposed to new scenes. Data augmentation, including rotation, scaling, flipping, and other suitable transformations, can increase the diversity of the training samples and reduce overfitting.

3.3 Image Preprocessing

Image preprocessing establishes a consistent input format for the learning system. The principal operations include image resizing, pixel normalization where required, format conversion, and dataset organization. For object detection, the image and its annotation must remain spatially consistent. Any transformation applied to the image must be reflected in the corresponding annotation data. The preprocessing stage is also important during deployment. The input received by the inference program must undergo the same essential preparation used during model development. Differences between training-time and deployment-time preprocessing can reduce detection reliability and create inconsistent system behavior.

3.4 Model Training and Validation

The model-training stage uses labeled images to optimize the parameters of the detection network. The dataset is separated into training, validation, and testing subsets. The training subset is used for parameter learning, the validation subset supports model and hyperparameter decisions, and the test subset provides an independent assessment of generalization. The training workflow includes model initialization, configuration of training parameters, iterative optimization, and monitoring of validation performance. The relevant performance measures may include precision, recall, F1-score, and detection accuracy. For a robotic application, the confidence threshold should also be considered because false detections may lead to incorrect sorting actions.

3.5 Yolov8-Based Inference

During inference, an input image or video frame is passed to the trained detector. The model returns detection information that may include the predicted class, confidence value, and bounding-box coordinates. The application program then filters or interprets these detections according to the sorting task. A conceptual inference sequence is represented as follows:

Image frame → YOLOv8 inference → detected class and confidence → decision rule → destination/action command

The separation of detection and decision logic is a key design feature. The detector answers the question “what is present in the image?”, whereas the decision layer answers the question “what should the system do with the detected object?” This distinction supports maintainability and enables the same detection model to be used in multiple applications.

3.6 Decision Logic for Robotic Sorting

The detected class is converted into a sorting instruction through a mapping table or conditional decision structure. For example, a recognized object may be assigned to a designated bin, rack location, or packaging destination. The decision module may also reject a detection when the confidence value is below a predefined threshold. This prevents uncertain detections from being immediately converted into physical actions. The decision layer can be extended to include multiple detections, priority rules, object coordinates, and application-specific constraints. In a conveyor-based application, the detection result may be associated with a moving object and subsequently used to coordinate the timing of robotic manipulation. The software architecture therefore provides a bridge between perception and motion planning without embedding the complete robot kinematics inside the detection model.

3.7 Software Tools

- Python: primary programming language for image-processing and machine-learning workflow

development.

- Visual Studio Code: development environment for program creation, testing, and debugging.
- TensorFlow and related machine-learning tools: used within the development ecosystem for model training and evaluation concepts.
- YOLOv8: object-detection engine for identifying objects in image data.
- COCO 2017 dataset: reference dataset used in the software-development ecosystem for object-recognition tasks.
- 3DEXPERIENCE/DELMIA: simulation environment supporting the broader robotic automation workflow.

4. Results and Discussion

The software workflow successfully established the computational sequence required for vision-guided sorting. The developed approach demonstrated that object recognition can be treated as a modular software service rather than as an inseparable component of the mechanical robot. The detection stage produced structured visual information that could be interpreted by application logic for subsequent sorting decisions.

4.1 Object Recognition Demonstration

The project demonstration included recognition of representative objects such as an apple, scissors, banana, and mouse. These examples illustrate the operation of the detection pipeline across objects with substantially different visual characteristics. The recognition outputs were used to demonstrate the transition from image interpretation to a machine-readable object identity.

4.2 Programming Pipeline Evaluation

The principal outcome of the study is the establishment of a repeatable programming pipeline. The sequence of data preparation, preprocessing, model training, validation, inference, and decision generation provides a systematic approach for extending the system to new object categories. The modular arrangement also simplifies troubleshooting because errors can be localized to a specific stage of the workflow. The software architecture provides three practical advantages. First, the model can be updated without redesigning the complete robotic system. Second, the decision rules can be modified for different sorting applications without retraining the detector when the visual classes remain unchanged. Third, the same detection output can support alternative downstream actions, including bin placement, storage allocation, or packaging.

4.3 Practical Challenges

- Detection performance may be affected by illumination, background clutter, occlusion, and object orientation.

- A high-confidence detection is not automatically equivalent to a successful robotic grasp; perception and manipulation must be coordinated.
- Real-time operation requires appropriate balance between detection accuracy and computational latency.
- Changes in the object population or operating environment may create data drift and require model updating.
- The interface between the detection program and the robotic controller must be designed to prevent uncertain detections from generating unsafe or incorrect actions.

