In: 2024 IEEE 22nd International Conference on Industrial Informatics (INDIN), 18-20 August 2024, Beijing, China
DOI:10.1109/INDIN58382.2024.10774307

Towards Robot Software Abstraction: ROS 2-based Framework for Object Handling within a Robot Cell

Mikel Bueno Viso
Centre for Robotics and Assembly
Cranfield University
Cranfield, United Kingdom
Mikel.Bueno-Viso@cranfield.ac.uk

Jingjing Huang
Centre for Robotics and Assembly
Cranfield University
Cranfield, United Kingdom
jingjing.huang.013@cranfield.ac.uk

Seemal Asif
Centre for Robotics and Assembly
Cranfield University
Cranfield, United Kingdom
s.asif@cranfield.ac.uk

Fahad Khan
Centre for Robotics and Assembly
Cranfield University
Cranfield, United Kingdom
Fahad.Khan@cranfield.ac.uk

Phil Webb
Centre for Robotics and Assembly
Cranfield University
Cranfield, United Kingdom
p.f.webb@cranfield.ac.uk

***Abstract*— Recent advancements in industrial automation have led to the development of increasingly adaptable and reconfigurable systems, driven by the necessity for flexibility and efficiency in manufacturing. This paper introduces a ROS 2-based software framework tailored for object handling within a robot cell, addressing challenges pertaining to reconfigurability, modularity, and interoperability. The proposed solution simplifies the deployment process of robotic applications and provides a standardized ROS 2-based platform, making it particularly beneficial for small and medium automation enterprises that require frequent reprogramming and adaptation of robot cells to different settings. By ensuring software and hardware agnosticism, the framework presents a comprehensive pipeline for designing, developing, and deploying object pick-and-place applications, demonstrated through an automated kitting use-case where a robot sorts industrial spare parts from a conveyor belt using a simple camera. The system integrates real-time object detection and classification capabilities with ROS 2 to facilitate effective communication between perception and robot action. This research advocates for open-source solutions and collaboration, aiming to enhance the adaptability and efficiency of robotic solutions across diverse industrial applications.**

Keywords—Reconfigurable Manufacturing, Intelligent Robotics, ROS 2 Framework, Industrial Automation, Modular Systems, Flexible Systems, Object Detection, Robot Pose Estimation

I. INTRODUCTION

In recent years, industrial automation has shifted towards adaptable and reconfigurable systems, driven by advancements in Cyber-Physical Systems (CPS) and the need for flexibility and efficiency in manufacturing [1]. Central to this shift is the principle of modularity, advocating for breaking down complex systems into smaller, interchangeable modules that can easily adapt to changing production needs [2]. A modular system structure would empower robot manipulators with increased accessibility, enhanced flexibility, and the ability to swiftly adapt to evolving production requirements. However, the challenges of robot modularity persist in the industry. Leading manufacturers dictate interfacing protocols and software tools tailored to specialized requirements, resulting in highly proprietary robotic hardware and software unsuitable for use by other manufacturers or companies, leading to closed supply chains, hindering interoperability and innovation [3]. Open-source solutions and broader collaboration present opportunities to address this challenge, promoting interoperability and innovation within the industry and academia.

In fact, in the evolving industrial automation landscape, characterized by a growing demand for flexible and adaptable production lines, traditional robot manipulators lag behind. Constrained by rigid programming and task-specific configurations, these manipulators underscore a significant gap in meeting the agile manufacturing requirements of today. Cell-level robot programming tasks are restricted to defining robot motion and program logic through GUIs or teach pendants, and integrating intelligence into robot controllers necessitates complex systems and reliance on PLCs, thereby introducing considerable complexity and cost [4]. This disparity, added to the closed robotics and automation market, underscores the critical necessity for adaptable and accessible robotics solutions.

Object handling stands as a foundation in industrial robotics due to its widespread application across various industries. While current specific robotic systems demonstrate satisfactory performance in handling various industrial objects, or fulfilling specific pick & place applications, the construction of a generalized robotic system remains challenging [5]. The aim of this research work is to address this gap, by developing a standardized framework, which would facilitate the handling of different objects within a robot cell while ensuring software and hardware agnosticism. With this system, we aim to offer significant potential for automation companies, especially small and medium-sized enterprises, seeking adaptable and efficient robotic solutions for various industrial applications.

The proposed work shows the methodology and the techniques used to develop and implement a standard ROS 2-based object handling robotic cell framework, and is evidenced with an autonomous robotic kitting cell, capable of picking industrial spare parts from a conveyor belt and placing them on

This research is supported by EPSRC grant (EP/V051180/1) for the Reconfigurable Robotics for Responsive Manufacture - R3M Project. The source code, testing results and videos are available in IFRA-Cranfield's GitHub: https://github.com/IFRA-Cranfield/ur10e_ConveyorPickPlace.

© 2024 IEEE. This is the Author Accepted Manuscript issued with: Creative Commons Attribution License (CC:BY 4.0).
The final published version (version of record) is available online at DOI:10.1109/INDIN58382.2024.10774307.
Please refer to any applicable publisher terms of use.

a kitting tray according to their type and color. The use-case demonstrator is illustrated in Fig. 1. The primary challenge tackled by this research lies in the intricate interplay of computer vision, robotics, and industrial automation. It focuses on accurately identifying moving objects on conveyor belts, categorizing them in real-time, and seamlessly integrating this system with ROS 2 for effective communication between perception and robot action. The contributions of the proposed system are listed as follows:

- A robot-agnostic, ROS 2-based open-source software framework that provides essential tools for operating any ROS 2-supported robot manipulator in both simulation and real-world environments, enabling seamless sim-to-real transition and straightforward integration of external software and hardware components.
- A streamlined pipeline to design, develop, and deploy any object pick-and-place application using our ROS 2-based framework. This includes the configuration of both the robot manipulator and end-effector, design and development of object detection and pose estimation modules, and the integration and interoperability of every module within the system.
- The automated kitting application, where a UR10e robot equipped with a custom electromagnetic gripper and assisted by a simple web camera, efficiently picks and sorts industrial spare parts from a conveyor belt. This use-case serves as a demonstration of the intuitive design and deployment process of complex robotic tasks using our approach.

