2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Development of a semi-autonomous manipulation pipeline for robotic shelf-picking operations

David Vázquez[†], Piero Vega Gutiérrez[†], Rafael Cisneros-Limón[†*], Kenji Kaneko[†], Fumio Kanehiro[†], Luis Alberto Muñóz[‡]

***Abstract*—This paper presents a modular manipulation pipeline for CALL-M, a mobile robot developed at CNRS-AIST JRL for semi-autonomous pick-and-place operations in convenience store environments. The system leverages a ROS 2-based architecture integrating 3D perception, grasp detection, and motion planning using *MoveIt 2*. The pipeline comprises modular stages—point cloud acquisition, object selection, grasp estimation, and trajectory generation—coordinated by a centralized task manager.**

Validation in both simulation and real-world scenarios demonstrated successful grasps. While simulation confirmed reliability under ideal conditions, real-world trials revealed challenges due to sensor noise, workspace constraints, and misalignments in grasp pose generation. Despite this, the system's modularity and adaptability make it a scalable solution for manipulation in semi-structured environments.

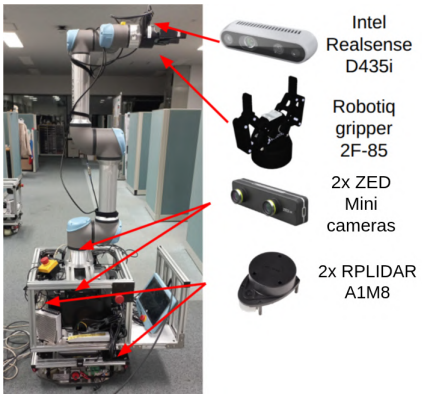
***Index Terms*—Grasping, Manipulation, Motion Planning, ROS2, Robotics**

I. Introduction

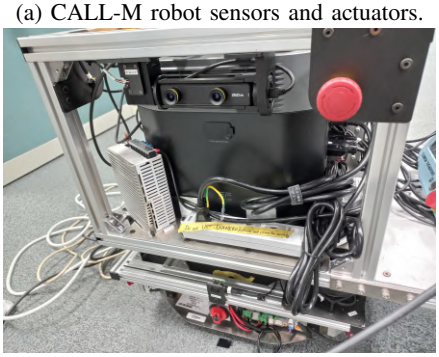
Mobile manipulators combine locomotion with dexterous manipulation, enabling complex tasks across industrial, service, and domestic domains. They support automation of repetitive labor, remote assistance, and operation in hazardous or dynamic environments [1], [2].

These systems typically follow one of three paradigms: manual, semi-autonomous, or fully autonomous. Manual systems rely on teleoperation and remain effective in unstructured environments like home care [2]. Fully autonomous systems perform end-to-end tasks in structured settings such as factories and warehouses [3], [4]. Semi-autonomous systems strike a balance by allowing high-level human input while autonomously handling low-level tasks like grasping and navigation [5].

This paper introduces a semi-autonomous manipulation pipeline for CALL-M [6], a mobile manipulator developed at CNRS-AIST JRL[1]. CALL-M features a *TriOrb* omnidirectional base[2], a *UR5e* arm with a *Robotiq 2F-85* gripper, and an *Intel Realsense D435i* depth camera (Fig. 1c). For navigation and obstacle awareness, it also includes two *ZED Mini* cameras (Fig. 1b) and two *RPLIDAR A1M8* LIDARs (Fig. 1a). The intended use of this robot is to improve the physical workload of convenience store workers through the use of semi-autonomous manipulation technologies, mainly



(a) CALL-M robot sensors and actuators.



(b) Frontal *ZED Mini* camera for navigation.

(c) End-effector-mounted *Intel Realsense D435i* at an angle of 45°.

Fig. 1: Overview of the CALL-M robot: sensors and cameras.

[†]CNRS-AIST JRL (Joint Robotics Laboratory), IRL3218, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan. Emails: david-vazquezleal@aist.go.jp, piero.vega-gutierrez@univ-tlse3.fr, rafael.cisneros@aist.go.jp, k.kaneko@aist.go.jp, f-kanehiro@aist.go.jp
[‡]School of Engineering and Sciences, Tecnológico de Monterrey, Mexico. Email: amunoz@tec.mx
*Corresponding author

[1]https://unit.aist.go.jp/isri/isri-jrl/en/
[2]https://TriOrb.co.jp/

for product management (i.e., stock and retrieve products) operations without modifying the environment where the system is intended to be used. This scenario reflects the need for technology to avoid physically performing long repetitive tasks, with the potential to be fully automated and scaled (see [7]).

The main contribution of this work is a modular, ROS 2-based software architecture for semi-autonomous grasping of unknown objects using real-time perception [8], [9], grasp detection [10], and Cartesian motion planning [11]. The system is validated in simulation and real-world settings, demonstrating successful grasp execution and revealing key limitations in constrained environments.

The remainder of this paper is organized as follows: Section II reviews related work; Section III details the system architecture; Section IV describes the grasping pipeline; Section V presents experimental results; and Section VI concludes with future directions.

