A software architecture for low-resource autonomous mobile manipulation

Pangcheng David Cen Cheng, Marina Indri, Federico Maresca, Antonio Ragazzo, Fiorella Sibona
Dipartimento di Elettronica e Telecomunicazioni
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
{pangcheng.cencheng, marina.indri, fiorella.sibona}@polito.it
fmare94@gmail.com, antonio.ragazzo@outlook.it

***Abstract*—Mobile manipulators can significantly contribute to enhance the flexibility of several processes, such as automated order-picking systems and various logistic applications, thanks to their capability to manipulate objects and deliver them to different locations. A primary role is envisaged for them in Smart Factories, as workmates of the operators, if they are able to safely navigate in human-shared environments. This paper proposes a lightweight and flexible ROS1-based software architecture, designed for low-resource mobile manipulators, to make them able to autonomously search for the items requested by a human operator, independently from the starting pose, and pick and place them in a predefined depot location. The validity of the proposed architecture, which is potentially applicable to different low-resource mobile manipulators, is proven through its experimental implementation on a Locobot mobile robot.**

***Index Terms*—Mobile manipulation, autonomous pick and place, ROS1**

I. Introduction

Robotic arms have been widely employed in assembly and manipulation tasks in many industrial applications. In recent years, there have been many developments of manipulators to safely work in environments shared with human operators, improving the performance of the overall system, as well as providing additional flexibility to satisfy the task requirements.

However, manipulators that are installed in a fixed place have limited working space and cannot provide the same flexibility as mobile agents. On the other hand, mobile manipulators can manipulate objects and deliver them to different locations, in a very useful fashion for various applications, e.g., for logistics, assistive assembly, automated order picking systems and ASRS (Automated Storage and Retrieval Systems), reducing time, cost and space in environments like warehouses [1].

The potentialities of the mobile manipulators can be fully exploited only through efficient solutions to the various problems intrinsically involved, such as path planning, safe navigation with collision avoidance and object recognition and grasping. Mobile manipulators path planning approaches can be divided into two groups: (i) the mobile platform and the manipulator are treated as separated subsystems and (ii) both the mobile base and the manipulator are considered as a single unit [2]. The former approach allows solving the path planning conveniently but obtaining sub-optimal results, while the latter provides better solutions but requires higher computational resources to deal with high DOF systems.

An Optimized Hierarchical Mobile Manipulator Planner (OHMP) is presented in [3]. The path planner decouples the motion planning of the mobile platform and the manipulator, and deploys the Probabilistic Roadmap (PRM) with a hybrid sampling approach to generate collision-free trajectories. However, the approach has only been validated in simulation environments. Furthermore, in [4], an approach based on Inverse Kinematics solutions is designed to compute collision-free trajectories in complex environments for mobile manipulators. In particular, it considers a reachability database and a query of poses that allow the robot to reach each desired grasping pose and joint angle.

In [5], a reactive motion combined with a holistic controller for a mobile manipulator allows the robot to execute visual-based grasping while maximizing manipulability. The algorithm considers the robot's mobile platform and the manipulator as a unique structure, and it can be implemented for both non-holonomic and holonomic platforms.

There are several techniques used to recognize an object to be grasped, in particular, fiducial markers are widely employed, since they provide useful information in an image. The authors in [6] compare the performance of several fiducial markers for pose estimation, e.g., ARTag, AprilTag, ArUco and STag. Among those markers, ARTags [7] showed the overall lowest computational cost, making them suitable for low-powered platforms.

This paper proposes a ROS1-based software architecture that allows the robot to (i) be able to receive commands and search for the requested items and (ii) pick and place them. The mobile manipulator is intended as able to navigate safely in a human-shared environment, thanks to a dynamic path planner, such as the one developed in [8], and briefly recalled in this paper. Thanks to the proposed architecture, a low-computational mobile manipulator working in indoor environments can autonomously find an item, requested by a human operator, and bring it to a desired depot location. Developed for a mobile manipulator with limited resources and not depending on predefined starting poses, the presented solution is lightweight and flexible. Note that we consider as low-computational or limited resource robot any platform that performs computations relying only on its CPU, i.e., it does not include a dedicated GPU for processing algorithms that require higher amounts of data, such as object detection.

The remainder of the paper is structured as follows: Section

II introduces some preliminaries related to the developed software architecture, while Section III unfolds the development steps of the proposed approach. Then, Section IV presents the experimental setup and illustrates the experimental testing outcomes. Finally, Section V draws some conclusions and open issues, as well as future works.

II. Preliminaries

The developed architecture and the adopted algorithms concepts are potentially applicable to different low-resources mobile manipulators. This paper illustrates their implementation on a Locobot mobile manipulator [9], starting from some initial findings presented in [10]; the code for such implementation is accessible online at [11]. Detailed information on the Locobot characteristics, as well as on the hardware setup and sensors, is available in Section IV.

The contribution of this paper will focus on the manipulator workflow for the pick-and-place of items requested by a human. To ease the description of the proposed framework, some preliminary information is provided hereafter.

A. Problem scenario

Mobile robot assistants are becoming increasingly popular to assist human workers with tasks such as picking, selecting, packaging, and shipping products. By using robots to handle and move products, the agility and quality of operations can be improved, while allowing humans to focus on more complex tasks or leading to collaborative solutions [12]. Moreover, using mobile manipulators for pick-and-place operations provides a highly flexible solution in plants and warehouses, for example to allow the human to command the robot to look for, pick and place specific products.

