Citation: Miguel Burgh-Oliván, Rosario Aragüés, and Gonzalo López-Nicolás. Ros-based multirobot system for collaborative interaction. In Danilo Tardioli, Vicente Matellán, Guillermo Heredia, Manuel F. Silva, and Lino Marques, editors, ROBOT2022: Fifth Iberian Robotics Conference, pages 411–422, Cham, 2023. Springer International Publishing.

ROS-based multirobot system for collaborative interaction

Miguel Burgh-Oliván, Rosario Aragüés, and Gonzalo López-Nicolás

Instituto de Investigación en Ingeniería de Aragón, Universidad de Zaragoza, Spain,
raragues@unizar.es and gonlopez@unizar.es

Abstract. This paper presents the design and implementation of a collaborative multi-robot system based on ROS. The goal is to manipulate objects with multiple cobots simultaneously by following commands given by a user via gestures. Two methods have been designed, developed, implemented and experimentally evaluated: The first one is based on the ROS package called MoveIt! and focuses mainly on configuration to allow simultaneous control of different cobots. The second method involves the development of a third-party motion planner that sends commands directly to the controllers responsible for executing the cobots' movements. The Leap Motion, a device that can be used for gesture recognition, is also integrated into the system to enable user interaction in object manipulation. The system has been tested in simulation using Gazebo and evaluated in a real UR10 robot. The main contribution of the proposed architecture is that it solves the problem of controlling multiple robots simultaneously in ROS. In particular, our approach allows simultaneous execution of tasks with different types of controllers, brands and models, as well as direct control of the robots by using the Leap Motion device.

Keywords: Collaborative Robots, Multi-robot, ROS, Robot Operating System, MoveIt!, Leap Motion

1 Introduction

The use of collaborative robots (cobots) is already widespread in manufacturing lines. They allow to combine the strengths of humans and robots to complete a given task with effectiveness while reducing risks to human workers [4] [7] [13]. With the proliferation of cobots, unskilled workers may have great difficulty interacting with or controlling the robotic system properly. One solution to this problem lies in the development of human-robot communication that does not require prior technical knowledge of the robotic systems [13], as industrial robots are not designed to interact with humans and are typically programmed to perform predetermined repetitive tasks [7]. There is also growing interest in replacing a single industrial robot with a team of cooperatively working robots [8] [9].

The focus of this article is on a multi-robot system integrated with cobots, rather than a dual-arm cobot [7]. One of the difficulties is configuring the system architecture with multiple robots. Although we use ROS as it is one of the

most established tools in the robotics community for dealing with such systems, there is little documentation on how to set up and configure such multi-cobot systems, and the scarce documentation available lacks important information and explanations. In addition, the multi-robot approach is not widely used due to its complexity at the mechanical and system levels [7]. The cobots in this system can cooperate and collaborate to accomplish tasks that are not feasible with only one cobot, such as manipulating a deformable object that is too large, too heavy, or difficult to grasp [6]. There is also the need to control a group of cobots, even if they are from different companies, which requires the use of different types of controllers.

Since the initial release of ROS (2007), there has not been much publicly available documentation on how to develop a system to control multiple cobots simultaneously in ROS. There are two main issues with the current setup of ROS. The first one is the configuration of the controllers, because we require them to be able to control each of the cobots simultaneously (instead of sequentially) even as the number of cobots increases. The second problem is the motion planner [2], whose task is to calculate the motion that each of the joints should perform until each cobot reaches the desired position. In this process, *preempted*s notifications are generated on the system scheduler when the tasks are performed simultaneously by the package MoveIt!. This means that the problem of simulating multiple cobots on the computer must be addressed, leading to better use of resources and better coordination in defining the tasks. Another point worth highlighting is the need to develop a human-robot interface that allows workers to control the robots without prior technical knowledge, such as through voice commands, gestures, or body movements [1] [2] [5].

