In: 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE).
28 August 2024 - 1 September 2024, Bari, Italy, pp. 2610-2617
DOI:10.1109/case59546.2024.10711476

Unlocking the potential of Robot Manipulators: Seamless Integration Framework

Mikel Bueno[1], Irene Bernardino[1], Seemal Asif[1], and Phil Webb[1]

***Abstract*— This study introduces a groundbreaking framework designed to enhance the adaptability and efficiency of robot manipulators in manufacturing, leveraging ROS 2 and a modular middleware to transcend traditional robotic constraints. The framework's efficacy is exemplified through a pick-and-place task, serving not merely as a demonstration but as robust evidence of the framework's ability to enable complex object manipulation tasks far beyond repetitive activities. By integrating advanced perception capabilities with a YOLOv8-based object detection model and an OpenCV-based pose estimation module, the framework showcases a seamless interaction between sophisticated software tools and robotic hardware. This integration not only simplifies the incorporation of intelligent components into robotic systems but also significantly broadens their applicability and enhances efficiency across diverse tasks and applications. The use-case, therefore, stands as compelling evidence of the framework's potential to revolutionize industrial and collaborative robotics, providing a unified and adaptable platform for the development, testing, and deployment of robotic solutions in modern manufacturing settings. The research aims to simplify the integration of intelligent components into robotic systems, thereby extending their utility and efficiency across a broader range of tasks and applications, ultimately advancing the capabilities of industrial and collaborative robotics for modern manufacturing needs.**

I. Introduction

With industries facing the need for flexible production lines capable of handling diverse tasks and smaller batch sizes, the limitations of current robot manipulators, rooted in their rigid programming and task-specific configurations, become increasingly apparent. This discord between the existing capabilities of robotic systems and the agile, adaptable manufacturing processes required today highlights a critical gap in industrial and collaborative robotics. Addressing this gap, our research proposes a transformative approach that leverages a novel software framework designed to enhance the flexibility and utility of robot manipulators. By integrating advanced software tools and modular design principles, our solution aims to unlock the untapped potential of these robots, enabling them to meet the complex demands of next-generation manufacturing environments. This paper sets out to explore the development and application of this framework, illustrating its impact through practical implementation and offering a glimpse into the future of adaptable, efficient robotic automation in industry.

Currently, cell-level robot programming tasks (performed by an operator) are limited to robot motion and program logic definition, typically through a GUI (Graphical User Interface) or the robot's teach pendant. Integrating intelligence or decision-making capabilities into a robot controller requires complex systems and reliance on Programmable Logic Controllers (PLCs), adding considerable complexity and cost to the setup. The aim of this research is to streamline this process by leveraging ROS 2 and its capability to seamlessly integrate external components. By operating the robot within our framework, we aim to simplify the integration of intelligent components and enhance the overall functionality of robotic systems within various applications.

A significant contribution of our work is evidenced through a practical implementation of an object manipulation task utilizing an industrial ABB robot and different cubes. Initially validated in simulation, our framework has been seamlessly deployed onto an ABB IRB-120 cell enclosure (Fig. 1). Through this use-case, we demonstrate the integration of external software components, including a YOLOv8-based object detection and classification model, alongside an OpenCV-based object Pose-Estimation module, enhancing the robot's perception capabilities within the operational environment.

Acknowledging the demand for more flexible robotic systems, we focus on developing solutions that enhance adaptability and versatility, effectively addressing this gap. The main contributions are listed as follows:

- We present a user-friendly software framework enabling the simulation and control of robot manipulators compatible with ROS 2, thereby enhancing their adaptability to diverse tasks.
- Our framework facilitates the integration of various hardware and software components into robotic systems, further enhancing their functionality and versatility. We propose a method to seamlessly integrate external modules and their functionality to the ROS 2-based robotic system.
- We propose a solution for object manipulation within robot cells, utilizing simple yet effective approaches for object detection and classification, along with precise robot pose estimation. This approach enables efficient pick-and-place operations even in dynamic environments. Importantly, our approach can be

* 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/irb120_PoseEstimation

[1]All authors are with the Centre for Robotics and Assembly, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, MK43 0AL United Kingdom. E-mails: Mikel.Bueno-Viso@cranfield.ac.uk (corresponding author), I.BernardinoSanchez.854@cranfield.ac.uk, s.asif@cranfield.ac.uk, and p.f.webb@cranfield.ac.uk .

© 2024 IEEE. Issued with: Creative Commons Attribution License (CC:BY 4.0).
The final published version (version of record) is available online at DOI:10.1109/case59546.2024.10711476.
Please refer to any applicable publisher terms of use.

effortlessly deployed across different robot arms supported by ROS 2, regardless of the underlying hardware, with a streamlined and consistent deployment process.

By addressing the limitations of traditional robot manipulators and providing a reproducible and extendable practical solution (exemplified through the ABB Robot's use-case), our research significantly advances the capabilities of industrial and collaborative robotics for modern manufacturing needs.

II. Related Work

A. Reconfigurable Manufacturing

In recent years, the landscape of industrial automation has undergone a profound shift towards reconfigurable systems, driven by advancements in CPS and the imperative for flexibility and efficiency in manufacturing [1]. At the heart of this transformation lies the principle of modularity, which advocates for the decomposition of complex systems into smaller, interchangeable modules that can be easily reconfigured to adapt to changing production needs [2]. This modular approach enables what is commonly referred to as plug-and-produce, where components seamlessly integrate into existing systems, reducing setup time, and facilitating rapid reconfiguration.