The target environment for a possible application of the framework is a warehouse, in which it is necessary to pick stored items and place them in a depot. This kind of environment includes static obstacles, such as shelves and walls, and dynamic obstacles with varying degrees of predictability, varying from other mobile manipulators to humans. The experimental tests reported in Section IV are relative to a laboratory demonstration, in which the problem is scaled down to a single room with some obstacles, humans, and some shelves where the products/items are stocked. The dimensions of the shelves and the considered items to be handled are compatible with the Locobot characteristics; in particular, the handled items are represented by cardboard packaging (small boxes), suitable for the gripper and compliant with the maximum payload.

B. Manipulator motion planners

Given the above problem scenario, our main objective during manipulation is to avoid collisions with the shelves when picking an item. Several planners and libraries have been tested to choose the most suitable ones. Among these, the probabilistic optimization framework STOMP (Stochastic Trajectory Optimization for Motion Planning) [13] proved to have good tunability and stable behaviour in scenes dense with obstacles. In particular, we tested the planner using linear interpolation as an initialization method for the optimization process, and a collision-related cost function, as the objective of our motion planning algorithm is to prevent colliding with obstacles. Another planner worth to be mentioned is the CHOMP (Covariant Hamiltonian Optimization for Motion Planning) algorithm [14], a gradient-based trajectory optimization method. It is highly tunable through parameters, but also subject to local minima issues leading to limitations in finding a feasible path in narrow passages, thus less preferable with respect to the STOMP planner. Nevertheless, to improve the above described methods performances, an optimized version of the Rapidly-exploring Random Tree algorithm, namely RRT* [15], can be used as a pre-processing algorithm.

C. Items identification and obstacle avoidance

For what concerns item recognition, the most recent and advanced solutions exploit deep learning. However, given we are considering a limited-resource platform, we preferred to use markers to identify items. To this aim, we decided to make use of ARTags, a fiducial marker system to support augmented reality. In particular, the ARTag maker system achieves a low rate of false positives and inter-marker confusion with minimal marker size, through the use of digital coding theory. It is designed to be robust against variations in lighting by utilizing an edge linking method [16]. In order to identify a specific item to be grasped, we took advantage of the possibility of generating unique ARTags to be recognized through a depth camera. This way, the object's pose can be retrieved and translated into a pose goal for the end-effector. The main drawback behind this approach is the limited scalability, due to the limited number of unique tags and the constraints associated with detection range and line-of-sight.

To accomplish obstacle avoidance, we exploited 3D occupancy grid planning and scene mapping techniques, in order to create a 3D model of an environment by estimating the occupancy of the space [17]. This technique involves dividing the 3D space into a grid and assigning a binary value to each cell, indicating whether it is occupied or free. The grid is typically constructed from sensor data obtained from cameras, LIDAR, or other sensors mounted on a robot.

III. Mobile manipulator framework description

The proposed framework has been developed using ROS1, employing a modular architecture so as to foster flexibility for the debugging process and functionalities enhancement. Furthermore, the framework proposed in this paper has the aim of demonstrating autonomous capabilities for low-computational mobile manipulators, such as the Locobot, for navigating in the environment and manipulating objects. In the considered scenario, the robot is required to pick and place specific objects in the working place. The ARTag markers associated to the objects to be manipulated provide relevant information to the robot, such as ID and location.

The developed software architecture is based on a decoupled planning of the entire task, from the object pickup request to the placement operation, by means of the communication manager, which commands the manipulator and the base individually through dedicated channels.

In particular, the structure is composed of three main nodes, as shown in Figure 1.

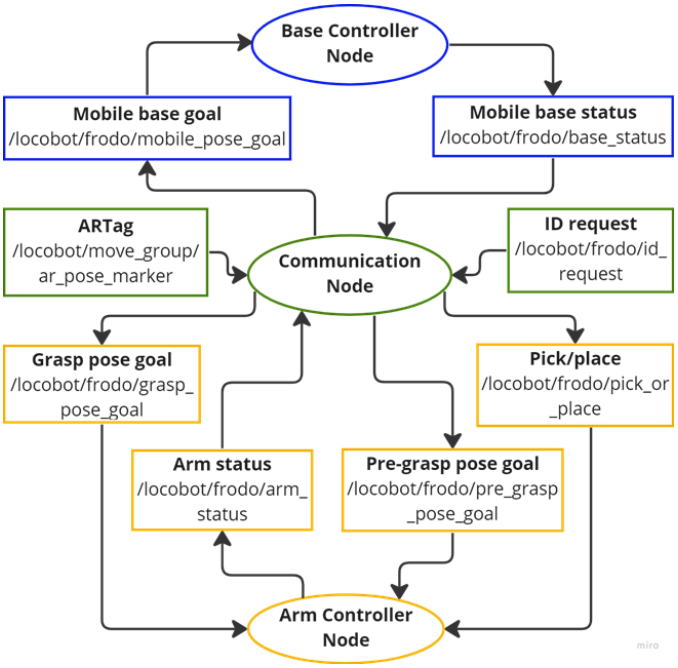

Fig. 1: Structure of the proposed framework using ROS1

- **Communication Node**: it manages the task requests and sends commands to control both the mobile platform and manipulator. In particular, it reads the IDs of the ARTags in the scene and compares them with the ID request: if both IDs match, the pick-and-place operation is enabled. Furthermore, it implements also a recovery behaviour in case of failure.
- **Base Controller Node**: it controls the motion of the mobile platform, and its working status (running, planning success or planning failure) is transmitted to the Communication Node.
- **Arm Controller Node**: it plans the motion of the manipulator to reach the desired pose and controls the opening or closure of the gripper. This node also communicates its working status to the Communication Node.