We propose the alternatives named ***MoveIt-1×M*** and ***Custom-N×M*** for building a multi-robot system with cobots based on modifying the Unified Robot Description Format (URDF) robot model [10] and using a motion planner to control the cobots independently. The robots can also be controlled through Leap Motion's human-robot interface [5] [11]. One of the solutions, called *Custom-N×M*, uses our customized motion planner that computes the motion and sends the commands directly to the controllers that are responsible for executing the motions of each cobot. The other solution, called *MoveIt-1×M*, uses the ROS package MoveIt! [3] and configures it to allow simultaneous control of different cobots separated by namespaces [10]. We also use Leap Motion as a solution that allows different movements to be performed based on hand motions and gestures, each of which can be tracked by multiple cobots.

The result of the developed solutions allows different brands of cobots to perform different tasks simultaneously with different types of controllers. For both solutions, we perform the pick and place task for one cobot, two cobots, and four cobots. In addition, for one and two cobots, we tested the human interaction using the Leap Motion device, and finally we tested the system in a real robot, the UR10 model from Universal Robots, for picking up and putting down objects. The complexity of controlling multiple cobots is manageable, as the concept is to replicate what is done for one cobot, making it easier to scale.

The remainder of this article is organized as follows. Section 2 explains the architecture of the multi-robot system, the requirements that must be met during design and development, it provides an analysis of the possible alternatives and the rationale for the chosen solutions, and it shows the integration of the Leap Motion device into the system and how it works. Section 3 shows the results obtained with both solutions on Gazebo and the real robot. Finally, section 4 concludes the paper and presents some future work. All the documentation and code to reproduce the approach described in this paper can be found on this website: https://github.com/Serru/MultiCobot-UR10-Gripper.

2 Design of the multi-robot system

In a conventional manufacturing process, workers typically performs repetitive manual tasks using primarily their hands, such as assembly, object handling, material processing, etc. The integration of collaborative robots (cobots) as a solution to significantly improve efficiency and reduce physical strain on workers performing tedious tasks is increasingly being used. A multi-robot system with collaborative robots enables the completion of a wider and more difficult variety of tasks than a single cobot solution.Our approach reduces the resources required to integrate a new cobot, enables awareness of the other robots, and facilitates tasks that require synchronization without the need for external devices such as the Kinect and RGB-D sensor to sense the work environment [2]. Another goal is to allow the human worker to interact naturally with two cobots simultaneously.

2.1 Architecture of the system

The architecture of the system is shown in Fig. 1. It must be scalable and open source. Therefore, we have chosen ROS, which allows the integration of new elements into the system by adding ROS nodes (locally or externally). Moreover, it is a framework that does not depend on a specific robot platform, but allows the integration of any robot brand, its robot models and sensors. The version of ROS used is *Kinetic Kame*, which runs on Ubuntu 16.04. For robot control and trajectory planning, we used the MoveIt! package which is mainly based on the *move_group* node and provides a variety of services. This node allows us to retrieve the current values of the cobot's joints and the position of the *end-effector*, choose between different planning algorithms, and perform the planning and execution of the planned movements. We considered other software based on message passing but they did not offer much support for robotics, were outdated or are proprietary.

To test the system, we see in Fig. 1 that there are three possibilities. To test one robot, we can use Rviz (a native tool of ROS), Gazebo (a simulator integrated in ROS) and finally directly the physical cobot. To properly test multiple cobots, we will only use Gazebo. We integrate a device that allows