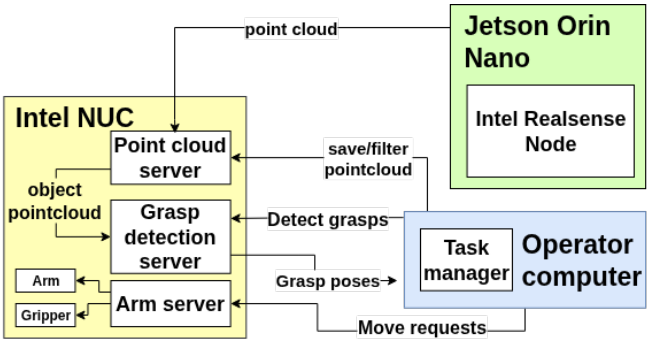


Fig. 2: ROS 2-based software architecture of the CALL-M robot. The Jetson Orin Nano (right-up block) handles point cloud acquisition and filtering. The Intel NUC (left-low block) runs ROS 2 nodes for grasp detection, planning, and control. The *task manager* coordinates the sequence of operations via service calls.

II. Related Work

Mobile manipulators have gained prominence in robotics by combining navigation and manipulation capabilities through multi-modal sensor integration. They have been deployed at varying levels of autonomy—ranging from manual control to fully autonomous operation—across diverse domains.

For instance, [2] presented a home-assistance robot for motor-impaired users, integrating a minimal user interface (e.g., head tracker) with autonomous navigation and grasping. Similarly, [12] demonstrated a fully autonomous mobile robot capable of retrieving and delivering store items using onboard perception and planning, even navigating elevators and queues without environmental modifications.

Hybrid systems strike a balance between autonomy and user control. In [1], operators manually defined grasp and drop points for trash collection, while the robot autonomously executed manipulation actions, demonstrating the effectiveness of shared control in semi-structured environments.

In contrast, the CALL-M system targets semi-autonomous manipulation in cluttered environments such as the shelves in convenience stores. It follows a semi-autonomous paradigm where grasp pose generation and motion planning are automated, while object identification and filtering are performed by the user via RViz.

Unlike many state-of-the-art approaches that rely on extensive object models, complex perception stacks, or fully autonomous decision-making, the manipulation pipeline for CALL-M is designed to be lightweight and accessible. This design reduces cognitive and manual effort by offloading key manipulation processes to the system, while maintaining human oversight.

The proposed ROS 2-based pipeline offers several advantages: (1) modularity for integrating alternative sensors or planners; (2) user-transparent control through interactive object selection; and (3) improved reliability via real-time OctoMap updates, which enhance planning safety under imperfect perception.

III. Proposed System Architecture

The CALL-M robot employs a distributed computing architecture across two onboard computers to balance computational load and support real-time operation. A Jetson Orin Nano handles sensor data acquisition and point cloud processing, interfacing with the end-effector-mounted *Intel Realsense D435i* and two side-mounted *ZED Mini* cameras. Meanwhile, an Intel NUC is responsible for motion planning and control of both the *UR5e* robotic arm and the *TriOrb* omnidirectional base. Each PC manages its directly connected components and communicates via a shared ROS 2 network (see Fig. 2).

System behavior is coordinated by a central script—the *task manager*—implemented as a state machine, similar to the approach in [13]. It orchestrates pipeline execution by sending service calls to modular ROS 2 server nodes responsible for point cloud filtering [9], grasp detection [10], and motion planning [11], [14].

The *task manager* runs externally on the operator's PC. Since it does not require direct hardware access, it can be executed on any machine within the ROS 2 network, allowing for seamless deployment in both simulation and real-world scenarios.

The proposed system consists of four main software modules, each implemented as a ROS 2 node or script to enable semi-autonomous object manipulation:

A. Task Manager

The *task manager* serves as the control unit of the pipeline. It communicates with other nodes using predefined ROS 2 clients and executes robot actions based on an internal state machine. This architecture allows for modular integration of high-level logic with low-level control.

B. Point Cloud Server

This node processes depth data from the *Intel Realsense D435i* (see Fig. 1c). Captured point clouds are saved as PCD files using the PCL library [9]. A fixed-height filter removes shelf surfaces, assuming that shelf levels remain constant, thus isolating potential grasp targets. It also supports multi-view concatenation (see Fig. 4) for better reconstruction of partially occluded objects.

C. Grasp Pose Detection Server

This module wraps the GPD (Grasp Pose Detector) algorithm [10] as a ROS 2 service. It receives a filtered point cloud and a configuration file defining the gripper model, approach directions, and scoring parameters. GPD uses a CNN (Convolutional Neural Network) to evaluate grasp candidates and returns the highest-scoring pose to the task manager (see Fig. 6).

D. Arm Server

The arm server provides services to control the *UR5e* arm via MoveIt 2 [11]. It supports both pose-based planning using OMPL (Open Motion Planning Library) [14] and Cartesian trajectories for fine motion control. Integration with OctoMap [8] enables real-time collision checking using a 3D occupancy map, ensuring safe manipulation in cluttered environments (see Fig. 3).