Several works in the literature emphasize the importance of software architecture [3] and middleware frameworks [4] in enabling reconfigurability and flexibility within manufacturing systems, aligning closely with the principles of CPS and PnP paradigms. SOA (Service-Oriented Architecture) facilitates the integration of hardware devices and software applications within reconfigurable manufacturing systems, enabling seamless connectivity and plug-and-produce capabilities [5], [6]. These research works highlight the significance of middleware frameworks, standardized interfaces, and a modular approach in enabling reconfigurability and enhancing connectivity within manufacturing systems.

B. ROS 2 as Software Middleware for Robot Systems

ROS 2 is an open-source framework providing a flexible and modular platform for developing robotic systems [7]. A framework of this type is especially suitable for robotic applications that require the interaction between different hardware and software components, where each is developed by different robot manufacturers offering specific interfaces [8]. ROS 2 offers tools and features to implement modular architectures, making it possible for companies to adapt robot manipulators for non-standard tasks without starting from scratch [9]. Robots are limited to their own interfaces, communication protocols and programming languages, and ROS 2 can help overcome those limitations. It can simultaneously enable perception, recognition, and re-planning capabilities [8], provide a real-time motion control system which communicates with external elements such as OpenPLC [10], or support both fast development and deployment of new software and hardware modules as well as the fast setup and reconfiguration of the system [11].

ROS 2 fosters faster development, collaborations, and the advancement of new technologies, with this trend expected to persist as it matures further [12]. The inherent flexibility, scalability, and modularity of its architecture make it an ideal middleware solution for reconfigurable systems, particularly in contexts where robot manipulators are tasked with executing diverse operations, requiring to be reprogrammed frequently.

C. Object Detection and Robot Pose Estimation

Object detection, including geometric shape and orientation classification, is crucial for understanding object location. Calibration is essential for post-processing images, varying based on dimensionality, such as 3D point clouds or 2D camera images. Visual markers, as seen in [13], are commonly employed for calibration, allowing precise distance calibration in 3D point clouds. Once calibrated, the robot can accurately estimate the object's position for effective picking [14], [15].

YOLO (You Only Look Once) is a widely known object detection algorithm. It divides images into grids, predicting bounding boxes and class probabilities simultaneously. In a robot cell, YOLO can analyze camera feeds to autonomously detect and classify objects. Its latest version, YOLOv8, has been widely used for many robot-object manipulation tasks, and advanced solutions have been proposed to detect smaller-sized objects [16] or to predict the orientation along with the position of the objects (YOLO-OBB labelling method) [17], [18].

D. Conclusions

Recent literature has increasingly focused on the development of adaptable frameworks and technologies to meet the dynamic demands of modern production environments. Various approaches have been proposed to enhance the flexibility and agility of manufacturing systems, ranging from modular hardware designs to flexible software architectures.

Our research aligns with the aforementioned trend, yet distinguishes itself by concentrating specifically on enhancing the adaptability and intelligence of robot manipulators. While existing literature showcases robots' capability to undertake complex and specific tasks, such as bin picking, autonomous decision making, or automated program generation, these pipelines typically necessitate complex, expensive systems and extensive deployment efforts. In response, our framework

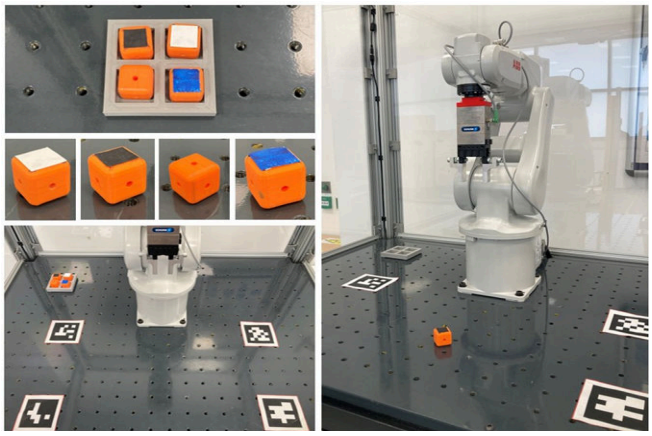


Fig. 1. ABB IRB-120 Enclosure (IA Lab, Cranfield University) setup for the Cube Detection and Pick-and-Place Use-Case.

offers a streamlined approach, requiring minimal deployment effort, and facilitates seamless integration of any custom hardware and software component. This emphasis on simplicity and ease of integration marks a significant departure from traditional approaches, promising heightened accessibility and applicability in real-world production environments, especially for SMEs that might need to re-program and re-design robot cells quickly and effectively in short periods of time.

III. Methodology

A. *ROS 2: Robot Simulation and Control Framework*

Our ROS 2-based system enables the simulation and control of any robot arm supported by ROS 2. Fig. 2 illustrates the software architecture, which has been fully developed and integrated within ROS 2 in an Ubuntu 22.04 PC. We have adapted and improved the default tools provided by ROS 2 (Gazebo, MoveIt!2 and ROS 2 control algorithms) to establish a standardized robot simulation and control pipeline that is agnostic to specific robot models. This development has been realized through three different custom ROS 2 packages.