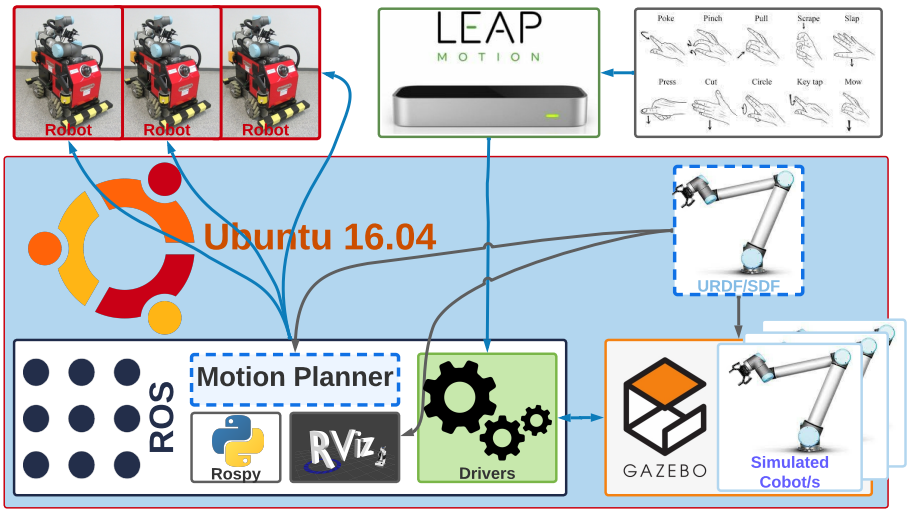


Fig. 1. General schema of the multirobot system with cobots

interaction between a user and the cobot, the Leap Motion[1]. There are several reasons for this choice, but mainly because of its low cost, robustness, accurate hand detection and positioning, and gesture interaction. Finally, the URDF file is part of the configuration and not part of the architecture. The role it plays in the system is critical, namely modeling the robot or robots we will control and test.

2.2 ROS-based multi-robot solutions

The proposed solutions are a combination of the *URDF file* describing the cobots and *Motion Planner*, as shown in Fig. 1. In this case, we use the MoveIt! package. However, it has a major drawback, since it is not designed for simultaneous control of multiple cobots. Solutions must allow simultaneous control of multiple cobots, with or without human intervention. This premise leads to the following five solutions (see Tab. 1). The table is split into three columns: *Proposed Solution*, *Motion Planner*, and *URDF File*. The *Proposed Solution* column shows the combinations we considered during the analysis, and we named them for clarity, e.g., *MoveIt-N×1* means (*Motion Planner*)-(*Num. Replications*)×(*Num. Cobots modeled*). The *Motion Planner* column is divided into two columns. The first one is labeled *From* and shows the name of the *Motion Planner* used. The second column is called *Number of Replications* and shows the number of replications of the *Motion Planner* used, ranging from *1 to N* replications. Finally,

[1]The Leap Motion Controller is an optical hand tracking module that captures the movements of your hands. Website: https://www.ultraleap.com/product/leap-motion-controller/

the *URDF File* column indicates the number of modeled robots described in it, ranging from *1 to M*.

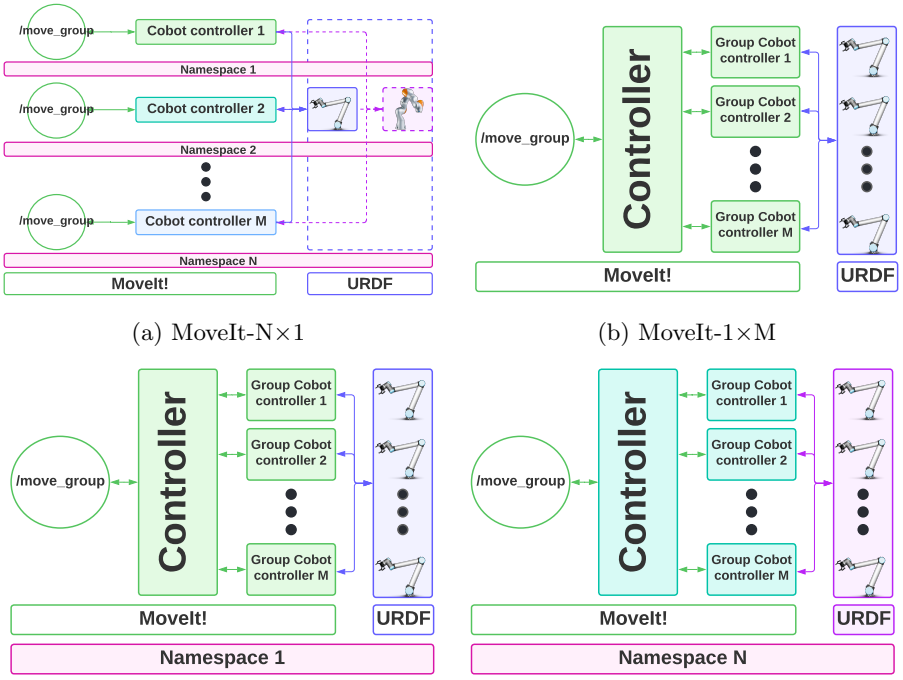
Table 1. Possible combinations between the motion planner MoveIt! and the robot modeling by the URDF file.