II. Related Work

A. Reconfigurable Robotics for Object Manipulation Tasks

Recent literature highlights the widespread use of robot manipulators for object manipulation tasks, showcasing advancements in various applications. These advancements encompass enhancements in dexterity [6], precision, and efficiency achieved through innovative gripper designs [7], control algorithms, and sensor integration techniques. Moreover, research efforts have explored the integration of machine learning and computer vision technologies to augment autonomy and adaptability in robotic systems [8], [9]. However, despite these strides, the construction of a generalized robotic system remains a formidable challenge. While existing specific systems exhibit satisfactory performance in fulfilling industrial requirements or addressing specific pick-and-place applications with high grade of customization, the development of a versatile system presents inherent complexities [5].

Recent works emphasize the pivotal role of standardized software frameworks [10], generalized middleware [11], and modularity in facilitating reconfiguration and adaptability in robotic systems. Efforts have also been made to develop standardized procedures aimed at reducing the deployment process when the environment or requirements change within the system [12]. These initiatives reflect a discernible trend towards hardware and software abstraction. Implementing these concepts would significantly contribute to the realization of a generic and reconfigurable robot manipulator cell.

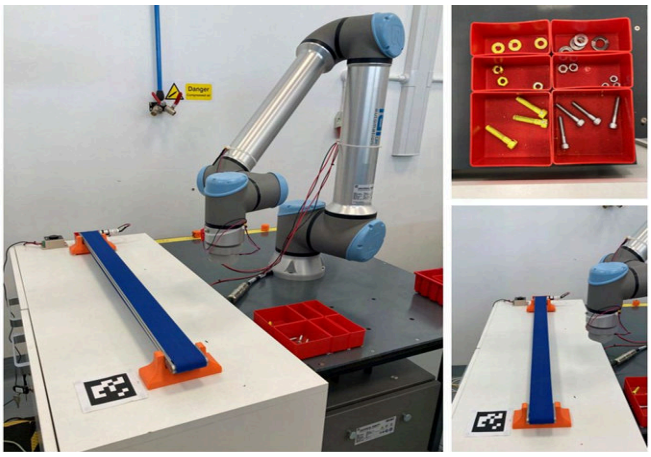

Fig. 1. Automated Kitting Station at Cranfield University Intelligent Automation Lab. Cell set-up on the left (A), with sorting slots on the top-right (B) and the conveyor belt on the bottom-right (C).

B. Object Detection in Robotic Systems

Object detection and classification techniques, essential for determining object location within a Robot Cell, often rely on calibration techniques, crucial for post-processing images. Visual markers, as exemplified in [13] and [14], facilitate precise distance calibration, aiding accurate object position estimation for effective picking. These techniques are pivotal in addressing the complexity of robotic tasks like Bin Picking.

Among object detection algorithms, YOLO (You Only Look Once) stands out, dividing images into grids to predict bounding boxes and class probabilities concurrently [15]. In robot cells, YOLO, particularly its latest version YOLOv8, autonomously analyzes camera feeds to detect and classify objects. Its versatility has been harnessed for various robot-object manipulation tasks, with advancements proposed for detecting smaller-sized objects or predicting object orientation alongside position using the YOLO-OBB labelling method [16].

C. ROS 2 as Software Middleware for Robotic Systems

ROS 2 is an open-source framework providing a flexible and modular platform for developing robotic systems [17]. Particularly suited for applications requiring interaction between diverse hardware and software modules [9], ROS 2 facilitates the implementation of modular architectures, enabling the adaptation of robot manipulators for a variety of tasks without starting from scratch [18]. Overcoming limitations inherent to robots' interfaces, communication protocols, and programming languages, ROS 2 simultaneously enables perception, recognition, and re-planning capabilities [9] and facilitates rapid development and deployment of new software and hardware modules, as well as system setup and reconfiguration [19]. Its flexibility, scalability, and modularity make ROS 2 an ideal middleware solution for reconfigurable systems, particularly beneficial in contexts where robot manipulators frequently require reprogramming for diverse operations [20]. With ROS 2's versatility, seamless integration of perception systems for object handling in reconfigurable robot cells becomes achievable.

III. Methodology

A. Object Detection and Classification

In our proposed methodology for object detection and classification within a robot cell, we utilize a customized YOLOv8 model trained using bespoke datasets labelled with the RoboFlow annotation tool. This approach enables accurate detection and classification of objects captured by a simple camera setup within the robot cell environment. The integration of RoboFlow simplifies dataset creation by providing a user-friendly platform for annotating images with bounding boxes and assigning corresponding object classes. This ensures the accuracy of the training dataset, essential for YOLOv8 to accurately detect and classify objects within the robot cell environment. RoboFlow's intuitive interface seamlessly incorporates labeled data into the YOLOv8 training pipeline, streamlining model training for various scenarios and use-cases.

YOLOv8's simplicity and ease of implementation make it accessible to users with varying expertise levels. Its intuitive training & testing pipeline enables users to customize the model to suit specific requirements and environments, facilitating real-time detection and classification of objects. Within the robot cell, YOLOv8 can operate on a per-frame basis, generating bounding boxes and classifying objects for each camera frame captured. This real-time detection and classification capabilities are essential for timely and accurate identification of objects within the dynamic environment of the robot cell, enabling seamless integration into automated processes.