A. Communication Node

The Communication Node handles the item request, commands the robot to search for the requested item in the scene, and manages the pick-and-place operations once the required item is identified. In case of failure, it implements a mechanism to deal with unsuccessful attempts.

1) Item request handling: A task is started after receiving an item-picking request, which is submitted by a human operator through a dedicated interface (not introduced here). The item is identified with an ID and the robot scans the working place searching for an ARTag that contains such an ID. The ARTag contains also information about the object's pose, so the trajectory to reach the goal position for the mobile robot and the manipulator can be easily computed. If the desired ARTag is not within the robot's field of view, then it proceeds to the searching phase.

2) Searching phase: The searching phase is triggered when the robot does not successfully find the ARTag with the requested ID. Within this phase, the robot explores the working place searching actively for the required item. During the exploration phase, the robot repeats a few times some predefined functions until the item is found: it first rotates to change its field of view and, if the item is still not found, then it moves to another location.

Note that the robot performs the searching process in a different location only after completing a rotation of 360° with steps of 30° clockwise. This behaviour is depicted in Figure 2. The searching phase is interrupted as soon as the requested marker is found, and the robot is then ready to perform the next task.

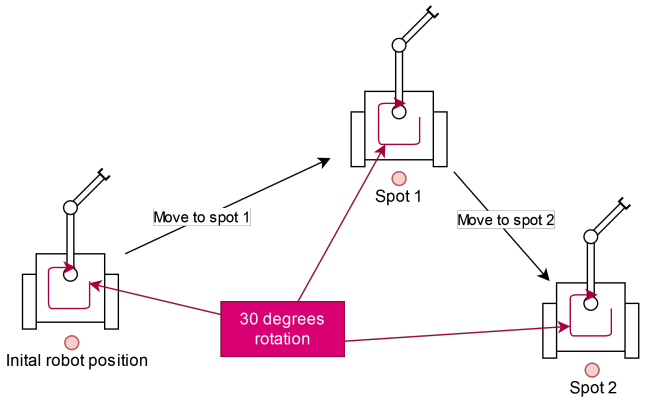

Fig. 2: Robot searching phase procedure to find he requested item

3) Pick-and-place routine: The pick-and-place routine is built taking into account the interaction between the mobile base and the arm through the Communication Node. Figure 3 illustrates the main steps for picking and placing an object.

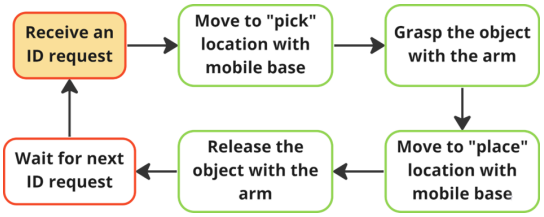

Fig. 3: Actions performed for picking and placing an object

It is worth underlining that a new task is executed only if the previous one has been completed; this is achieved by implementing the status topic, where the mobile base or robotic arm communicates its status: running, failing or succeeding. Furthermore, in order to improve the success rate of completing each task, the motion of the mobile platform and the robotic arm has been split into two sub-motions: the robot base is first moved in a neighbourhood of the object to be picked (or the depot for placing), and then the robotic arm is properly moved by assigning to the end-effector suitable poses for picking/placing.

4) Recovery behaviour: The Communication Node has been designed to deal with those situations in which the robotic arm or the mobile base does not successfully complete the assigned tasks. In any case, the mobile manipulator checks the feasibility of the planned plan before executing it, but if a failure occurs due to incorrect positioning (related to path planning or software issues), the recovery behaviour is triggered. It consists in repositioning the mobile platform or the robotic arm in case it could not complete its task. In particular, the recovery behaviour considers two cases:

- *Case 1 - robotic arm fails*: If the robotic arm is not able to reach the goal pose, it is possible to exploit the additional degree of freedom provided by the mobile base, so as to reposition the mobile manipulator. The mobile base reposition behaviour is similar to the one described in the search phase, in which the robot moves to another spot and rotates until it finds another feasible path to reach the goal.
  In this case, the arm will enter in the **fail mode**, so that the Communication Node can handle the repositioning request. First, the arm controller node sends to the Communication Node the status `FAIL` to inform that probably the position reached by the mobile base is not the desired one, so it is required to be relocated. To this aim, the mobile platform then returns to the `HOME` position and performs a further searching phase to re-estimate the correct marker position. This procedure can be done no more than twice, before considering the item unreachable and returning to the `HOME` position, ready to take another command. An example of an arm fail-handling procedure is presented in Figure 4.