	Motion Planner		URDF File
Proposed Solution	From	Num. Replications	Num. Cobots modeled
Default	MoveIt!	1	1
MoveIt-N×1	**MoveIt!**	**N**	**1**
MoveIt-1×M	MoveIt!	1	M
MoveIt-N×M	MoveIt!	N	M
Custom-N×M	**Custom**	**N**	**M**

The first four options in Tab. 1 are the solutions we can get by combining the *motion planner* and the robot described by the *URDF file*, and the last one (*Custom-N×M*) is an alternative to the *MoveIt-N×M* solution. We can discard the *Default* solution since it is not a multi-robot system. We have chosen the solutions *MoveIt-N×1* and *Custom-N×M*, and will also explain in detail why we have not chosen the proposed solutions *MoveIt-1×M* and *MoveIt-N×M*.

This *MoveIt-N×1* solution from Tab. 1 is an extension of the Default solution based on replication of all nodes in the ROS package MoveIt!. Moreover, each replication is separated by a *namespace* that allows adding up to N cobots, as shown in Fig. 2a, which reduces the complexity of scaling the system. Each replication can use a different URDF File, i.e. you can also control cobots of different brands (UR, Kuka, etc.) and with different types of controllers (position, speed, etc.). The disadvantage of replication is the loss of free resources, since all services are active without doing productive work. On the other hand, this means that each robot is ready to use the full capabilities of MoveIt!. In addition, the *MoveIt Setup Assistant* feature allows a very fast setup.

Before we get to the *Custom-N×M* solution, we need to go through *MoveIt-1×M* and *MoveIt-N×M* (Fig. 2) to explain why these potential solutions were discarded. Let us start with the *MoveIt-1×M*, shown in Fig. 2b. There is a major difference here from *MoveIt-N×1* in Fig. 2a. Although *MoveIt-1×M* appears to be a definitive solution, it currently lacks adequate performance. This is due to the limitations of the architecture of *MoveIt!*, which prevents it from meeting the requirement of computing simultaneous motions. It is not easy to detect this limitation, but it is due to the fact that it can only compute the inverse kinematics of one of the cobots at a time, which means that it would get the inverse kinematics of each cobot in turns. This bottleneck becomes more apparent as the number of cobots in the system increases. At a certain point, it can be seen that the cobots pause until all computations for each of them are completed before executing the movement. If one of the cobots encounters a singularity while



(a) MoveIt-N×1

(b) MoveIt-1×M

(c) MoveIt-N×M

Fig. 2. Proposed solutions (*MoveIt-N×1*, *MoveIt-1×M* and *MoveIt-N×M*)

calculating its inverse kinematics, the entire system would stop. It is also limited to only one type of controller. To add more cobots to the system, a subgroup (*Group Cobot N controller*) must be added to control one of the cobots, which also imposes the constraint that it must be the same controller type as the main group (*Controller*). In this solution, the complexity of controlling the cobots increases with the number of cobots in the system. This is because *MoveIt!* controls the cobots by setting the pose of each cobot in advance rather than at runtime, which means that the target of the pose cannot be changed spontaneously. Because of this property, it is not possible to control the movements of multiple cobots interactively via an external device controlled by a human. This *MoveIt-1×M* proposal has already been implemented by TEAM O2AC [12].

For the above reasons, we discard the solution *MoveIt-1×M* as an option and discuss the next solution *MoveIt-N×M* shown in Fig. 2c. This one is similar to *MoveIt-N×1* (Fig. 2a), where the system is scaled by replicating MoveIt! via *namespaces*, while cobots are added via the URDF file, which is the main feature of *MoveIt-1×M* (Fig. 2b). When only one cobot is defined in the URDF file, we get the solution *MoveIt-N×1*, but the moment multiple cobots are added in the URDF file, this solution inherits the drawbacks of the solution *MoveIt-*

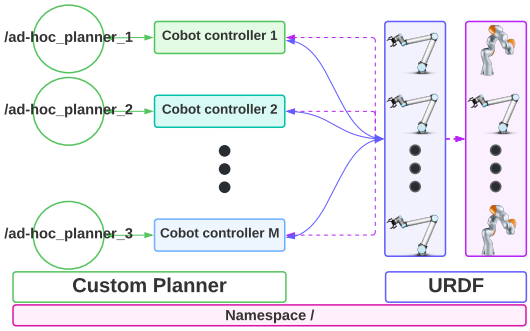


Fig. 3. Proposed solution *Custom-N×M*

1×M (Fig. 2b), which is suitable for automating repetitive tasks but not for interacting with the user.