B. Object Tracking: 2D-Coordinates using a USB Camera

Object tracking and precise localization are essential components of robotic manipulation tasks, enabling robots to interact with objects effectively within their workspace. In this section, we present a method for estimating the 2-Dimension pose coordinates of objects detected within a robot workspace. Our approach leverages a standard web camera and an ArUco marker within the workspace to precisely calculate the 2D pose coordinates of detected objects. We perform 2D pose estimation under the assumption that objects to be detected are placed on a flat surface with a known height. Even though the depth of the detected objects is calculated throughout the process, this assumption limits the pick-and-place task to a flat surface, simplifying the robot's end-effector pose estimation process.

To perform the pose estimation task, knowledge of the camera's calibration parameters, including the camera matrix and distortion coefficients, is imperative. This calibration process entails several steps: first, the creation of an ArUco board, followed by capturing images of the board from various distances and angles (Fig. 2). Subsequently, ArUco marker detection is conducted for each image, leading to the estimation of internal parameters such as focal length, optical center, and distortion, as well as external parameters like rotation and translation vectors for each image. Following parameter estimation, distortion correction and refinement procedures are applied. Finally, the resulting calibration parameters are saved to a file for utilization throughout the pose estimation process.

In our method, an ArUco marker is placed within the robot's workspace, and the object's 2D pose is calculated relative to this marker. The process begins with the detection of the ArUco marker, and upon successful detection, the marker's pose is calculated using OpenCV's *aruco.estimatePoseSingleMarkers()* function [13]. Subsequently, the 2D pose of the detected object is determined using the bounding box provided by YOLOv8.

The (x_0, y_0) 2D-coordinates of the box are calculated using the formulas below, where (CP_{x_BB}, CP_{yBB}) represent the center of the object's bounding box detected by YOLOv8 in pixel coordinates, (c_x, c_y) represent the optical center of the image, and (f_x, f_y) represent the focal lengths across the x and y axes. With the known z coordinate of the ArUco marker and the vertical focal length f_y, as well as the vertical principal point c_y, the depth of the object, denoted as z_0, is computed (1). Subsequently, the object's x and y coordinates are determined (2), (3).

$$z_0 = (CP_{yBB} - c_y) \times \frac{z_{ARUCO}}{f_y} \quad (1)$$

$$x_0 = (CP_{xBB} - c_x) \times \frac{z_0}{f_x} \quad (2)$$

$$y_0 = (CP_{y_BB} - c_y) \times \frac{z_0}{f_y} \quad (3)$$

The (x_0, y_0) 2D-coordinates obtained represent the object's 2D-pose relative to the ArUco marker. For this information to be used by the robot, a simple 2D coordinate transformation from relative to global values is done, by applying the ArUco marker's fixed position within the robot's workspace.

C. ROS 2: Robot Simulation and Control Framework

We have opted for ROS 2 as the middleware solution for our Robot Simulation and Control framework, primarily due to its inherent flexibility, scalability, and modularity. These attributes make it well-suited for accommodating the needs of flexible and reconfigurable systems, particularly in contexts where robot manipulators are required to execute diverse operations and undergo frequent reprogramming. In our specific scenario, where two different external software modules (YOLOv8 and OpenCV) and a hardware module (USB web camera) need to be seamlessly integrated and communicate with the robot, ROS 2 emerges as the most suitable solution. Its capacity to facilitate effective communication and coordination among various components ensures the smooth interaction and operation of our system.

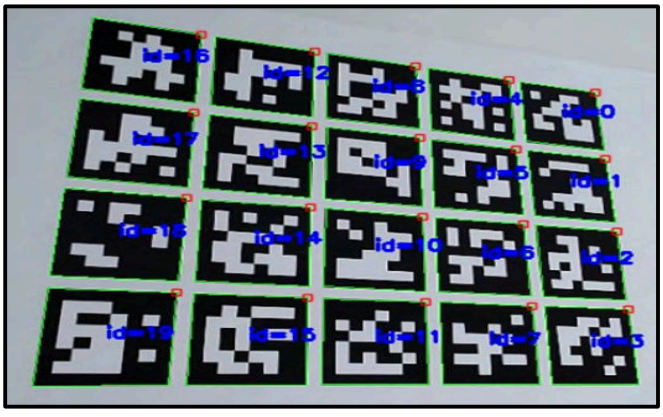

Fig. 2. ArUco Marker Calibration Board. An ArUco marker is a distinctive fiducial marker used in computer vision for object tracking and pose estimation. It features a black border surrounding a pattern of white squares, facilitating easy detection and recognition by computer vision systems.

MoveIt!2 is a powerful software framework designed for motion planning, manipulation, and control of robot manipulators in ROS 2. Notably, MoveIt!2 facilitates a seamless transition from simulation to real-world scenarios, ensuring consistency in the operation and control of robot manipulators across both environments. This feature empowers users to rigorously test and validate robotic use-cases prior to their deployment in real-world settings. Moreover, MoveIt!2 promotes robot agnosticism by providing a uniform platform for robot manipulators from various manufacturers, irrespective of the protocols and languages they support. This standardized tool enables users to interact with different robotic systems in a consistent and streamlined manner, enhancing overall efficiency and versatility in robotic applications, without being tied to any specific hardware or software platform.

In addition to MoveIt!2, ROS 2 provides a robust communication platform facilitating seamless interaction among various system modules. Each module, encapsulated as a ROS 2 Node, performs a specific task and communicates to different elements of the system using the native ROS 2 communication mechanisms, comprising Topics, Services and Actions. By leveraging ROS 2's communication infrastructure, and combining it with Gazebo and MoveIt!2, we have created a cohesive platform for efficient and standardized robot arm operation in both simulation and real world.

D. ROS 2-based System Architecture

Our ROS 2-based Robot Simulation and Control Framework enables the operation of any robot arm that supports ROS 2 connectivity. In our system, the robot is considered as an independent exchangeable module, meaning that, no matter the robot's manufacturer, model or variant, as long as it is supported by ROS 2, it can be controlled by MoveIt!2, and is fully functional within our ROS 2-based system.