IV. Methodology

To perform semi-autonomous grasping of objects placed on shelves, the CALL-M manipulation pipeline is organized into five sequential stages: environment scanning, point cloud acquisition, object filtering and selection, grasp pose detection, and path planning. Each stage corresponds to a specific state within the *Task Manager*, which coordinates the execution of the individual ROS 2 modules (server nodes, Fig. 2).

A. Environment Scanning

Environment scanning combines autonomous and manual steps to construct a collision-aware 3D occupancy map using OctoMap [8], [11]. The robot first performs four 90° in-place rotations to capture a panoramic view of the surroundings for initial obstacle mapping (see Fig. 3).

To further refine the map, the operator manually guides the arm to additional viewpoints (Fig. 4). Point clouds from these poses are fused into the existing OctoMap to enhance scene fidelity. Although OctoMap visualization is hidden after initialization, its voxel data remains active and continues to support collision-aware planning.

B. Camera Positioning and Point Cloud Acquisition

This stage focuses on capturing point clouds of the object. The operator guides the arm to different viewpoints, with live and accumulated point clouds shown in RViz to assist view selection (Fig. 4). Unlike the previous stage, this phase emphasizes object detail rather than obstacle mapping.

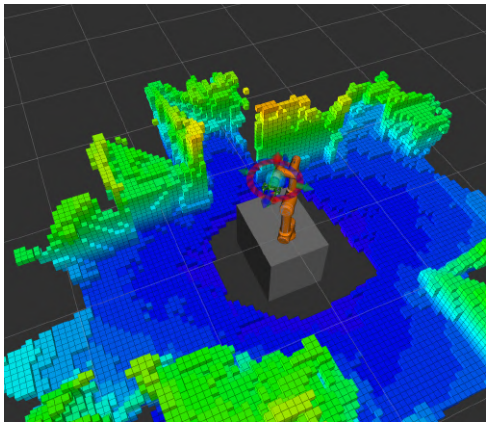


Fig. 3: OctoMap generated during the environment scanning phase using the end-effector-mounted depth camera.

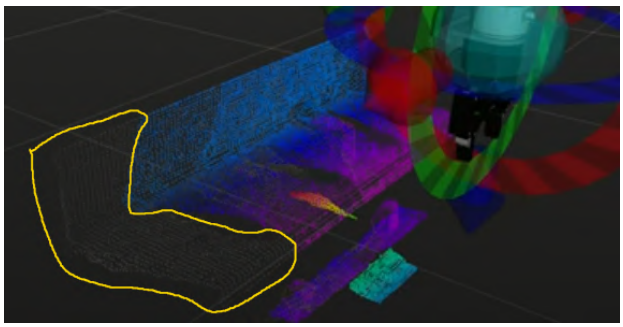


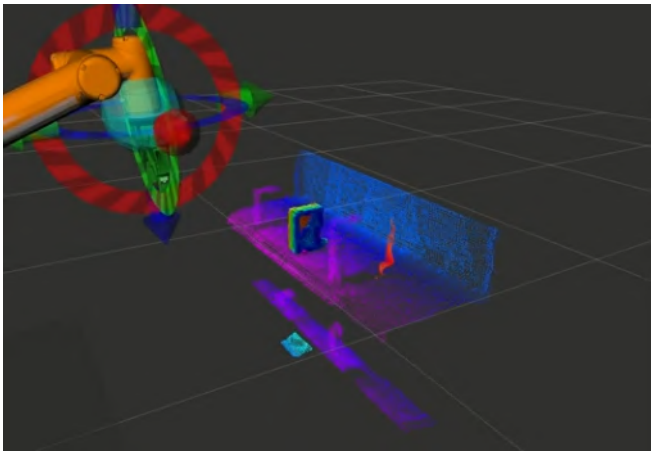
Fig. 4: Accumulated point cloud (colored) overlaid with the live camera feed. Points highlighted with the yellow contour at the left end of the shelf represent the most recent sensor data.

To reflect typical convenience store scenarios, the object is assumed to rest on a shelf at a known approximate height, which simplifies the filtering process. Multiple views are concatenated into a single point cloud for subsequent analysis.

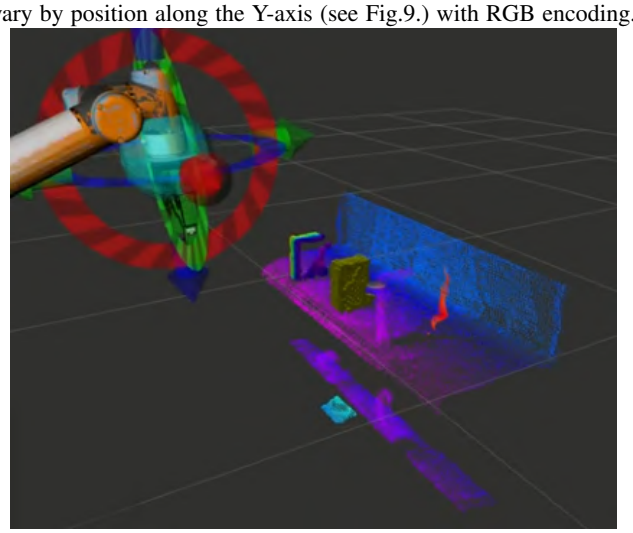
C. Point Cloud Filtering and Object Selection

After the complete point cloud is built, a two-stage filtering process is applied. First, a fixed-height filter removes all points that fall outside the vertical range of the shelf. This is based on the assumption that shelf levels in convenience store-like environments are placed at known, consistent heights. By keeping only the points within a narrow height band, the system eliminates irrelevant surfaces such as the floor, ceiling, or higher/lower shelf levels.