At this point, the solutions *MoveIt-1×M* and *MoveIt-N×M* shown in Fig. 2 are possible solutions, but they do not meet the requirements needed by our system. As a consequence, the *Custom-N×M* solution is proposed, which implements a custom planner from scratch based on the functionality of MoveIt!. In this solution, the mentioned problems for the proposed solutions *MoveIt-1×M* and *MoveIt-N×M* do not occur. As can be seen in Fig. 3, each cobot has its own motion planner and controller. This solves the problem that the inverse kinematics can only be computed for one cobot at a time, it also solves being able to only use one type of controller, and the problem of interacting with users that occurred with *MoveIt-1×M* and *MoveIt-N×M*. These problems are eliminated by adding custom motion planners and a separate controller type for each cobot that can operate under the same *namespace*. At the same time, the ability to add more cobots to the system via the URDF File is preserved. In addition, this solution consumes fewer resources than *MoveIt-N×1*, as no services are left unused when scaling.

2.3 Integration of human machine interface with Leap Motion

Fig. 4 shows the integration of Leap Motion into the system for simultaneously control of up to two robots, for the solutions in Fig. 1. Basically, the ROS node, which controls the movements of the cobot, only needs to subscribe to the topic *leapmotion/data1* or *leapmotion/data2* and use this data input accordingly to control the cobot's movements. Notice that we have two topics. The reason is that Leap Motion can recognize up to two hands, so the *right hand* (right.msg) and the *left hand* (left.msg) data are separated into different topics.

Fig. 4 shows the design of the integration with nodes and topics, showing a representation of the general behavior and its components. Here, the Leap Motion library is used to identify the gestures and obtain the data needed to control the gripper, the cobot's movements, and the end-effector's orientations.