Fig. 3 illustrates the standard Software Architecture of our Framework, including the robot, Gazebo, MoveIt!2, our custom communication platform, and the external YOLOv8 and OpenCV modules. The system is divided into 3 main parts: The Robot Simulation and Control ROS 2 Packages, the ROS 2 Middleware as the main communications and integration platform, and the additional Hardware/Software components.

For the seamless simulation and control of robot arms using ROS 2, we have adapted and improved its default tools (Gazebo, MoveIt!2 and ROS 2 control algorithms) to develop a standardized pipeline that is agnostic to specific robot models or manufacturers. This development is based on 3 main ROS 2 packages. Firstly, the *Gazebo package* contains robot information in .urdf format, including parameters for ROS 2 controllers, CAD files, and Gazebo Simulation parameters. Secondly, the *MoveIt!2 package* provides motion control details, including robot controller parameters, and the Motion Planning interface. Finally, the *Robot Bringup package* links ROS 2 with the real robot arm, enabling complete motion control and communication with the robot via ROS 2 nodes through the ROS 2 driver of the robot. Combining Gazebo and MoveIt!2 simulates robot arm operation, while MoveIt!2 and Robot Bringup both together control the real robot. This method offers a versatile framework for prototyping, testing, and deploying robotic systems in ROS 2 environments.

Following the establishment of our framework, we have integrated two distinct features aimed at enhancing, streamlining, and standardizing robot manipulator operation within ROS 2. First, leveraging its modular architecture, we have encapsulated robot arm motion control commands into

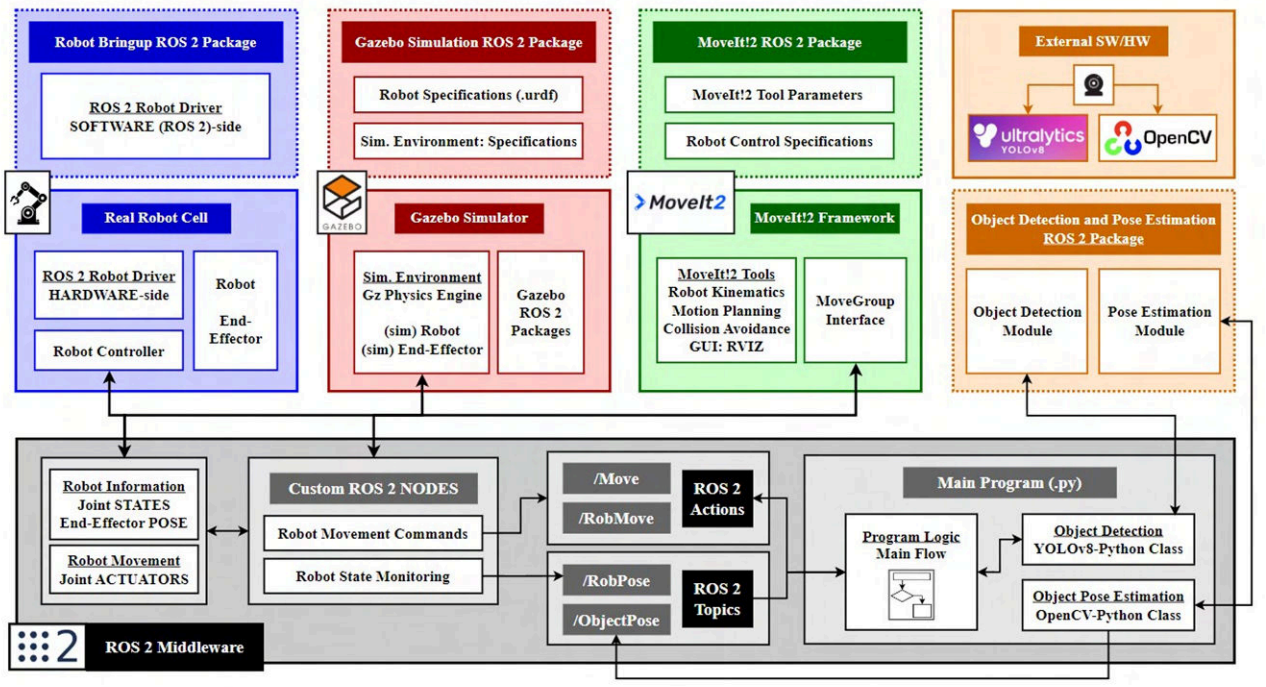

Fig. 3. Software Architecture of the ROS 2-based Framework. Any Robot Manipulator which supports ROS 2 connectivity can be simulated and controlled by the system. The framework is formed by 3 main components: The Robot Simulation and Control packages (Robot Bringup, Gazebo and MoveIt!2), the ROS 2 Middleware (communication platform), and the external HW/SW components (web camera, YOLOv8 and OpenCV).

discrete modules, accessible via ROS 2 Actions. The /Move action offers a diverse array of robot movements, accepting inputs such as joint values (MoveJ), individual joint values (MoveR), Cartesian vectors (MoveL), or rotations (MoveROT, MoveRP). Conversely, the /RobMove action empowers users to guide the robot to specific end-effector poses, providing various movement types (point-to-point or linear), joint speed adjustments, and precise pose control (position and orientation). Second, we have developed a dedicated ROS 2 Node tasked with retrieving the 6-degree-of-freedom (6DOF) pose information of the robot from MoveIt!2, subsequently publishing it to a designated ROS 2 Topic (/RobPose).