Firstly, the *Gazebo package*, which contains all the information related to the robot, represented in .urdf (Unified Robot Description Format) format, in addition to the parameters required to tune the ROS 2 controllers and the CAD files and information about the Gazebo environment. Secondly, the *MoveIt!2 package*, that provides the MoveIt!2 tool with all the information for the motion control of the robot. Information such as robot specifications (obtained from the Gazebo package), controller parameters and the Motion Planning interface are provided in this package. Finally, the *Robot Bringup package*, which acts as the main link between the ROS 2 system and the real robot arm. As mentioned before, the robot arm control needs to be supported by ROS 2, in the form of a ROS 2 driver. The driver, running in both the robot controller and ROS 2, enables the complete robot motion and state control through different ROS 2 nodes. The Gazebo and MoveIt!2 packages are combined in order to simulate the operation the robot arm on a Gazebo/ROS 2 environment, and the MoveIt!2 and Robot Bringup packages are combined to control and operate the real robot arm. The developed method provides the user with a versatile and comprehensive framework for prototyping, testing, and deploying robotic systems within ROS 2 environments.

Utilizing ROS 2's modular architecture and following the PnP and SOA principles, we have encapsulated robot arm movements and tasks into distinct modules, invoking them through ROS 2 Actions (see bottom-left, Fig. 2). Actions were deliberately chosen for their call-response architecture, continuous feedback, and built-in cancellation capabilities. The /Move action offers a wide range of robot movements, accepting inputs such as joint values (MoveJ), individual joint values (MoveR), Cartesian vectors (MoveL), or rotations (MoveROT, MoveRP). In contrast, the /RobMove action empowers users to guide the robot to specific end-effector poses, providing movement types (point-to-point or linear), joint speed, and precise pose control (position and orientation). In addition to motion actions, the /Sequence ROS 2 action facilitates the execution of predefined motion sequences. These ROS 2 actions are encapsulated within Python classes, which talk to MoveIt!2's Motion Control tool through its Python API and enable the direct control of the robot arm through simple ROS 2 Action Client calls. This approach simplifies the integration of any external software module, which can talk to ROS 2 (and therefore, communicate with the robot) directly through Python (this can be implemented through C++ as well).

In addition to the ROS 2 actions allowing the direct control of the robot's movement, we have developed supplementary tools to enhance simulation and control capabilities within our

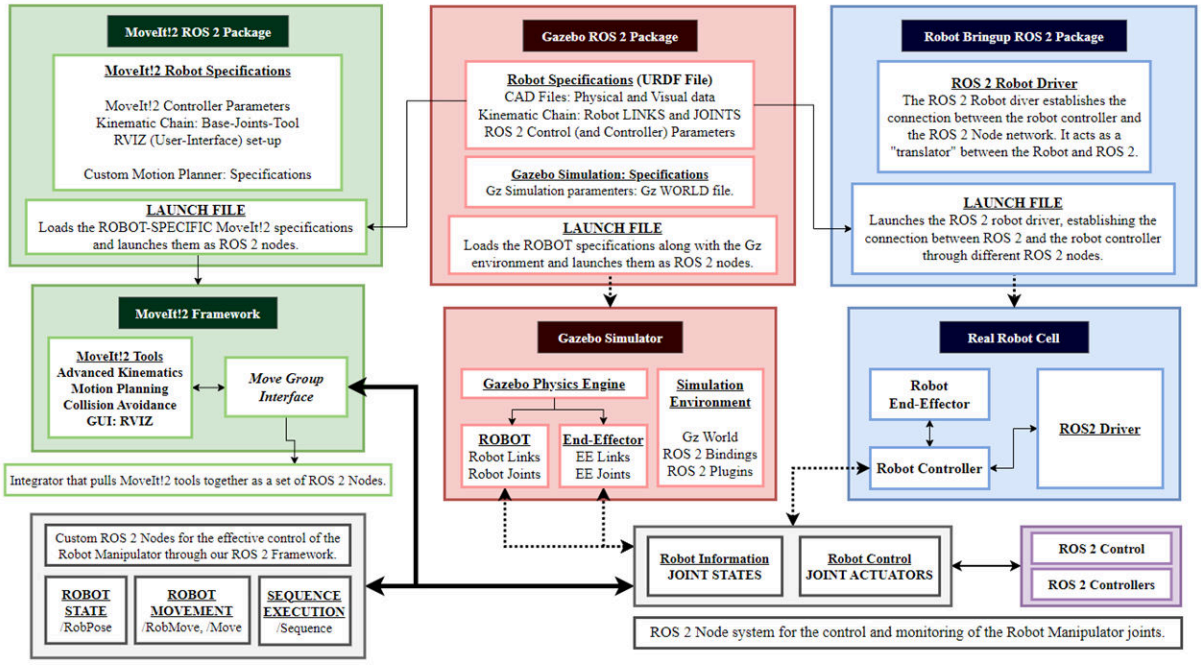


Fig. 2. Software Architecture of our ROS 2-based Robot Simulation and Control Framework. Any Robot Manipulator which supports ROS 2 connectivity can be simulated and controlled by our system. The framework is formed by 3 main components: The Gazebo, MoveIt!2 and Robot Bringup ROS 2 packages.

system. Firstly, we have developed a dedicated ROS 2 Node to retrieve the 6-degree-of-freedom (6DOF) pose information of the robot from MoveIt!2, subsequently publishing it to a designated ROS 2 Topic (/RobPose). Additionally, we have created custom ROS 2-Gazebo plugins that publish the precise pose of any robot link or object spawned to the Gazebo environment to a specific ROS 2 Topic. These additional features culminate our whole robot simulation and control setup, significantly improving the system's usability and intuitiveness, and enhancing user experience.