Next, Euclidean clustering is performed using the PCL library [9] to segment individual object candidates (see Fig. 5a). Then, the operator selects the target object by clicking on it using the RViz *Publish Point* tool [15]. The system identifies the nearest cluster using a KD-tree [16] search and highlights it for confirmation (Fig. 5b). Once confirmed, grasp pose detection is triggered.



(a) Object candidates segmented using Euclidean clustering. Colors vary by position along the Y-axis (see Fig.9.) with RGB encoding.



(b) Selected cluster highlighted in yellow after user input in RViz.

Fig. 5: Object filtering and selection using height filtering and Euclidean clustering.

D. Grasp Pose Detection

Grasp pose candidates are generated using the Grasp Pose Detection (GPD) algorithm [10], which is integrated into the system as a ROS 2 service. The service receives a point cloud along with a configuration file that defines the gripper geometry, approach constraints (such as prioritizing grasps along the$\pm$Y axis, see Fig. 9), and camera viewpoints. GPD evaluates each candidate pose using a CNN trained to predict grasp success likelihood. The pose with the highest confidence score is returned to the task manager for execution (see Fig. 6).

E. Path Planning and Execution

Upon receiving the selected grasp pose, the task manager invokes the arm server, which uses the Open Motion Planning Library (OMPL) [14] in conjunction with MoveIt 2 [11] to compute a collision-free trajectory. For approach motions—such as closing in on the object—Cartesian planning

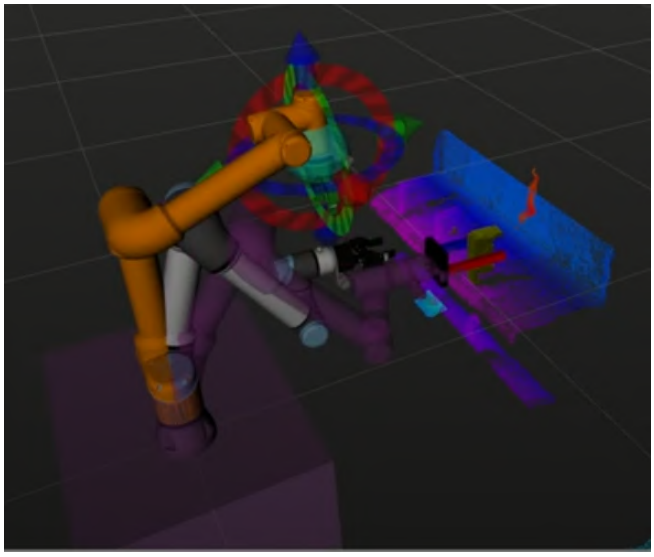


Fig. 6: Grasp planning and execution visualization. The orange frame represents the operator-defined pre-grasp pose. The blue-and-red gripper shows the best grasp candidate returned by GPD [10]. The purple model indicates the planned trajectory of the arm using MoveIt 2, while the robot in true colors is shown executing the motion of the real one.

is employed to ensure precision and linearity. Once the end-effector reaches the target pose, the gripper closes to secure the object (Fig. 6).

V. Experiments

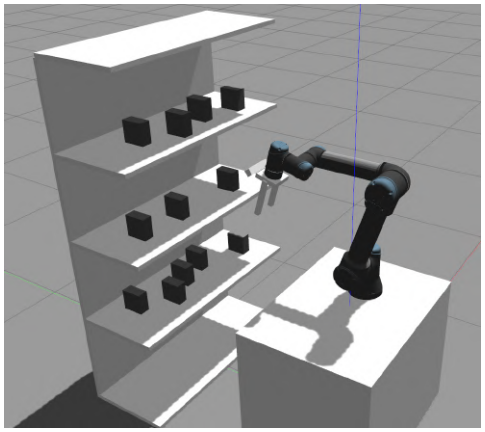
The manipulation pipeline was evaluated through a series of pick-and-place tasks in both simulation and real-world environments. These tests aimed to validate the system's robustness and modular design, focusing on the coordination of point cloud acquisition, grasp pose detection, and motion planning. Evaluations involved grasping previously unseen, box-like and cylindrical objects.

A. Simulation Tests

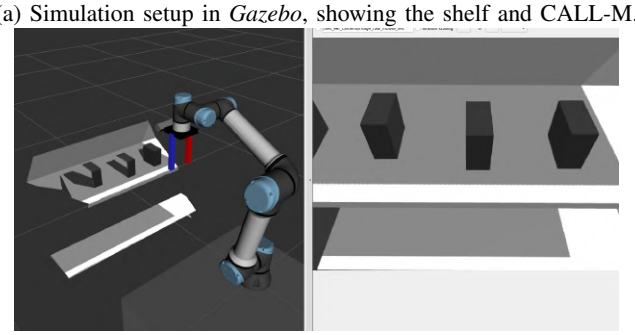
Simulation experiments were performed using the *Gazebo* simulator [17], chosen for its compatibility with URDF models and its simulation of sensors and actuators through *gazebo_ros_pkgs* [18]. The simulated platform included a *UR5e* robotic arm [19], a generic two-finger gripper, and a 3D camera simulated using the *gazebo_ros_openni_kinect* plugin.