The ROS 2 Actions are encapsulated within Python classes, facilitating direct communication with MoveIt!2's Motion Control tool through its Python API, allowing for straightforward control of the robot arm via simple ROS 2 Action Client calls. Meanwhile, the /RobPose ROS 2 Topic enables any system component to access the robot's information during execution by simply subscribing to the ROS 2 Topic. With this approach, we ensure that any component of the system can seamlessly talk to the robot by using ROS 2's communication mechanisms.

Incorporating YOLOv8 and OpenCV into our ROS 2 Framework has been achieved through the development of a custom ROS 2 (python-based) package, integrating both software through their Python APIs. This integration allows for the seamless execution of YOLOv8 and OpenCV source code within the ROS 2 middleware. Specifically tailored for our application, where real-time access to the 2D pose of objects within the robot workspace is crucial, ROS 2 Topics serve as the communication mechanism for sharing this information. Upon initialization of the ROS 2 environment for the Robot Cell, a continuous loop is executed wherein YOLOv8 detects and classifies objects within the workspace, OpenCV estimates their 2D pose, and a ROS 2 Node publishes the real-time 2D pose information to a dedicated ROS 2 Topic (/ObjectPose). This enables any other component of the system to access the object pose information during execution, facilitating seamless coordination and interaction within the robotic environment.

IV. Use-Case: Conveyor Belt Kitting Station

The technologies introduced in the methodology section ensure a scalable and reproducible robotic pick-and-place application. YOLOv8 and OpenCV, custom-trained for specific objects, provide accurate detection and pose estimation, making the system adaptable to diverse objects and applications. The ROS 2 framework supports simulation and control of any robot arm, enabling broad applicability across different manipulators. Its straightforward integration of external software and hardware allows for easy replacement or addition of modules, ensuring system flexibility and robustness.

The practical implementation of our framework is exemplified through an automated robotic kitting application, which serves not only as a demonstration but as a clear indication of our solution's ability to:

- Enable the robot to interface seamlessly with different software and hardware modules in ROS 2 to efficiently perform object pick-and-place tasks.
- Control and operate any ROS 2-supported robot arm equally in both simulation and real world.
- Facilitate the integration of external hardware and software components into the system by encapsulating their functionality within ROS 2 communication mechanisms, enabling seamless interaction with the robot and other framework elements.

A. Automated Kitting of Bolts, Nuts and Washers

The study evaluates the performance of object detection, classification, and tracking algorithms integrated with ROS 2-based robot control in an automated kitting scenario (depicted in Fig. 1). The kitting process comprises several key steps: firstly, an object is introduced onto a continuously moving conveyor belt (Fig. 1C); subsequently, it undergoes detection and classification via a basic web camera. To facilitate robot manipulation, object coordinates are transformed from camera to robot coordinates. The robot's pose is precisely estimated for accurate object retrieval, followed by placing the object in its designated slot (Fig. 1B). This comprehensive sequence assesses the system's capabilities in real-world object manipulation scenarios.

In order to render the object detection and classification task sufficiently challenging and to introduce a level of complexity suitable for the use-case, a selection of bolts, nuts, and washers, each distinguished by two different colors (yellow and silver), has been employed. This approach yields six distinct object classes that require detection and classification. The choice of common industrial spare parts serves as an exemplar, showcasing the potential implementation of the proposed system within an industrial context. The experimental cell setup, as depicted in Fig. 1A, encompasses a USB webcam, a conveyor belt, a robotic arm, and six designated slots for the placement of workpieces.

B. Object Detection: Data Collection and Model Training

The main goal of the object detection module is to precisely detect and locate workpieces on a conveyor belt using YOLOv8, harnessing a USB camera to capture the live video feed of the conveyor area. To train YOLOv8 for this task, we compiled a custom dataset comprising 1401 images, labelled using RoboFlow as shown in Fig. 4A, encompassing six distinct workpiece classes. This dataset underwent rigorous partitioning, with 80% allocated for training and the remainder for validation and testing, ensuring robust model training and generalization.

To boost the model's adaptability, we employed different data augmentation techniques on the training dataset. These techniques, including random rotations, brightness adjustments, Gaussian blurring, and realistic noise incorporation, aimed to simulate real-world scenarios and disturbances. This augmentation strategy substantially enriched the dataset, broadening its exposure to diverse scenarios and fortifying the model's detection accuracy and robustness. Following augmentation, the dataset was expanded to 3400 images, providing YOLOv8 with enhanced capabilities to detect and localize objects amidst real-world disturbances. Among YOLOv8's pre-trained detection models, YOLOv8m (medium) emerged as the optimal choice for our real-time application, effectively balancing performance and accuracy requirements.

Upon execution, the trained model accurately identifies workpieces in the camera feed, presenting real-time bounding boxes and object class predictions, as shown in Fig. 4B.

C. Object Pose Estimation & Tracking

In our Automated Kitting Cell, the Object Pose Estimation method capitalizes on the output provided by YOLOv8 for each camera frame. YOLOv8 generates a vector that contains crucial information about the detected objects, including their class labels and the coordinates of the bounding boxes encompassing each object. Leveraging this output, our method computes the 2D Pose of every identified object within the workspace (conveyor belt, in this case). This computation involves the application of the formula (1) to compute the pose ArUco marker first, and formulas (2) and (3) to compute the 2D Pose of each object from the bounding box (pixel) coordinates provided by YOLOv8.

By systematically processing each component of the vector generated by YOLOv8, we are able to derive the precise 2D Pose of every detected object for every camera frame captured by the system. This real-time estimation of object positions serves as a foundational element for facilitating accurate robotic manipulation tasks and optimizing the efficiency of the kitting process within the Automated Kitting Cell.