Our system is robot agnostic, and allows the user to interact with the simulated robot and the real robot exactly in the same way through MoveIt!2. This uniformity facilitates a smooth transition from simulation to real-world deployment, allowing users to test robot applications even before the physical cell deployment has been done. To facilitate wider adoption and collaboration, our custom ROS 2 packages have been made publicly available on the *ROS 2: Sim-to-Real Robot Control*[2] GitHub repository. This repository includes comprehensive documentation and detailed instructions, enabling researchers and robotics engineers to adapt and extend our proposed framework to any custom robotics application using ROS 2 effectively.

B. Integration of External Software and Hardware Tools

ROS 2 offers a high degree of customization, facilitating the integration of external components more efficiently than traditional robotic or automation systems. The absence of requirements for PLCs or complex communication protocols or interfaces further simplifies this process. This flexibility permits the incorporation of external software and hardware packages, providing a robust communication platform that interfaces seamlessly with the robot simulation and control framework introduced in the previous section. Fig. 3 illustrates this integration process and demonstrates how external software and hardware elements interact with ROS 2, showcasing its high degree of customization compared to traditional robotic or automation systems.

ROS 2 packages and scripts are developed in either Python or C++. Therefore, any external software or program can be directly accessed through its Python or C++ API, and included directly in the ROS 2 application logic or source code. Apart from that, different SW functions and methods can be encapsulated into ROS 2 Topics, Services and Actions, which can be accessed by any ROS 2 node in the system. These communication mechanisms, which are facilitated and fully managed by ROS 2, enable efficient multitasking and parallel execution of various features and tools within the system.

Integrating Hardware components into ROS 2 requires an initial translation step, where the HW element and its functionality are accessed and made available through a wrapper, known as the ROS 2 driver. This integration process involves mapping the hardware's capabilities to ROS 2 interfaces, such as Nodes, Topics, Services, or Actions, to enable seamless communication with the ROS 2 ecosystem. However, this process may present challenges, such as ensuring compatibility between the hardware and ROS 2, handling real-time constraints, and managing data synchronization. Solutions to these challenges often involve developing custom drivers tailored to the specific hardware, implementing robust error-handling mechanisms, and optimizing data transmission protocols. Additionally, thorough testing and validation are essential to ensure the reliability and stability of the integrated hardware within the ROS 2 framework. Through diligent attention to these aspects of the integration process, developers can effectively leverage the full potential of ROS 2 to interface with a diverse range of hardware components for robotic applications.

C. Object Detection and Classification

Our object detection approach is centered on training a customized YOLOv8 model using bespoke datasets, which are previously labeled using the RoboFlow annotation tool. RoboFlow enables the labeling process by providing a user-friendly platform to annotate images with bounding boxes and assign corresponding object classes. This tool simplifies the creation of a customized dataset for training the YOLOv8 model, ensuring accurate object detection and classification. RoboFlow's intuitive interface allows for seamless integration of labeled data into the YOLOv8's training pipeline, which makes it simpler to train a custom YOLOv8 model with a set of pictures that have been previously labelled on its platform.

We acknowledge that YOLO-OBB offers oriented bounding box detection, which would be highly beneficial for this use-case. However, its effective implementation requires a substantial and well-trained dataset [18], which exceeds our current objectives and compromises the simplicity of the proposed object detection pipeline. Therefore, we propose a simpler approach utilizing YOLOv8 for object detection, complemented by a straightforward image processing method to determine the 2D orientation of the object to be manipulated. Integrating OpenCV adds further value to our work, demonstrating our capability to incorporate additional components into our system, thereby enhancing its functionality.

D. Object Pose Estimation

The object Pose Estimation process is divided into 3 main parts. First, the result of the object detection performed by the YOLOv8 model is visualized within the Perspective Image,

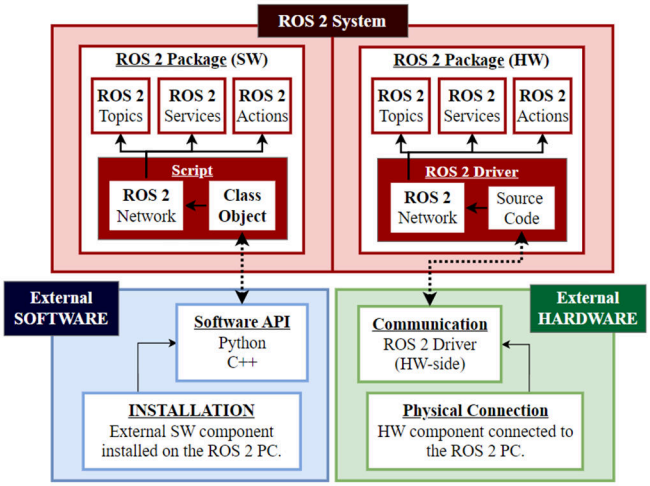


Fig. 3. Diagram of the integration of external SW and HW components and their functionality into the ROS 2-basd Robotic System.