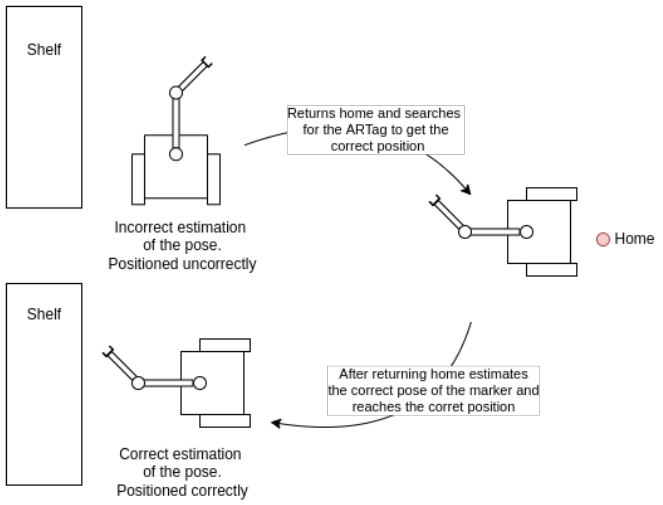

Fig. 4: Mobile base repositioning behaviour when the robotic arm fails to succeed in the current task.

- *Case 2 - mobile base fails*: The feasibility of the planned path to reach the destination is first checked by the global path planner provided by the ROS Navigation Stack. For instance, the destination goal corresponds to the pose of the ARTag marker that matches the requested ID. A failure occurs when the navigation system is not able to generate a valid path to reach the detected ARTag marker, e.g., due to sensor noise and map uncertainty. In other words, the estimated pose of the ARTag marker does not correspond to the one in the real environment. To overcome this issue, the Communication Node modifies the distance-to-goal of the marker, until a feasible path is found again. The repositioning procedure can be attempted twice, after which the system resets and waits for the next task.

B. Base Controller Node

This node publishes the status of the mobile base in order to be read by the Communication Node. The node status can take several values. Namely:

- `BASE TO GOAL`: this status lets the communication manager know that the base is moving to the goal and has yet to conclude its movement.
- `BASE GOAL OK`: this status signals the successful completion of the movement.
- `BASE GOAL FAIL`: it indicates that a re-planning phase is necessary.
- `BASE IDLE`: while sending a goal to the base, the communication manager checks for this flag. If the base is idle, then it can receive new goals; if not, the communication manager waits for it to become available.

Whenever the sent mobile goal is not associated to reach an item for picking or placing operations, this is recorded in a Boolean flag variable, as this information could be useful in future framework extensions.

C. Arm Controller Node

This node publishes the status of the robotic arm in order to be read by the Communication Node. This node status can take several values. Namely:

- `ARM FAIL`: it is implemented to allow the replanning of the mobile base, if the pick or place actions fail.
- `ARM SUCCESS`: this status is the one sent at the end of a pick/place action that is performed successfully. After 5 seconds, the status is changed back to `ARM IDLE`.
- `ARM IDLE`: this is a necessary step to make the communication node aware that the arm completed its motion. Without this status published, the base cannot move.
- `ARM RUNNING`: this message is exchanged only to make the user aware that the current node is running.

The **grasp pose** and the **pre-grasp pose** store the pose goals for the arm during the pick phase.

D. Description of the pick_or_place topic

The routine of movements to be performed changes if dealing with picking or placing. The sequence of actions to be executed is published on a specific topic, from which the arm can interpret two options: `PICK` or `PLACE`.
The routine is accordingly split into two parts, as it can be seen in Figure 5. The main difference between these two phases regards the update of the planning scene and the presence or not of some intermediate poses to approach the target position.

Once the **Arm controller** has received the goal pose and the action to be carried out, the routine starts and performs the desired moves taking into account the planning scene.

1) Pick routine: As sketched in Figure 5, the picking routine is the one that has the highest number of actions to be performed. The steps that take place, from the motion planning point of view, are:

a) Perform the motion to go in the **pre-grasp pose**.
b) Give the command to **Open** the gripper.
c) Move in the **grasp pose**.
d) Give the command to **Close** the gripper.
e) Perform the motion to reach the **retraction pose**.

The pre-grasping and retraction poses references frames are shown in Figure 6. At the same time, the planning scene must be updated so that the motion planner can take into account the presence of the item attached to the end-effector. To do so, we have to perform the following actions in sequence:

a) **Add** the item in the planning scene.
b) **Attach** the item to the gripper, so that the motion planner can consider its presence.

The pre-grasp, grasp, and retraction poses are computed as follows, starting from the knowledge of the position and the orientation of the ARTag and the end-effector with respect to the map frame. The roto-translational transformation between the ARTag frame and the map frame is then computed.