The 2D Poses obtained through our method are initially relative to the ArUco marker positioned within the robot workspace. To translate these coordinates into the requisite 6 Degrees of Freedom (6DoF) Pose of the robot necessary for object manipulation, a transformation from ArUco marker coordinates to robot coordinates is imperative. As depicted in Fig. 5, given the assumption of a fixed and known height of the conveyor belt relative to the robot, the transformation primarily involves adjusting the x and y coordinates of the objects. This adjustment is achieved by applying the offset of the ArUco tag (relative to the robot base) to each object's pose. Once the object poses are derived, they are shared across the ROS 2 Network via the /ObjectPose ROS 2 Topic. This topic serves as a conduit for transmitting a vector comprising the precise poses of all objects within the workspace during execution. This live streaming of object pose information enables the robot (and any other component within the system) to access pertinent data regarding the objects within its operational environment, facilitating timely and accurate object manipulation tasks.

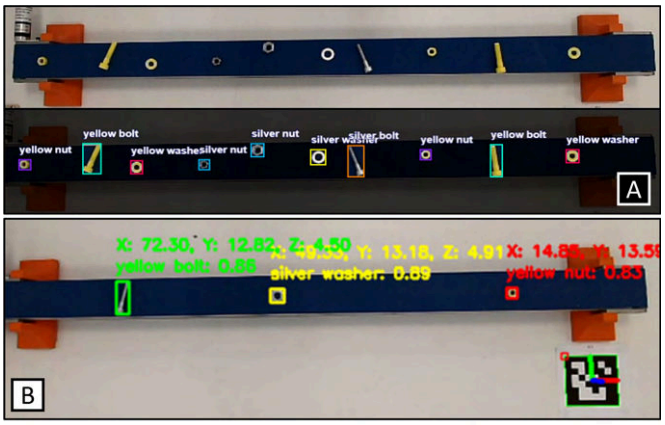

Fig. 4. Representation of how the yellow and white bolts, nuts, and washers are labelled in RoboFlow (A) for the posterior object detection and classification model training in YOLOv8, which predicts the object type and position in pixel coordinates for every single camera frame (B).

D. Integration & Deployment using ROS 2

The ROS 2 Middleware serves as the backbone facilitating seamless communication among diverse system components, streamlining the integration of novel software and hardware elements. Upon integration, the functionality of these components is encapsulated within ROS 2 Nodes, enabling interaction with the entire platform through ROS 2 communication methods. For instance, in the Automated Kitting use-case, a Python script orchestrates various tasks. It subscribes to the /ObjectPose ROS 2 Topic to acquire real-time object positions, computes the robot's 6 Degrees of Freedom (6DoF) necessary for object manipulation, and triggers robot movements by invoking the /Move and /RobMove ROS 2 Actions. These operations are effortlessly managed within the same Python script, leveraging simple class instances for enhanced modularity and efficiency.

To ensure the robustness and efficacy of the Automated Kitting Cell's operation and control, as well as to validate the automated kitting capabilities of the UR10e robot, comprehensive validation was conducted within the Gazebo simulation environment. The simulation scenario involved a straightforward cube pick-and-place task from the conveyor belt, meticulously designed to assess the robot's waypoints,

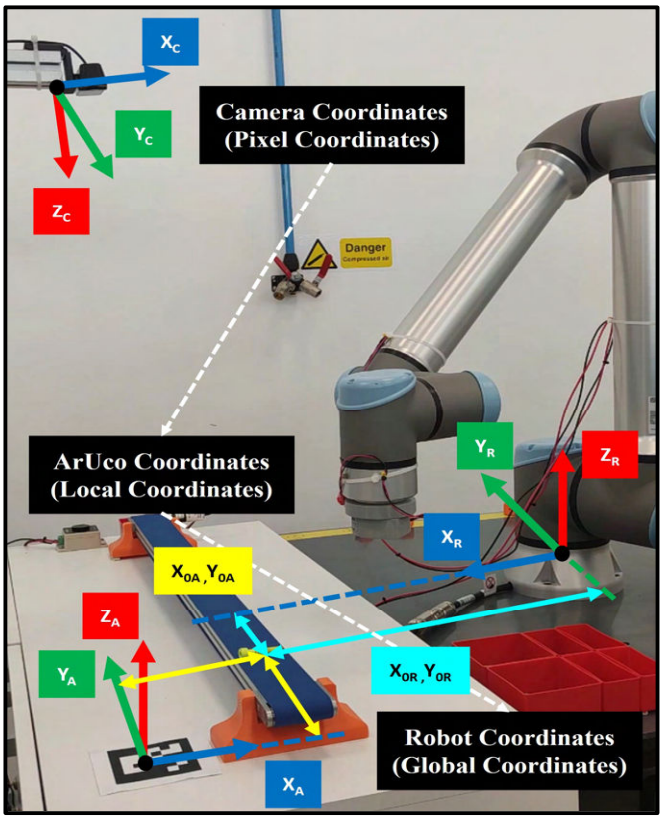

Fig. 5. UR10e, Conveyor Belt and Camera setup for the automated kitting of the colored industrial spares. The Camera (c), ArUco (A), and Robot (R) coordinate frames are represented, as well as the object's position in both ArUco and Robot coordinates.

motion control, and collision avoidance mechanisms. Notably, the simulation excluded the integration of YOLOv8 and OpenCV, as their performance validation necessitated real-world testing to accurately evaluate object detection and pose estimation under varied environmental conditions. To simulate these functionalities, a custom plugin was developed in Gazebo, emulating the behavior of the object detection and pose estimation module by publishing object poses to the /ObjectPose ROS 2 topic during execution. Subsequently, the Kitting Application's program was devised within the simulation environment, executing the cube pick-and-place operation seamlessly. Leveraging this simulation-centric approach, the identical program developed in Gazebo seamlessly transitioned to the real cell environment, where it accessed real-time object pose information through subscription to the /ObjectPose topic. This methodology ensures a streamlined transition from simulation to real-world deployment, paving the way for future exploration of simulation-based object detection and pose estimation models' deployment in real-world settings, thereby enriching the sim-to-real capabilities of the system.