[2] https://github.com/IFRA-Cranfield/ros2_SimRealRobotControl

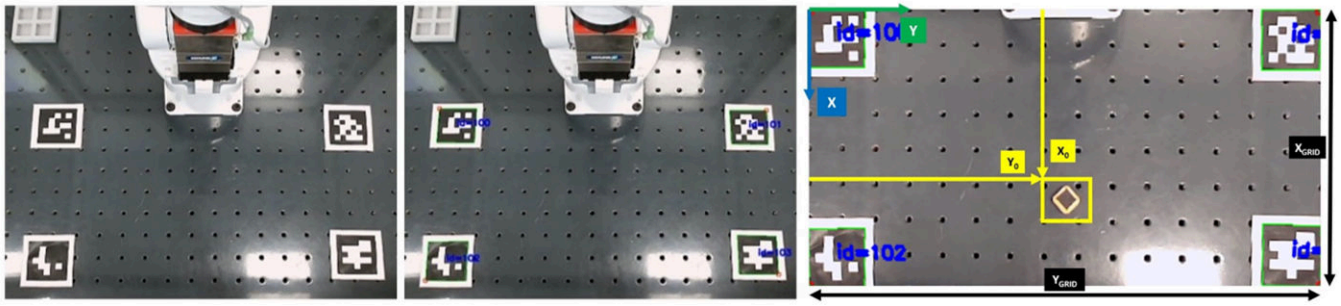


Fig. 4. ArUco Tag detection and calibrated image acquisition in the ABB IRB-120 Cell. Left: Input webcam feed; Center: Detection output with ArUco Tags located and labelled; Right: Transformed perspective image with coordinate frames and (x,y) values. In our system, camera calibration simplifies to 2D-calibration, assuming the cube is always on a known-height flat surface. Using OpenCV's *warpPerspective* function, we transform the camera input based on ArUco markers, ensuring accurate spatial understanding for the robot's manipulation of the cube. The x/X and y/Y pixel/distance ratio is known in the transformed image, where (x,y) denote Pixel Coordinates and (X,Y) represent Distance Coordinates (mm). This allows the accurate calculation of the object pose from a given image, by calculating its location in pixel coordinates first and subsequently applying the transformation ratio to get the distance value in millimetres.

the transformed image obtained after the calibration process (Fig. 4, right). Subsequently, the center of the object and its rotation are calculated using image processing techniques, and finally the values obtained are transformed from pixel coordinates to global robot coordinates.

The YOLOv8 model provides object class and yellow bounding box pixel coordinates, as depicted in the right picture of Fig. 4. To identify the center of the object within the provided bounding box, we employ contour detection (followed by color filtering) through image processing, using the OpenCV open-source library. After converting the image to HSV color space and applying Gaussian blur, contours are detected based on hue, saturation, and lightness. A precise bounding box, distinct from YOLOv8's, is assigned, incorporating orientation using OpenCV's *minAreaRect()* function. This process yields accurate object location in Pixel Coordinates, transformed to Distance Coordinates via calibration-derived pixel/distance ratios.

IV. Use-Case Demonstration

The practical implementation of our framework is exemplified through a complex object manipulation task, which serves not merely as a demonstration but as a pivotal evidence of our solution's ability to transcend traditional robotic limitations. This use-case is strategically chosen to emphasize the enhanced adaptability and efficiency our framework imparts to robot manipulators, addressing the core challenges of flexibility and task diversity in manufacturing environments. Central to our framework's implementation is the seamless integration of ROS 2, facilitating a cohesive ecosystem where hardware and software components interact harmoniously. The cube pick-and-place task leverages this integration, enabling a small industrial ABB robot to perform complex object manipulation tasks that extend beyond its conventional capabilities. The use-case has been designed and validated on Gazebo Simulation first, and has been deployed onto the real ABB IRB-120 cell enclosure at the Intelligent Automation Lab at Cranfield University. The system, equipped with a standard web camera, adeptly detects the randomly spawned cube within the workspace and proceeds to classify it based on its distinct colored feature. The intricate task of manipulating the cube involves precise pose estimation, ensuring its placement within the designated tray and alignment with the corresponding slot, with the identified feature positioned upwards (see Fig. 1).

This section follows a sequential approach mirroring the steps an engineer would take when deploying our framework for this pick-and-place application. Starting with the ROS 2 packages for the simulation and control of the robot, integrating YOLOv8 and OpenCV modules, designing object detection and pose estimation tasks, and concluding with the development of the program logic for the effective task execution within the robotic cell.

A. ROS 2 Framework: Integration of YOLOv8 and OpenCV

The ROS 2 package stack that controls the ABB Robot (red box in Fig. 5) consists of the Gazebo, MoveIt!2 and Bringup package combination, directly connected to the robot controller through the ABB-ROS 2 driver. Following the approach described in the methodology, the ABB IRB-120 is fully controlled and monitored within the ROS 2 framework, and its movements are executed and perceived through the custom ROS 2 Actions (/Move and /RobMove) in Python. In addition, it is worth mentioning that this framework allows the user to fully manage the robot exactly in the same way for both simulation and the real robot, enabling testing through simulation prior to real cell deployment.