The robot was positioned facing a five-level shelf (Fig. 7a), each level providing 40 cm of vertical clearance. Objects were randomly distributed on the shelves with similar orientations—rectangular objects placed such that one external face was parallel to the shelf surface to simplify grasping.

During the trials, the robot remained stationary while the operator scanned the environment and selected objects using RViz [15] (see Fig. 7b), triggering grasp pose detection and motion planning via the *Publish Point* tool. Although grasp



(a) Simulation setup in *Gazebo*, showing the shelf and CALL-M.



(b) Point cloud and simulated camera 2D image.

Fig. 7: Simulation and visualization interfaces used to validate the manipulation pipeline.

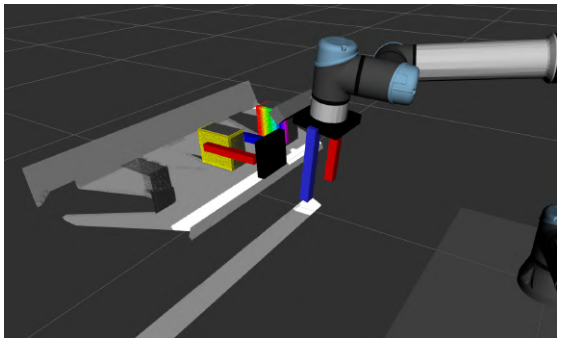


Fig. 8: Grasp pose detection performed on the user-selected cluster (highlighted in yellow). An example of an identified object cluster is shown using RGB encoding.

execution was not simulated[3], the robot consistently reached the planned end-effector poses, validating the pipeline's performance under ideal conditions (see Fig. 8).

B. Real-World Experiments

The system's real-world performance was evaluated with the CALL-M robot positioned 0.45 m from a physical shelf,

[3]Grasping actions were omitted in simulation due to the complexity and limited fidelity of simulating grasp dynamics in Gazebo. The evaluation focused on the computation of valid grasp poses and generation of safe trajectories.

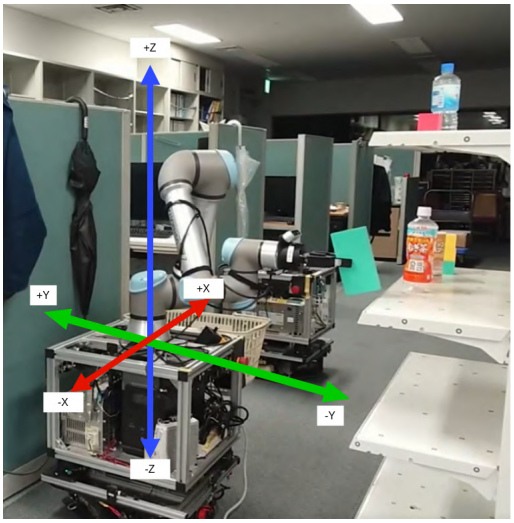


Fig. 9: CALL-M's coordinate frame used for transforming grasp poses.

measured from the robot's front edge to the shelf's nearest edge. The shelf measured 2.1 m in height and 1.7 m in width, with four levels spaced 0.3 m apart. Only the third level was used for testing, as the lower two and uppermost levels were excluded due to limited reachability and potential kinematic singularities near the robot's base. Grasping trials involved box-shaped toys and cylindrical containers such as PET bottles.

During all trials, the robot remained stationary. Object selection and point cloud acquisition were performed using RViz. Once a target object was selected, the point cloud was filtered, a grasp pose was computed using GPD [10], and a Cartesian trajectory was planned and executed. The object was deposited into a container (attached basket) by reversing the same trajectory (see Fig. 12).

A total of 45 trials were conducted across nine experimental conditions, defined by three lateral placements (center, left, and right) and three distances from the shelf edge (0, 8, and 15 cm), with five repetitions per condition. The lateral positions were spaced 20 cm apart from the center.

Grasp success rates were recorded to evaluate performance. An example of a successful grasp is shown in Fig. 10, and the overall results are summarized in Table I.

In addition to success rates, the Euclidean error between

TABLE I: Grasping Success: Successful attempts / repetitions (Success Rate)

Distance (cm)	Position	Successful Attempts / Repetitions (Rate)
0	Center	5/5 (100%)
	Left	5/5 (100%)
	Right	4/5 (80%)
8	Center	2/5 (40%)
	Left	3/5 (60%)
	Right	4/5 (80%)
15	Center	2/5 (40%)
	Left	2/5 (40%)
	Right	3/5 (60%)

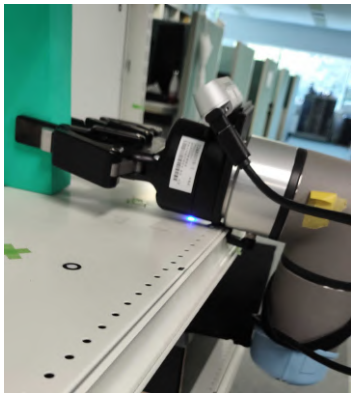


Fig. 10: Example of a grasping operation with the object placed 15 cm from the shelf edge.