Let $\mathbf{T}^{map}_{ARTag} \in \mathbb{R}^{4x4}$ be the homogenous roto-translational matrix representing the ARTag frame in the map frame, defined as:

$$\mathbf{T}^{map}_{ARTag} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{R} \in \mathbb{R}^{3x3}$ is a rotational square matrix and $\mathbf{t} \in \mathbb{R}^{3x1}$ the translation vector. The pre-grasp position $\mathbf{pre\_grasp}_{ARTag} \in \mathbb{R}^{3x1}$ is defined in the ARTag frame as:

$$\mathbf{pre\_grasp}_{ARTag} = \begin{bmatrix} 0 & 0 & 0.1 \end{bmatrix}^T$$

(i.e., at a distance of 10 cm along the Z-axis of the ARTag frame), and then transformed in the map frame as:

$$\begin{bmatrix} \mathbf{pre\_grasp}_{map} \\ 1 \end{bmatrix} = \mathbf{T}^{map}_{ARTag} \begin{bmatrix} \mathbf{pre\_grasp}_{ARTag} \\ 1 \end{bmatrix} \quad (1)$$

The obtained vector $\mathbf{pre\_grasp}_{map}$ then represents the position that the end-effector has to reach before grasping the object.

The grasp pose is set in correspondence of the marker, with a suitable orientation for grasping.
The retraction pose is defined similarly to the pre-grasp one at 10 cm away along the X-axis and 10 cm higher to simulate the action normally a human would perform.

The vector of the retraction position $\mathbf{retraction}_{ARTag} \in \mathbb{R}^{3x1}$ is then defined in the ARTag frame as:

$$\mathbf{retraction}_{ARTag} = \begin{bmatrix} 0 & -0.1 & 0.1 \end{bmatrix}^T$$

and subsequently transformed in the map frame as:

$$\begin{bmatrix} \mathbf{retraction}_{map} \\ 1 \end{bmatrix} = \mathbf{T}^{map}_{ARTag} \begin{bmatrix} \mathbf{retraction}_{ARTag} \\ 1 \end{bmatrix} \quad (2)$$

2) Place routine: The picked object must be placed in the target location by correctly moving the robot base and then imposing the required sequence of actions to the manipulator. The **place routine** manages the entire motion, according to the following steps:

a) Perform the motion to go to the **target place pose**.
b) Give the command to **Open** the gripper.
c) Perform the motion to go in the **retraction pose** as depicted in Figure 7.
d) Give the command to **Close** the gripper.

Also, in this case, the planning scene has to be updated to consider the object's presence in the grasping hand. Since we added and attached the item in the previous routine, we have only to perform two actions:

a) **Detach** the object from the end-effector.
b) **Remove** the object from the planning scene.

Once completed the previous steps, a retraction position $\mathbf{retraction}_{ARTag} \in \mathbb{R}^{3x1}$ is defined similarly to the pre-grasp one. In particular, as depicted in Figure 7, such position is set at 10 cm away along the Z-axis of the ARTag frame, as:

$$\mathbf{retraction}_{ARTag} = \begin{bmatrix} 0 & 0 & 0.1 \end{bmatrix}^T$$

and then transformed in the map frame as:

$$\begin{bmatrix} \mathbf{retraction}_{map} \\ 1 \end{bmatrix} = \mathbf{T}^{map}_{ARTag} \begin{bmatrix} \mathbf{retraction}_{ARTag} \\ 1 \end{bmatrix} \quad (3)$$

IV. Experimental validation

The software architecture has been built using ROS1 Noetic, and tested in a laboratory environment. The Locobot WX250 mobile manipulator by Trossen Robotics, whose technical specifications are available in [9], has been used to test the proposed framework. It has a Kobuki mobile platform and a WidowX250 6-DOF manipulator (Figure 8). The mobile platform has differential wheels and active bumpers to sense if there is any contact with obstacles. Moreover, it is equipped with an RPLIDAR A2M8 (360° 2D LIDAR) and an Intel RealSense D435 (Stereo RGB-D), used for both manipulation and navigation tasks.

The mobile base implements the reactive strategy for human-obstacle avoidance and the physics-based technique for predicting the path, developed in [8]. In particular, a social navigation layer has been incorporated to avoid humans during path planning, using a Gaussian-based costmap approach. The standard ROS1 Navigation Stack has been utilized, with A* serving as the global planner and Timed-Elastic Band (TEB) [18] as the local planner. The TEB local planner complements the global planner A* by reducing the stair-like pattern in the global plan. With the focus on the pick-and-place manipulation steps, the RRT* method alone has been used to reduce latency and overall computational time. In fact, using a combination of RRT* and the STOMP algorithm (introduced in Section II) would have allowed for smoother paths, but at the expense of much longer computational times.

The ARTag recognition implementation exploits a node, provided by the wrapper ROS package for the Alvar ARTag

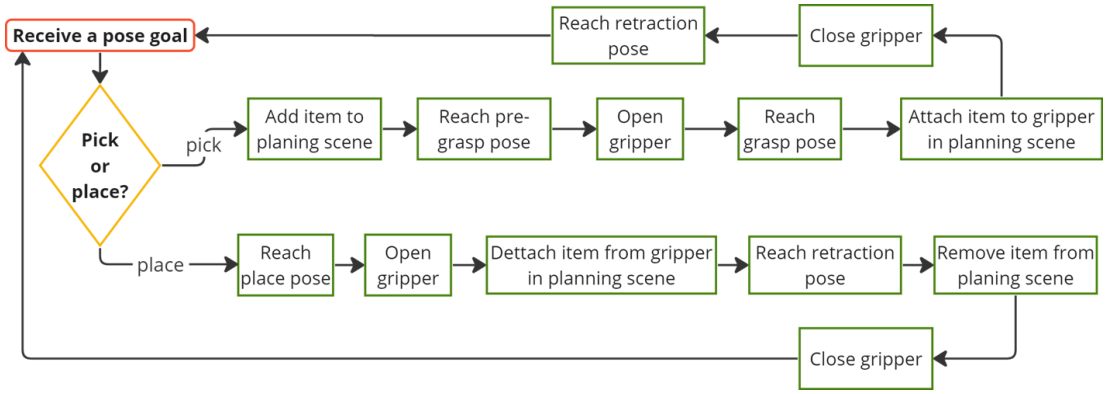

Fig. 5: Flowchart for understanding in which order the actions are performed.