The successful execution of industrial spares pick-and-place tasks within our system was made possible through the implementation of a custom electro-magnetic gripper, purpose-built for this specific application. Effortlessly integrated into the ROS 2 framework, the electro-magnetic gripper's functionality, including activation and deactivation, has been encapsulated within a ROS 2 Service, facilitating seamless interaction with other system components. Controlled via the Robot Controller's I/O, the gripper's integration underscores the platform's flexibility and adaptability to accommodate custom hardware components. This deliberate integration serves as a testament to the system's capability to incorporate bespoke hardware effortlessly, underscoring its versatility for diverse industrial applications.

V. Results and Evaluation

The evaluation of our system's performance involved a comprehensive series of 90 test runs, each consisting of 15 trials per workpiece class. These tests scrutinized the system's ability to execute critical tasks such as object detection, classification, tracking, and sorting as objects traversed the conveyor belt. During each test iteration, a bolt, nut, or washer was positioned on the conveyor belt. The object detection, classification, and pose estimation module continuously operated, promptly transmitting the pose of the detected object to ROS 2. Simultaneously, the robot module subscribed to this topic, initiating the robotic arm's movement upon detecting an object, immediately executing the pick-and-place operation to position it into its designated slot.

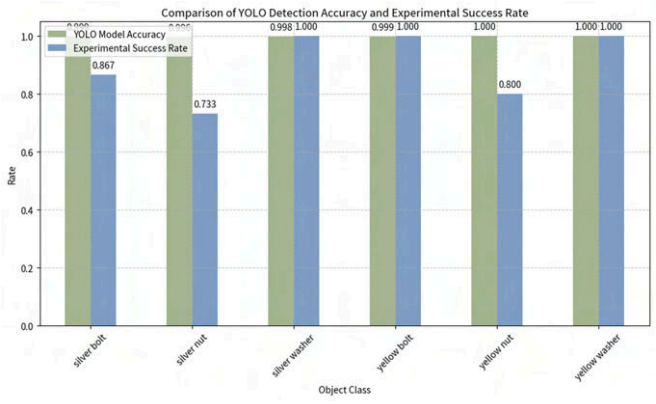

Fig. 6. Results of the Conveyor Belt Automated Kitting Use-Case testing, highlighting the comparison between YOLOv8 model detection accuracy against the overall experimental success rate.

The system exhibited an overall success rate of 90%, signifying its competence in correctly detecting, grasping, and placing objects in the majority of cases. The few instances of failure were exclusively attributed to object misplacement, originating from erroneous initial classifications made by the YOLOv8 model during the detection phase. Additionally, the system consistently demonstrated precision by achieving a perfect 100% accuracy rate in providing object coordinates across all experiments. This unswerving accuracy underscores the robustness and reliability of our coordinate conversion algorithm, coupled with the integration of ArUco markers, in practical deployment scenarios. Fig. 6 portrays a comparative analysis of YOLOv8's object detection accuracy among various object classes, juxtaposed with their corresponding experimental success rates. The results unequivocally illustrate the model's exceptional detection accuracy, with precision values predominantly approaching or reaching 1.0, further affirming its reliability in real-world applications.

The system's robustness and high fidelity are underscored by its stable communication among diverse modules, including YOLOv8, OpenCV, the Robot, End-Effector, and MoveIt!2, seamlessly integrated into ROS 2. Throughout extensive testing, the reliability of communication channels, facilitated by ROS 2 Topics, Services, and Actions, remained consistently robust, in both simulation and real world, ensuring smooth interaction and data exchange among system components. This reliability ensures the system's suitability for real-world deployment scenarios, bolstering confidence in its performance and efficacy within complex operational environments.

VI. Conclusion

This research introduced an open-source ROS 2-based framework designed to enhance object handling within industrial robot cells, emphasizing modularity, interoperability, and real-time object detection and classification using YOLOv8 and OpenCV. The framework's efficacy was evidenced through an automated kitting use-case. Its successful implementation underscored the robustness and versatility of the proposed framework, facilitating the seamless integration of diverse components, and demonstrating that our approach enables the rapid and intuitive deployment of any robotic application. This is particularly advantageous for small and medium automation enterprises that require frequent reprogramming and adaptation of robot cells to different settings, enhancing operational flexibility and efficiency.

The proposed methodology is scalable and reproducible for any robotic object handling application. YOLOv8 and OpenCV can be custom-trained for specific objects, ensuring accurate detection and pose estimation. The ROS 2 framework supports the simulation and control of any robot arm, allowing broad applicability across various manipulators. Additionally, the framework's simple integration of external software and

hardware ensures that it can be easily adapted or expanded, making it suitable for diverse industrial scenarios.

Future work could involve testing the complete object detection process within the simulation environment first, allowing for evaluation of the model's performance in a virtual setup before deployment in the real robot cell. This approach would facilitate a seamless transition from simulation to the real world, enhancing the framework's utility even before commissioning the actual cell. Improving the current simplification of the robot's motion—by implementing a tracking algorithm—would enable more precise object following, optimizing manipulator performance. Exploring AI-based robot control algorithms could further augment the system's intelligence and flexibility. Aligning the features and characteristics of the ROS 2-based framework with Industry 4.0 and modern manufacturing principles could pave the way for future standardization, ensuring its relevance and applicability in advanced industrial contexts.

ACKNOWLEDGMENT

We express our gratitude to Jamie Rice and Daniel Oakley, Lab Technicians, and James Fowler, Senior Technical Officer at the Intelligent Automation Lab, for their support and technical expertise, which were essential for the project's success.

REFERENCES

[1] A. W. Colombo, S. Karnouskos, and T. Bangemann, "Towards the Next Generation of Industrial Cyber-Physical Systems," in *Industrial Cloud-Based Cyber-Physical Systems*, Cham: Springer International Publishing, 2014, pp. 1–22. doi: 10.1007/978-3-319-05624-1_1.