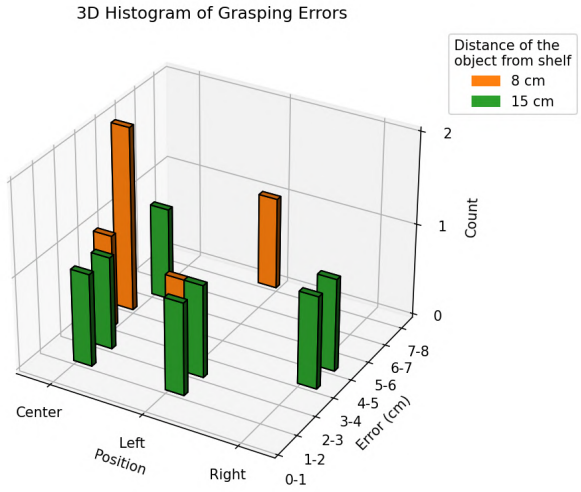


Fig. 11: Distance error between planned grasp poses and actual object positions in failed trials for each lateral position. The 0 cm distance was omitted because there were no errors in those trials.

the planned grasp pose and the actual object location in failed attempts was analyzed. Fig. 11 shows the error distribution, highlighting a correlation between object distance and localization error.

C. Observations and Analysis

The experimental results reveal several key trends:

- **Close proximity improves success:** At a distance from the shelf of 0 cm, grasping success reached 100% in most cases. Performance declined noticeably at 8 and 15 cm, particularly at the center and left positions.
- **Right-side grasps were more robust:** Success rates on the right remained relatively high (60–80%) across all distances, possibly due to better sensor coverage or more favorable arm kinematics.
- **Greater distance increased perceptual error:** Increased object distance degraded point cloud quality and localization accuracy, contributing to grasp pose failures.
- **Center grasps were most affected by error:** Success rates at the center dropped from 100% to 40% as distance increased, likely due to occlusions or reduced depth sensing along the robot's frontal axis.

While the pipeline performed reliably under controlled conditions, its performance degraded in more realistic scenarios involving sensor noise, workspace constraints, and object misalignment. Notably, the GPD module [10] often failed to generate viable grasp candidates when objects were not horizontally aligned. The current system supports only horizontal approach vectors (Fig. 13), limiting its robustness when handling diverse object poses.

These findings highlight the need for more adaptive grasp estimation and trajectory planning strategies to improve performance in unstructured environments. Enhancing the perception stack (with higher-fidelity cameras or improved depth estimation techniques) could also reduce point cloud errors and increase grasping reliability.

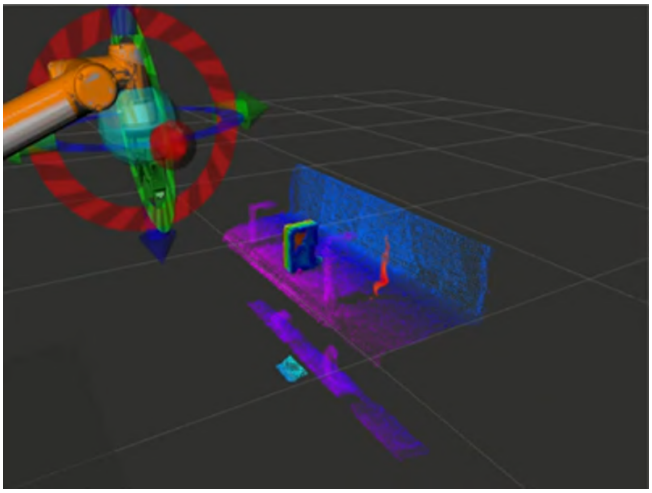
VI. Conclusions

This paper presented a modular manipulation pipeline for the CALL-M mobile manipulator, developed on a ROS 2-based architecture that integrates point cloud acquisition, grasp pose detection, and motion planning. The system was evaluated through simulation using *Gazebo* and real-world experiments with a *UR5e* robotic arm and a *Robotiq 2F-85* gripper.