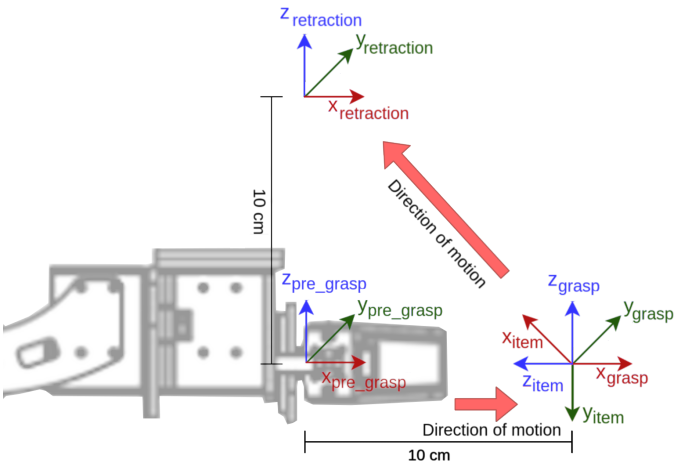

Fig. 6: Pre-grasp pose with respect to the item, 10 cm away along the Z-axis of the item reference frame. Definition of the retraction pose starting from the grasp pose, 10 cm away along the Z-axis and the X-axis of the item frame.

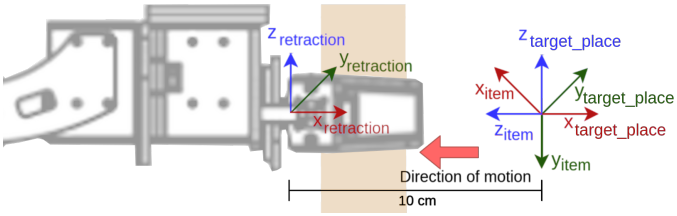

Fig. 7: Representation of reference frames during a placing action.

tracking library [19], which subscribes to the topics of the camera and computes the pose of the ARTag marker with respect to the camera frame. As the ARTag marker of interest is detected, a message containing its ID, pose and relative frame of reference is published on an ad-hoc topic.

For what concerns the 3D occupancy grid planning and scene mapping we adopted Octomap [17], a probabilistic 3D mapping framework that uses an octree data structure to represent occupancy information of an environment. The generated 3D occupancy grid map is capable of modelling the environment without prior information, constrained by the field of view of the camera. In our case, the data are sourced from a depth camera with a maximum range of 4 m and used for planning collision-free paths. Each time the frame is updated, the map is reconstructed according to a user-defined rate. In particular, for the simulation validation we have set an update rate of 10 Hz. For the real robot instead, in order to reduce the computational effort, we decided to impose some delays in the code and to have an update rate of 1 Hz.

The experimental validation has been performed both in simulation, within a Gazebo world that reconstructs the laboratory environment, and in the real-world laboratory setting (Figure 9). In particular, simulation was used to validate the framework algorithm workflow, while the execution on the real mobile manipulator allowed to test scenarios with different obstacles and objects positions.

A. *Testing in simulation*

Figure 10 shows the *rviz* representation of the framework validation in simulation. The tested working phases for the pick-and-place of an item of interest are described hereafter.

a) The robot receives the command, and approaches the identified item. Here only the mobile base is working.

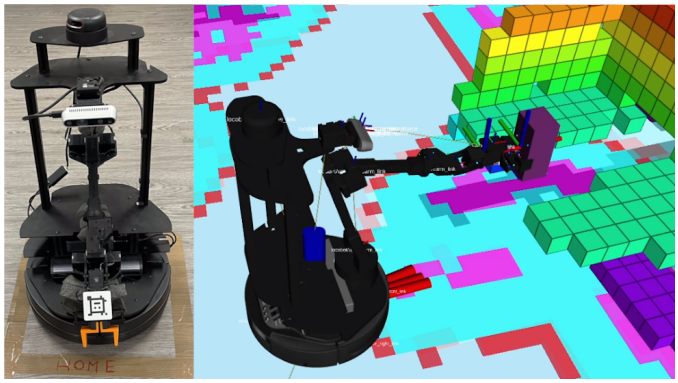

Fig. 8: The employed laboratory demonstrator (left) and its 3D visualization (right).

Fig. 9: The laboratory environment in Gazebo (left) and the real word shelves setup (right).

b) The base reaches the desired pose and the pre-grasp pose is taken. The item to be grasped is added to the planning scene as an object (green box).
c) As the manipulator is ready to pick the object, the gripper is opened, and the box added previously in the planning scene is attached to the end effector link, so that the robot is aware of its presence (the box turns purple).
d) Once the object is picked, the place pipeline is executed. After reaching the home pose, the mobile manipulator reaches the placing pose. Here the gripper is opened and the item is detached (the box returns green instead of purple).
e) The manipulator goes to the retraction pose and finishes its operations, eventually returning to its home pose ready for the next command.