[2] R. J. Alattas, S. Patel, and T. M. Sobh, "Evolutionary Modular Robotics: Survey and Analysis," *J Intell Robot Syst*, vol. 95, no. 3–4, pp. 815–828, Sep. 2019, doi: 10.1007/s10846-018-0902-9.

[3] Y. Zou, D. Kim, P. Norman, J. Espinosa, J.-C. Wang, and G. S. Virk, "Towards robot modularity — A review of international modularity standardization for service robots," *Rob Auton Syst*, vol. 148, p. 103943, Feb. 2022, doi: 10.1016/j.robot.2021.103943.

[4] M. Wojtynek, J. J. Steil, and S. Wrede, "Plug, Plan and Produce as Enabler for Easy Workcell Setup and Collaborative Robot Programming in Smart Factories," *KI - Künstliche Intelligenz*, vol. 33, no. 2, pp. 151–161, Jun. 2019, doi: 10.1007/s13218-019-00595-0.

[5] H. Paul, Z. Qiu, Z. Wang, S. Hirai, and S. Kawamura, "A ROS 2 Based Robotic System to Pick-and-Place Granular Food Materials," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, Dec. 2022, pp. 99–104. doi: 10.1109/ROBIO55434.2022.10011782.

[6] M. Blatnický, J. Dižo, J. Gerlici, M. Sága, T. Lack, and E. Kuba, "Design of a robotic manipulator for handling products of automotive industry," *Int J Adv Robot Syst*, vol. 17, no. 1, p. 172988142090629, Jan. 2020, doi: 10.1177/1729881420906290.

[7] N. Elangovan, L. Gerez, G. Gao, and M. Liarokapis, "Improving Robotic Manipulation Without Sacrificing Grasping Efficiency: A Multi-Modal, Adaptive Gripper With Reconfigurable Finger Bases," *IEEE Access*, vol. 9, pp. 83298–83308, 2021, doi: 10.1109/ACCESS.2021.3086802.

[8] S. D'Avella, C. A. Avizzano, and P. Tripicchio, "ROS-Industrial based robotic cell for Industry 4.0: Eye-in-hand stereo camera and visual servoing for flexible, fast, and accurate picking and hooking in the production line," *Robot Comput Integr Manuf*, vol. 80, p. 102453, Apr. 2023, doi: 10.1016/j.rcim.2022.102453.

[9] A. Bonci, A. Di Biase, M. Cristina Giannini, F. Gaudeni, S. Longhi, and M. Prist, "ROS 2 for enhancing perception and recognition in collaborative robots performing flexible tasks," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, Sep. 2023, pp. 1–4. doi: 10.1109/ETFA54631.2023.10275338.

[10] E. Trunzer *et al.*, "System architectures for Industrie 4.0 applications: Derivation of a generic architecture proposal," *Production Engineering*, vol. 13, no. 3–4, pp. 247–257, Jun. 2019, doi: 10.1007/S11740-019-00902-6/FIGURES/4.

[11] R. Arrais, P. Ribeiro, H. Domingos, and G. Veiga, "ROBIN: An open-source middleware for plug'n'produce of Cyber-Physical Systems," *Int J Adv Robot Syst*, vol. 17, no. 3, p. 172988142091031, May 2020, doi: 10.1177/1729881420910316.

[12] M. Zimmer, P. Ferreira, P. Danny, A. Al-Yacoub, N. Lohse, and V. Gentile, "Towards a Decision-support Framework for Reducing Ramp-up Effort in Plug-and-Produce Systems," in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, IEEE, May 2019, pp. 478–483. doi: 10.1109/ICPHYS.2019.8780369.

[13] Open Source Computer Vision (docs.opencv.org), "OpenCV: Detection of ArUco Markers." [Online]. Available: https://docs.opencv.org/3.4/d5/dae/tutorial_aruco_detection.html

[14] S. Lee and Y. Lee, "Real-Time Industrial Bin-Picking with a Hybrid Deep Learning-Engineering Approach," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2020, pp. 584–588. doi: 10.1109/BigComp48618.2020.00015.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[16] H. Lou *et al.*, "DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor," *Electronics (Basel)*, vol. 12, no. 10, p. 2323, May 2023, doi: 10.3390/electronics12102323.

[17] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," *ICRA workshop on open source software*, vol. 3, no. 3.2. 2009.

[18] A. Bonci, F. Gaudeni, M. C. Giannini, and S. Longhi, "Robot Operating System 2 (ROS2)-Based Frameworks for Increasing Robot Autonomy: A Survey," *Applied Sciences*, vol. 13, no. 23, p. 12796, Nov. 2023, doi: 10.3390/app132312796.

[19] M. Simonic *et al.*, "Modular ROS-based software architecture for reconfigurable, Industry 4.0 compatible robotic workcells," in *2021 20th International Conference on Advanced Robotics (ICAR)*, IEEE, Dec. 2021, pp. 44–51. doi: 10.1109/ICAR53236.2021.9659378.

[20] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Sci Robot*, vol. 7, no. 66, May 2022, doi: 10.1126/scirobotics.abm6074.

https://dspace.org/

Towards robot software abstraction: ROS 2-based framework for object handling within a robot cell

Viso, Mikel Bueno; Huang, Jingjing; Asif, Seemal; Khan, Fahad; Webb, Phil

2024-08-18
Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/indin58382.2024.10774307
Viso MB, Huang J, Asif S, et al., (2024) Towards robot software abstraction: ROS 2-based framework for object handling within a robot cell. In: 2024 IEEE 22nd International Conference on Industrial Informatics (INDIN), 18-20 August 2024, Beijing, China

Downloaded from a DSpace repository. https://dspace.org/