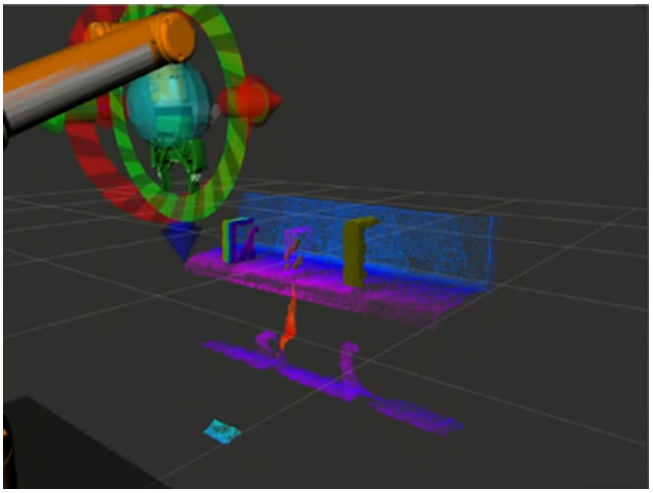
Simulation results demonstrated that the pipeline can consistently generate valid, collision-free trajectories under ideal conditions. Real-world experiments validated the system's practical viability but also revealed limitations related to sensor noise, workspace constraints, and difficulties in grasping non-aligned objects. Additionally, although no critical errors were observed during testing, executing Cartesian trajectories near the boundaries of the arm's workspace may pose control and stability challenges due to proximity to kinematic singularities. Another limitation is the lack of consideration for the grasped object when retrieving the arm, which could cause collisions with nearby objects in more challenging scenarios.

Despite these considerations, the results validate the feasibility of a modular, semi-autonomous manipulation framework. The system successfully completed pick-and-place operations across varying positions and distances, highlighting its adaptability and modular structure as a foundation for further development.

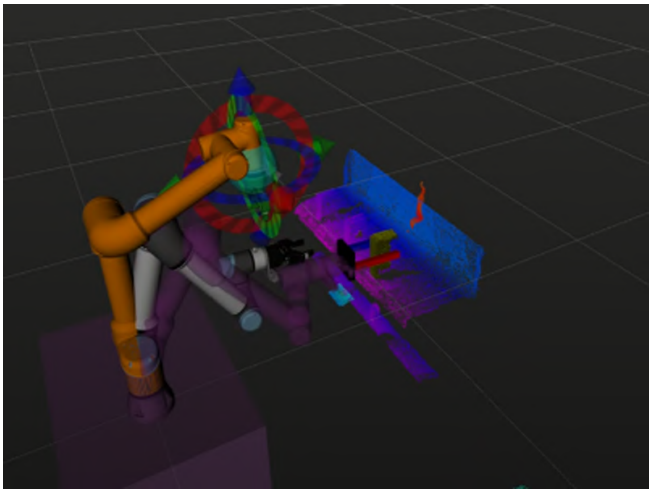
Future work will focus on increasing robustness in real convenience store operations by focusing on cluttered and dynamic environments. Key directions include improving grasp pose estimation, enhancing point cloud filtering under noisy conditions, and developing more adaptive motion planning strategies that explicitly account for interactions between the grasped objects and their surroundings. Integrating mobile base repositioning to improve reachable workspace and relaxing constraints on Cartesian planning are expected to significantly boost the robot performance in multi-object restocking or item-retrieval scenarios.



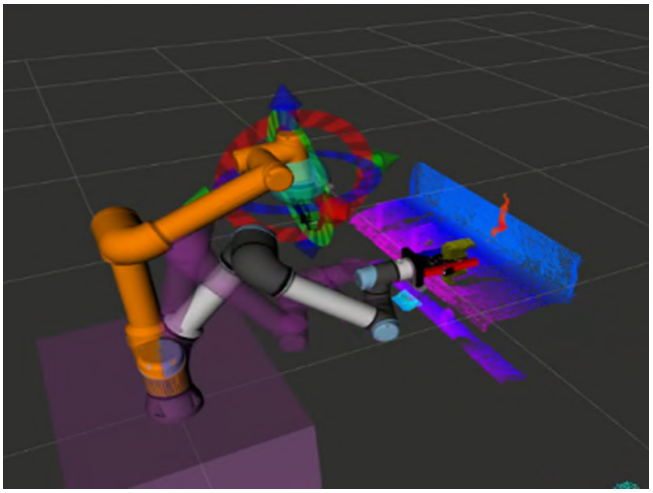
(a) Filtering phase: cluster identification from the point cloud.



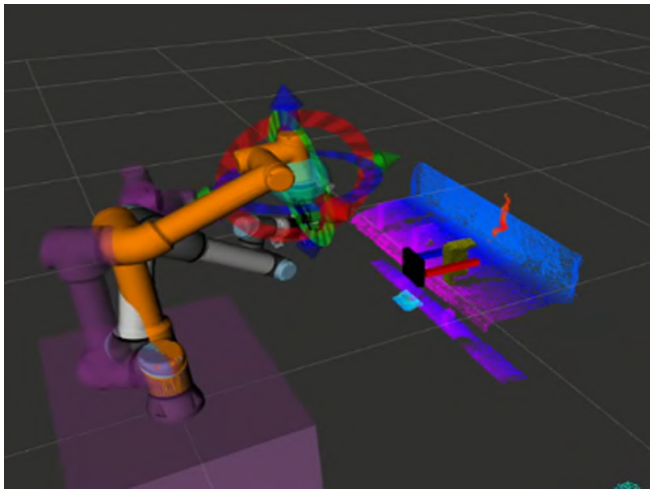
(b) Selection phase: user selects a cluster using RViz [15].



(c) Arm planning toward the grasping pose.



(d) Planning to a pre-grasp pose.



(e) Arm reaching the object drop location.

Fig. 12: Sequence of the autonomous grasping execution: cluster filtering (same color code as Fig. 5 and Fig. 6), selection, planning, and object deposition.