B. Testing scenarios in the laboratory setup

In the experimental validation in the real world, not only the workflow is tested, but several shelf configurations have been considered to demonstrate the adaptability of the proposed framework. A brief demo video showcasing the experimental validation is available at [20]. In particular, first a general framework workflow is showcased (00:00 to 01:34). After a searching phase, then some focus is put on the pick and place phases (01:35 to 02:39). Another example of the framework validation is provided from 02:40 to 03:24. Finally, as can be seen at 03:25, if the item is already in the field of view of the robot, the searching phase is avoided, because not necessary.

Hence, the workflow of the proposed framework can be summarized as follows (Figure 11): (a) the mobile manipulator looks for the requested item, performing the searching phase if needed; (b) the mobile manipulator reaches the picking pose to pick the item and then heads to the placing pose; (c) the robot reaches the place pose and deposits the item, then eventually goes back to the home pose, ready for the next command (d).

V. Conclusions

A software architecture for autonomous mobile manipulation has been presented in this paper, with a particular focus on the case of low-computational mobile manipulators. Furthermore, a working open-source example is introduced, developed in ROS1 on a low-resource mobile manipulator and then validated in simulation and tested in a laboratory setup.

The proposed implementation allows to entrust to an autonomous mobile manipulator the searching and picking of a desired item, as requested by a human operator. This includes the exploration of an area to search for a specific item by scanning its surroundings.

The use of ARTag markers demonstrated to be a valid approach for low-computational powered platforms to estimate the object's pose to be grasped, making the overall system independent of the definition of predefined poses. Nevertheless, the use of ARTag markers limits the objects that can be manipulated, since the size of the marker is a fundamental factor to be considered, i.e., the smaller the marker, the less accurate the pose estimate.

Future works will involve the substitution of ARTag markers with an object recognition system using machine learning techniques, leveraging object affordance to improve the robot's manipulation and grasping. However, this improvement might require higher computational resources, for example, a dedicated GPU. This way, the overall system will be more flexible and adaptable to other application scenarios. A comparative analysis with similar architectures should also be considered in the future, to highlight pros and cons of the proposed one.

References

[1] L. Custodio and R. Machado, "Flexible automated warehouse: a literature review and an innovative framework," *The International Journal of Advanced Manufacturing Technology*, vol. 106, pp. 533–558, 2020.
[2] T. Sandakalum and M. H. Ang Jr, "Motion planning for mobile manipulators—a systematic review," *Machines*, vol. 10, no. 2, p. 97, 2022.
[3] Q. Li, Y. Mu, Y. You, Z. Zhang, and C. Feng, "A hierarchical motion planning for mobile manipulator," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 15, no. 9, pp. 1390–1399, 2020.
[4] J. Xu, K. Harada, W. Wan, T. Ueshiba, and Y. Domae, "Planning an efficient and robust base sequence for a mobile manipulator performing multiple pick-and-place tasks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 018–11 024.
[5] J. Haviland, N. Sünderhauf, and P. Corke, "A holistic approach to reactive mobile manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3122–3129, 2022.
[6] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers," *Journal of Intelligent & Robotic Systems*, vol. 101, pp. 1–26, 2021.
[7] M. Fiala, "Designing highly reliable fiducial markers," *IEEE Transactions on Pattern analysis and machine intelligence*, vol. 32, no. 7, pp. 1317–1324, 2009.
[8] P. D. Cen Cheng, M. Indri, F. Maresca, A. Ragazzo, and F. Sibona, "Dynamic path planning in human-shared environments for low-resource mobile agents," in *IEEE 32nd International Symposium on Industrial Electronics (ISIE 2023)*, 2023.
[9] T. Robotics, "LoCoBot WidowX-250 6-DOF," https://docs.trossenrobotics.com/interbotix_xslocobots_docs/specifications/locobot_wx250s.html, [Online; accessed April 2023].
[10] F. Maresca and A. Ragazzo, "ROS-based autonomous navigation and object recognition for a mobile manipulator operating in a warehouse environment," Master's thesis, Politecnico di Torino, 2022.
[11] "Github repository," https://github.com/AntoRag/thesis, [Online; accessed April 2023].
[12] A. Pasparakis, J. De Vries, and R. De Koster, "Assessing the impact of human–robot collaborative order picking systems on warehouse workers," *International Journal of Production Research*, pp. 1–15, 2023.
[13] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