Fig. 5 illustrates the architecture of the ROS 2 Framework for this specific use-case. The integration of the YOLOv8 and

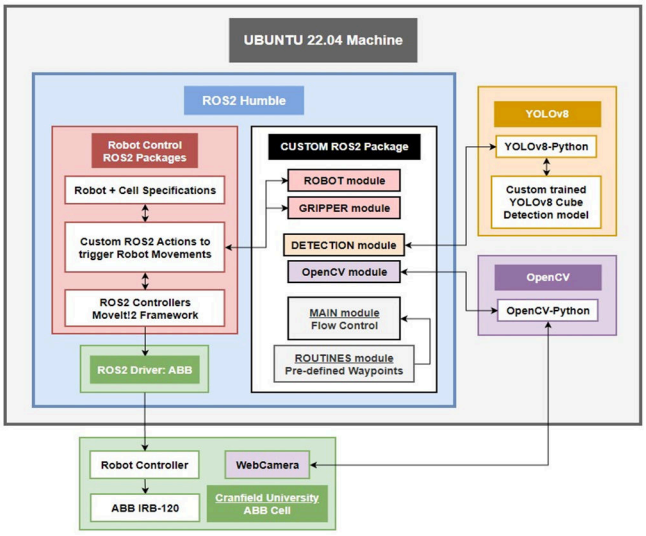


Fig. 5. Software Architecture, combining ROS2 Humble (Robot Control and Simulation), OpenCV, YOLOv8 modules and the real ABB IRB-120 Robot.



Fig. 6. Representation of the model training, object detection, classification and pose estimation pipeline. (A) Data acquisition and labelling process. Top-left: Sample acquisition from Gazebo Simulation; Bottom-left: Image transformation (cube isolation) as pre-processing to improve the training dataset; Right: Data labelling in RoboFlow. (B) Cube and Color Feature detection and classification using the custom trained YOLOv8 models in Real Robot and Gazebo Simulation. (C) Transformed Perspective and Region-Of-Interest (ROI) images for the accurate estimation of the Cube Pose.

OpenCV modules is realized through Python; a custom ROS 2 package has been created, where every single component of the system allows direct interaction through Python classes. Therefore, the Robot and Gripper (controlled through the ROS 2 packages introduced above), Detection (YOLOv8) and OpenCV modules can be easily executed and controlled from a single Python script, resulting in an intuitive and straightforward method to design and execute the program logic, which, in this case, consists of a cube pick-and-place task. The execution of any task done by a module on our system is based on a call-response method, enabling the engineer or operator to easily design any routine and the main flow of the program by simply calling each module execution or task one after the other.

This approach facilitates direct interaction between the robot manipulator and external software or hardware components responsible for specific tasks or decision-making within the manufacturing or production process. Consequently, it empowers the robot with enhanced capabilities, eliminating the necessity for PLCs, complex communication protocols, or additional automation equipment.

In the same manner that YOLOv8 and OpenCV have been seamlessly integrated into the system, any other hardware or software component can be incorporated through a custom ROS 2 package. This approach ensures that new components can communicate and coordinate with the robot using the robust communication mechanisms provided by ROS 2. Such integration empowers robotic engineers and researchers to enhance robot manipulators with diverse skills and functionalities, achieved through the integration of custom, sophisticated algorithms and software/hardware packages within the robot ecosystem.

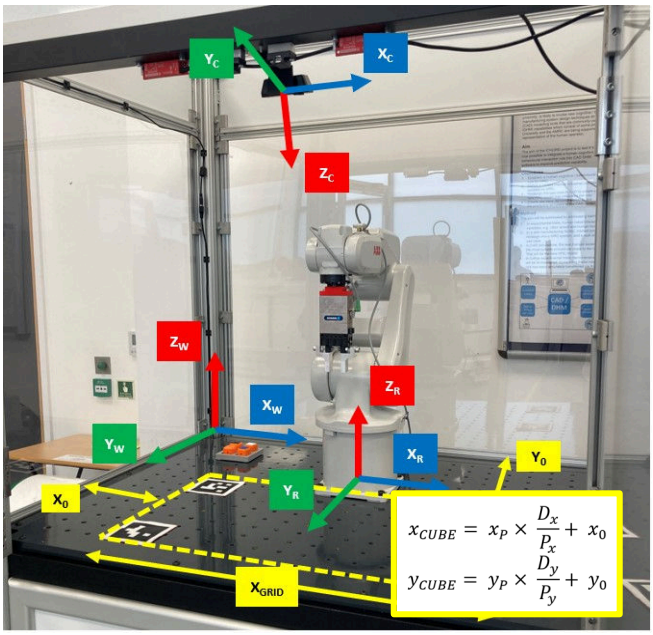


Fig. 7. ABB IRB-120 Enclosure setup for the Cube Detection and Pick-and-Place Use-Case. The Workspace Coordinate System (w), Robot Coordinate System (r), Camera Coordinate System (c) and the Detection Grid are represented. For transformation purposes, it is known that the detection grid is displaced (x_0, y_0) from the base (global coordinates).

B. Cube Detection, Classification and Pose Estimation

We have developed and trained two distinct YOLOv8 models for cube detection. One tailored for Gazebo simulation, and another specifically designed for the real ABB IRB-120 cell setup. Our training dataset encompasses a diverse range of images, categorized into five classes: Cube, Sticker, White, Black, and Blue. Introducing the "Sticker" class enables the robot to identify colored features atop the cube, particularly in scenarios where color differentiation is challenging. The training dataset consisted of 500 images for the Gazebo Simulation model and 1000 images for the real cell model, labelled and processed using RoboFlow, as shown in Fig.6A.

The tailored YOLOv8 models have been tested using the YOLOv8 “predict” function. As illustrated in Fig. 6B, the performance evaluation of both trained models demonstrated accurate cube detection with a confidence level exceeding 80% and precise color feature detection and classification, achieving a confidence level surpassing 60%.