4.4 Comparison with the earlier Project Focus

The associated project work included mechanical design of the manipulator, gripper development, kinematics, material selection, DELMIA simulation, and industrial use cases. The present paper intentionally does not reproduce those areas as its primary contribution. Instead, it treats the robot as the application platform and studies the programming layer responsible for perception, model inference, and decision generation. This distinction creates a separate technical contribution from a software-engineering and artificial-intelligence perspective.

5. Conclusions

A modular software pipeline for vision-guided robotic sorting was formulated and examined using Python-based development and YOLOv8 object detection. The study demonstrates that the programming architecture should be designed as a sequence of independent but connected modules: dataset preparation, preprocessing, model training, inference, detection interpretation, and sorting decision generation. The use of a separate decision layer allows the visual-recognition model to be updated without requiring complete redevelopment of the robotic application. The programming-focused approach provides a distinct contribution from mechanical studies of the associated robotic manipulator. Representative object-recognition demonstrations confirmed the practical operation of the perception workflow, while the modular architecture provides a basis for future extension to additional classes, improved datasets, real-time camera streams, and more advanced robotic control interfaces. The principal conclusion is that reliable vision-guided automation depends on the integration of machine-learning performance with disciplined software architecture and clearly defined interfaces between perception and action.

Acknowledgements

The authors acknowledge the support from the Dassault Systèmes company for mentoring the project.

References

1. B. S. Pachaiyyapan and T. Sridhar, Design, and analysis of an articulated robotic arm for various industrial application, 2015
2. FANUC America Corporation, Automated Bottle Pallet Unloader with FANUC Pick and Place Robot Clear Automation, 2016
3. Govender P. ANN Based Intelligent Vision and Sorting System, IEEE International Conference on Industrial Engineering & Engineering Management, 2010.
4. Ke X, Weng Z. Workpieces sorting system based on industrial robot of machine vision International Conference on Systems & Informatics, 2017.
5. Weyrich M, Wang Y, Winkel J, et al. High-speed vision based automatic inspection and path planning for processing conveyed objects, Procedia, 3, 442-447, 2012.
6. Chen J. Z., Wang G. T., Chen J. Q., Automatic inspection system of small plastic gears based on machine vision, Key Engineering Materials, 2012, 522, 628-633.
7. Arko Djajadi, Fiona Laoda, Rusman Rusyadi, Tutuko Prajogo, Maralo Sinaga, A model vision of sorting system application using robotic manipulator, Indonesian Journal of Electrical Engineering, Vol. 8 (2), 2010, 137-147.
8. I. Andras, E. Mazzone, F. W. B. van Leeuwen, G. De Naeyer, M. N. van Oosterom, S. Beato, T. Buckle, S. O'Sullivan, P. J. van Leeuwen, A. Beulens, N. Crisan, F. D'Hondt, P. Schatteman, H. van der Poel, P. Dell'Oglio and A. Mottrie, Artificial intelligence and robotics: a combination that is changing the operating room, *World Journal of Urology*, 38(10) (2020) 2359–2366
9. S. Rege, S. Khot and A. Kulkarni, 2D geometric shape and color recognition using digital image processing, *Int. Jou of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2(6) (2013) 2479–2487.
10. Chinmay B. Patil, Ashwin K. Bhakare, Sanket B. Wakchaure, Harsh M. Rade, Study of vision-based object sorting robot manipulator, KT Journal of Mechanical Engineering, 2(2), 2025, 39-47. https://doi.org/10.64188/3048956325023
11. Z. Chen, H. Zou, Y. Wang, H. Liu and S. Xie, A vision-based robotic grasping system using deep learning for garbage sorting, *Proceedings of the 36th Chinese Control Conference (CCC)*, (2017) 11201–11206.
12. R. Kumar, A. Sharma and S. Kumar, Object detection and recognition for a pick and place robot, *Proceedings of the Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, IEEE, (2014) 1–6.
13. X. Yu, S. Yu, J. Zhang, Y. Wei and J. Zhang, Novel SGH recognition algorithm-based robot binocular vision system for sorting process, *Journal of Sensors*, (2016) Article ID 2498197, 1–8.
14. M. F. Ashby, Materials Selection in Mechanical Design, Elsevier Butterworth-Heinemann, Oxford, 2005.