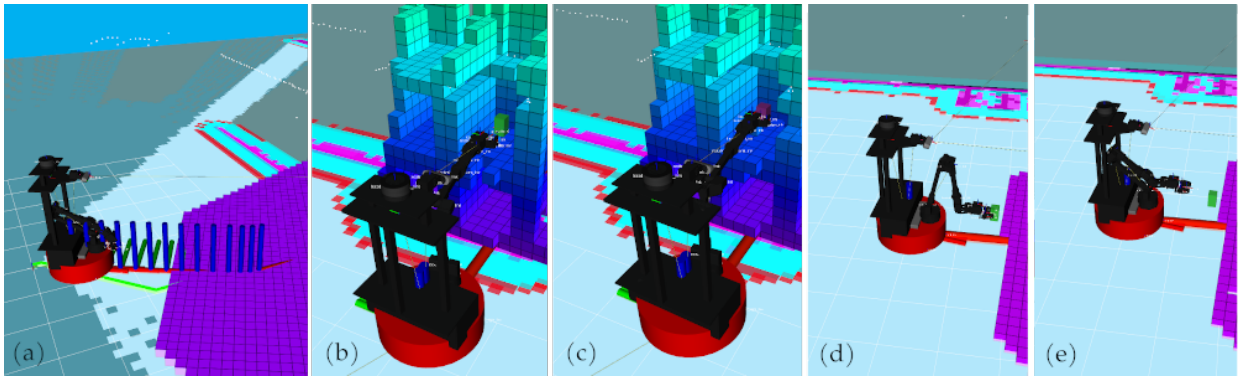

Fig. 10: Workflow validation in the Gazebo world setup.

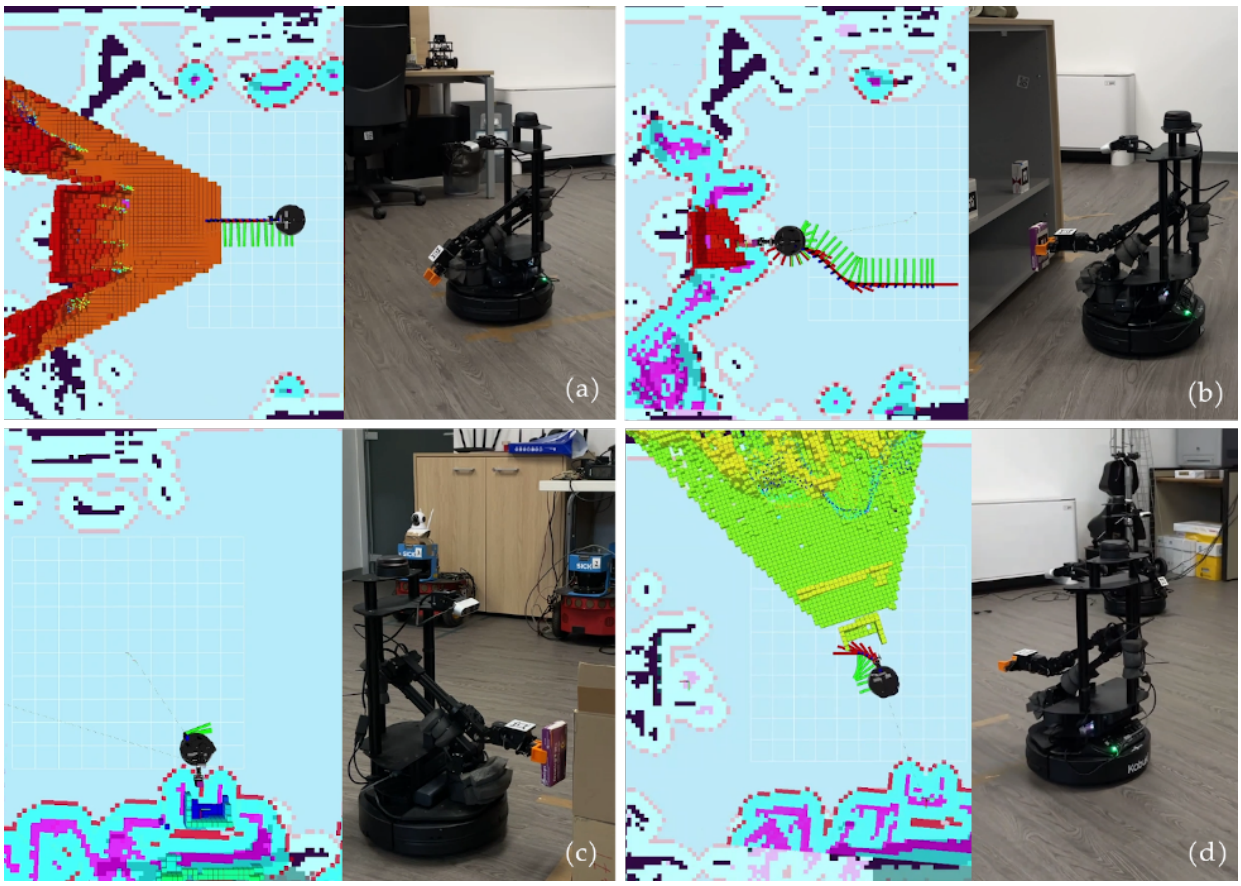

Fig. 11: Experimental validation sequence in the real-world setup and its relative *rviz* visualization.

[14] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[15] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[16] M. Fiala, "ARTag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, 2005, pp. 590–596 vol. 2.

[17] 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

[18] "ROS Timed-Elastic Band," http://wiki.ros.org/teb_local_planner, [Online; accessed April 2023].

[19] S. Niekum, "ROS wrapper for alvar, an open source ARTag tracking library," http://wiki.ros.org/ar_track_alvar, 2015, [Online; accessed April 2023].

[20] "Experimental test video demo," https://youtu.be/aAbBya1FV8o, [Online; accessed April 2023].