Upon executing the YOLOv8 model, the output provides the detected object's class and bounding box, which is then transformed into a Region-of-Interest (ROI) image for further processing, as depicted in Fig. 6C. OpenCV is subsequently utilized to identify the contours of the object, enabling precise calculation of its position and rotation within the workspace. The location and orientation of the cube are determined in local pixel coordinates (relative to the ArUco grid, Fig. 7), with transformations to global coordinates computed through a two-step process as shown in the equations in Fig.7: Initially, transforming pixel values into distance values (mm), followed

by the addition of the grid offset. The relation between distance and pixel values is denoted by D/P, where (x0, y0) represents the grid's location within the workspace.

MoveIt!2 converts any 6DoF pose from global to robot coordinates. It automatically computes the pose of the robot given the cube's position in global coordinates. In our framework, this is realized through the /RobMove ROS 2 Action, which is used to move the robot according to a specific end-effector pose. Therefore, the result of the Pose Estimation module is directly feeded to the /RobMove action, which guides the robot to the cube pick-up position.

C. *Program Logic: Cube Pick-Check-and-Place*

The execution process unfolds as follows: Initially, the cube is randomly spawned within the grid defined by ArUco markers. Upon activation of the camera, the YOLOv8 model classifies the cube into two classes: cube or sticker. In the case of a sticker classification, indicating the presence of a colored feature atop the cube, the robot assesses whether the color type is discernible. If so, the relevant information is relayed to the robot, and the cube is placed in its corresponding slot on the kitting tray. In instances where color identification is challenging, the robot optimizes detection by moving the cube closer to the camera before proceeding with placement. For cubes classified as cube, denoting an unknown feature location, the robot initiates a systematic check of the front, back, and bottom sides. Successful detection prompts rotation for optimal feature visibility, culminating in placement on the kitting tray. Unsuccessful detection prompts a 90-degree rotation for examination of side faces. Subsequent successful detection results in final placement, while repeated failure leads to placement in the designated "no detections" slot on the kitting tray. This adaptive and thorough process ensures accurate classification and effective placement of cubes with diverse feature configurations.

V. Results Analysis

The performance evaluation for the use-case involved a comprehensive analysis based on 100 test runs in simulation and 78 in the real cell, with execution results and individual module success rates shown in Table 1, and overall execution success rates illustrated in Fig. 8. In each test run, a cube was randomly spawned within the designated workspace, adhering to the defined grid. Significant attention was devoted to evaluating the integration and communication between YOLOv8, OpenCV, the robot, and ROS 2. This involved recording every instance where each module executed a task and interacted with the program logic. Additionally, the evaluation criteria encompassed the accurate detection of the spawned cube, correct execution of the picking process, precise classification and location of the colored feature, and the correct placement of the cube onto the kitting tray.

	Simulation	Real Robot
Total Executions	100	78
Connectivity	Lost: 1 times	STABLE
Interactions w/ YOLOv8	449	280
Interactions w/ OpenCV	549	358
Executions: /RobMove	1142	760
Executions: /Move	2309	1853 vs 5
Cube Detection & Classification	100% Success	97.43% Success
Pose Estimation	100% Success	98.7% Success
Robot Performance	74% Perfect	74.36% Perfect

Table 1. Total number of interactions of the YOLOv8, OpenCV and Robot modules with the ROS 2 system, and the overall success of each module undertaking their specific task.

The integration of YOLOv8, OpenCV and robot movements (ROS 2 Actions) was rigorously tested, revealing robust communication between the modules and the system. In simulation, connectivity was lost only once (Gazebo Simulator stopped working), while in the real cell, the communication between the robot controller and ROS 2 remained stable during the whole testing process. Throughout all executions, 449(sim)/280(r) YOLOv8-System interactions and 549(sim)/358(r) OpenCV-System interactions were recorded, all successful. Additionally, the /RobMove and /Move actions were executed 1142 and 2309 times, respectively, in simulation, and 760 and 1858 times, respectively, in the real robot, with only five errors attributed to collision detection by MoveIt!2. The YOLOv8 detection results showcase a robust performance, revealing a remarkable success rate of over 97% in the initial cube detection process. Furthermore, the colored feature detection exhibited high accuracy, with only one failure noted. Overall, these findings emphasize the effectiveness of the YOLOv8 model and indicate the potential for achieving a 100% success rate in detecting specific objects within controlled workspaces with meticulous model training and refinement. The cube picking accuracy surpasses 98%, underscoring the exceptional quality and precision of the cube pose estimation process achieved through image processing.

However, the overall success rate of the entire process is influenced by challenges encountered during the cube placing phase, involving rotation, hovering, and placement steps. Notably, there is substantial room for improvement on the ROS 2 driver side of the ABB Robot, particularly concerning the precision, repeatability, and speed of the robot movements. Currently, these aspects fall short of the standards achieved by the native ABB Robot motion controlled through the ABB Robot Controller, and a refinement would lead to a more precise and reliable control of the robot's actions, thereby enhancing the overall performance and effectiveness of the robotic system.