REFERENCES

[1] B. Zapata-Impata, V. Shah, H. Singh, and R. Platt, "Autotrans: an autonomous open world transportation system," 10 2018.

[2] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, "Mobile manipulation through an assistive home robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5313–5320.

[3] N. Ghodsian, K. Benfriha, A. Olabi, V. Gopinath, and A. Arnou, "Mobile manipulators in industry 4.0: A review of developments for industrial applications," *Sensors*, vol. 23, no. 19, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/19/8026

[4] W. Xu, "From automation to autonomy and autonomous vehicles: Challenges and opportunities for human-computer interaction," *interactions*, vol. 28, 01 2021.

[5] S. Al-Hussaini, S. Thakar, H. Kim, P. Rajendran, B. C. Shah, J. A. Marvel, and S. K. Gupta, "Human-supervised semi-autonomous mobile manipulators for safely and efficiently executing machine tending tasks," 2020. [Online]. Available: https://arxiv.org/abs/2010.04899

[6] O. Noel, R. Cisneros-Limón, K. Kaneko, and F. Kanehiro, "A Lightweight Approach to Efficient Multimodal 2D Navigation and Mapping: Unified Laser-Scans as an Alternative to 3D Methods," in *2025 IEEE/SICE International Symposium on System Integration (SII)*, 2025, pp. 617–624.

[7] M. Seki, K. Wada, and T. Tomizawa, "Development of automated display shelf system for new purchasing experience by dynamic product layout changes," *Journal of Robotics and Mechatronics*, vol. 36, no. 6, pp. 1527–1536, 2024.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at https://octomap.github.io. [Online]. Available: https://octomap.github.io

[9] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4.

[10] A. ten Pas, M. Gualtieri, K. Saenko, and R. W. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, pp. 1455 – 1473, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3990641

[11] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study," *ArXiv*, vol. abs/1404.3785, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:13939653

[12] A. Pratkanis, A. E. Leeper, and K. Salisbury, "Replacing the office intern: An autonomous coffee run with a mobile manipulator," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013, pp. 1248–1253.

[13] A. Flores, E. Flores, I. Romero, A. Chapa, F. Salas, A. Coeto, D. Vazquez, D. Hernandez, O. Arreola, Y. Reyes, A. Guerrero, M. Villanueva, A. Cervantes, A. Munoz, and C. Vazquez, "RoBorregos Team Description Paper for RoboCup @Home 2024," 2023, team Description Paper. [Online]. Available: https://athome.robocup.org/wp-content/uploads/OPL-RoBorregosTDP2023.pdf

[14] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

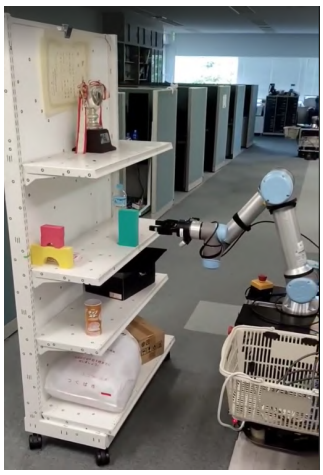
[15] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: a toolkit for real domain data visualization," *Telecommunication Systems*, vol. 60, pp. 337 – 345, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:6697099

[16] J. Friedman, J. Bentley, and R. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, pp. 209–226, 09 1977.

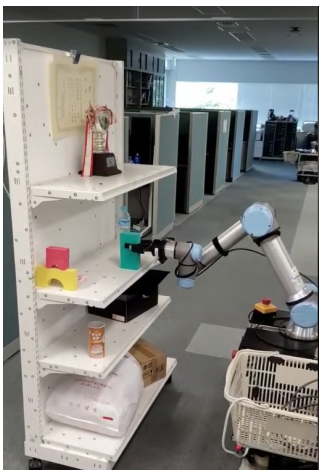
[17] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[18] R. S. Team, "gazebo_ros_pkgs: ROS 2 packages for Gazebo simulation," 2024, accessed: 2024-11-29. [Online]. Available: https://github.com/ros-simulation/gazebo_ros_pkgs

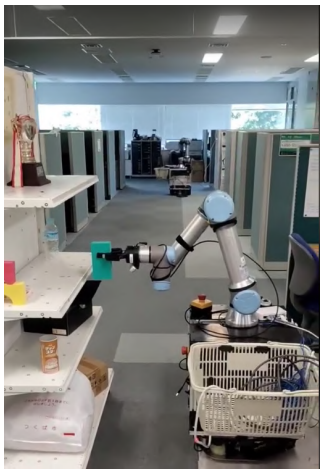
[19] U. Robots, "Universal Robots ROS2 Gazebo Simulation," https://github.com/UniversalRobots/Universal_Robots_ROS2_Gazebo_Simulation, 2024, accessed: 2024-11-29.



(a) Motion planning in execution after the user selects the target.



(b) Grasping execution after reaching the desired pose.



(c) Arm reversing the extraction motion.

(d) Arm releasing the object into the basket.

(e) Arm returns to the initial observation pose.

Fig. 13: Snapshots of the robot executing the pipeline in a real-world scenario.