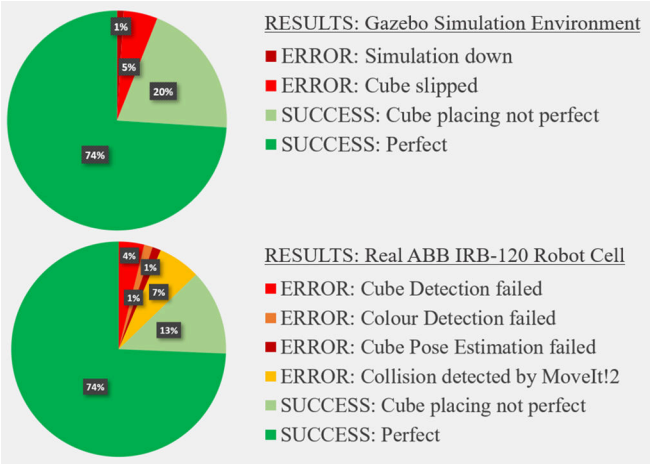


Fig. 8. Overall performance of the system undertaking the cube detection, pose estimation and pick-and-place task, in both simulation and the real robot.

VI. Conclusion

Our research presents a novel ROS 2-based framework that markedly simplifies the deployment process for robotic cells, irrespective of the robot, its end effector or application. This innovation enables engineers and developers to pivot their focus towards refining application logic, rather than being encumbered by the intricacies of integration and deployment tasks. At the heart of our framework lies its ability to streamline the deployment process, making it straightforward and uniform across different types of robotic applications. This universality not only democratizes access to advanced robotics for small and medium-sized enterprises in the automation industry but also significantly reduces the time and resources typically required for custom robotic solutions.

Our research offers a reproducible and extendable solution, demonstrated with the object manipulation use-case on the ABB IRB120 Robot, significantly advancing industrial and collaborative robotics. The inherent flexibility, scalability, and modularity of the ROS 2 architecture make it ideal for reconfigurable systems, supporting diverse operations and frequent reprogramming needs.

Looking ahead, our future endeavors will concentrate on standardizing interfaces for the ROS 2 integration of external hardware and software, further enriching the ecosystem's capabilities. This effort is not just a step towards enhancing our framework but also a call to action for the community to coalesce around the common goal of advancing robotic simulation and control technologies. In sum, our research contributes a significant leap towards more adaptable, accessible, and efficient robotic deployments, promising to shape the future of robotics in manufacturing and beyond. Our commitment to open collaboration and continuous improvement underlines our belief in the transformative potential of ROS 2-based frameworks in the ever-evolving landscape of industrial robotics.

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 invaluable 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] E. Trunzer, S. Lötzerich, and B. Vogel-Heuser, "Concept and Implementation of a Software Architecture for Unifying Data Transfer in Automated Production Systems," 2018, pp. 1–17. doi: 10.1007/978-3-662-57805-6_1.

[4] F. Gosewehr, J. Wermann, W. Borsych, and A. W. Colombo, "Specification and design of an industrial manufacturing middleware," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, IEEE, Jul. 2017, pp. 1160–1166. doi: 10.1109/INDIN.2017.8104937.

[5] S. Karnouskos, A. W. Colomboy, F. Jammesz, J. Delsing, and T. Bangemann, "Towards an architecture for service-oriented process monitoring and control," *IECON Proceedings (Industrial Electronics Conference)*, pp. 1385–1391, 2010, doi: 10.1109/IECON.2010.5675482.

[6] P. Leitão, J. Barbosa, A. Pereira, J. Barata, and A. W. Colombo, "Specification of the PERFoRM architecture for the seamless production system reconfiguration," *IECON Proceedings (Industrial Electronics Conference)*, pp. 5729–5734, Dec. 2016, doi: 10.1109/IECON.2016.7793007.

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

[8] 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.

[9] 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.

[10] J. He, J. Zhang, J. Liu, and X. Fu, "A ROS2-Based Framework for Industrial Automation Systems," in *2022 2nd International Conference on Computer, Control and Robotics (ICCCR)*, IEEE, Mar. 2022, pp. 98–102. doi: 10.1109/ICCCR54399.2022.9790247.

[11] 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.

[12] 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.

[13] 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.

[14] A. S. Olesen, B. B. Gergaly, E. A. Ryberg, M. R. Thomsen, and D. Chrysostomou, "A Collaborative Robot Cell for Random Bin-picking based on Deep Learning Policies and a Multi-gripper Switching Strategy," *Procedia Manuf*, vol. 51, pp. 3–10, 2020, doi: https://doi.org/10.1016/j.promfg.2020.10.002.

[15] X. Yue, H. Li, M. Shimizu, S. Kawamura, and L. Meng, "YOLO-GD: A Deep Learning-Based Object Detection Algorithm for Empty-Dish Recycling Robots," *Machines*, vol. 10, no. 5, 2022, doi: 10.3390/machines10050294.

[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] S. Zhou *et al.*, "Framework of rod-like crops sorting based on multi-object oriented detection and analysis," *Comput Electron Agric*, vol. 216, p. 108516, Jan. 2024, doi: 10.1016/j.compag.2023.108516.

[18] G. Jocher, "Oriented Bounding Box (YOLO-OBB): Datasets Overview," https://docs.ultralytics.com/datasets/obb/.



https://dspace.org/

Unlocking the potential of robot manipulators: seamless integration framework

Bueno, Mikel; Bernardino, Irene; Asif, Seemal; Webb, Phil

2024-08-28
Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/case59546.2024.10711476
Bueno M, Bernardino I, Asif S, Webb P. (2024) Unlocking the potential of robot manipulators: seamless integration framework. In: 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE). 28 August 2024 - 1 September 2024, Bari, Italy, pp. 2610-2617

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