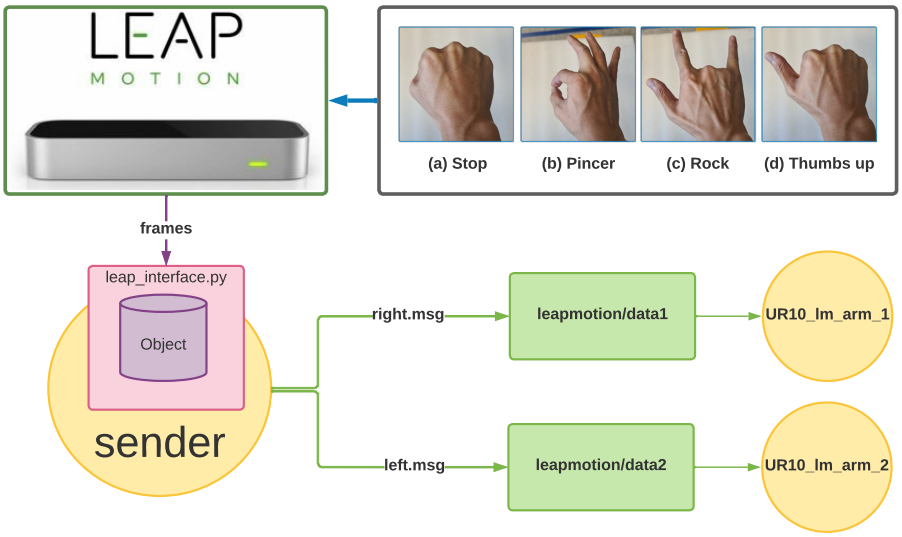


Fig. 4. Leap Motion's integration scheme

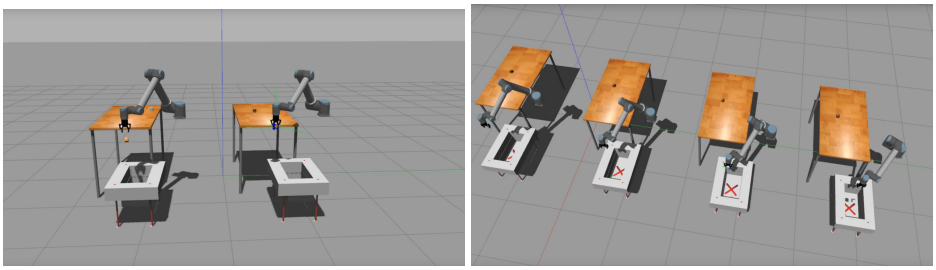
The Leap Motion library allows waiting for events (frames from Leap Motion) and retrieves the required data at each event and stores it in an object, which is then accessed by the *sender node* using the API provided by Leap Motion. The *sender node* receives the information, stores it in a message and publishes it using the *topic leapmotion/data1* for the right hand data. The *ROS node* is subscribed to this topic and sends the commands to the cobot with the information received from the *topic leapmotion/data1*. Finally, the *ROS node* executes the motion. Four types of gestures have been implemented using the API provided by Leap Motion (see Fig. 4). The *fist gesture* indicates to stop sending instructions, the *pincer gesture* is applied to control the cobot's gripper, the *thumbs up gesture* indicates that it is ready, and the *rock gesture* indicates that it takes the current position of the hand as a reference frame.

3 Experimental validation

3.1 Results with Gazebo

Several tests were performed in the Gazebo simulator, where each of the cobots picked up three cubes placed on a table and placed them in a basket. The same test was executed for the two proposed solutions *MoveIt-N×1* and *Custom-N×M*. The first test (Fig. 5) is a simulation with two cobots, showing how both cobots simultaneously pick up each of the cubes with the gripper and place them in a basket. As an example of scaling the solution for N cobots, the same test was run with four cobots in the system. The setup environment is a *case of use* where objects and cobots are mapped in the *world* to get a better overview of the

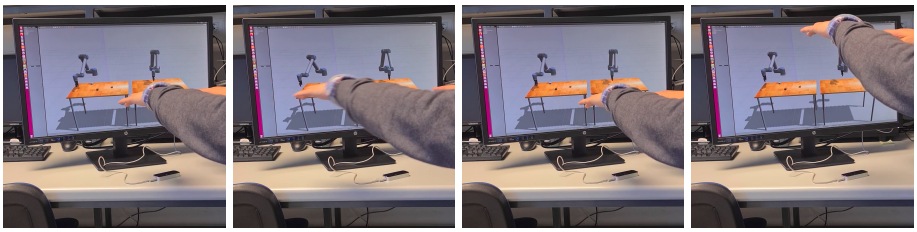
simultaneous movements of each cobot; alternatively, it would also be possible to create a simple test where, e.g., robots building a tower together.



(a) Pick & place by two cobots (b) Pick & place by four cobots

Fig. 5. Results of the pick & place on Gazebo with *MoveIt-N×1* & *Custom-N×M* solutions (see the testing in the video[3]).

Finally, Fig. 6 shows the tests performed with the Leap Motion device lying on the table with the hands directly above the device which is connected to the computer. This allows the exchange of data provided by a user's hands movements which are represented on Gazebo by the cobots. This test shows that both solutions (*MoveIt-N×1* and *Custom-N×M*) can control the cobots correctly. They reproduce the movements performed by the user with both hands, and the interaction is quite smooth, as there is hardly any delay between the movement of the hand and the execution of the movement in Gazebo.



(a) Side movements (b) Up-and-down movements

Fig. 6. Results of performing hand movements obtained from Leap Motion on Gazebo with *MoveIt-N×1* and *Custom-N×M* solutions (see the full movements in the video[3]).

In terms of robustness, note that in the hand-tracking recognition test, it may happen that the system misinterprets the detected gesture with a similar one. In addition, it is important to set the frame rate appropriately; otherwise the gestures may not be recognized within a reasonable time frame. Furthermore, there is a clear difference between *MoveIt-N×1* and *Custom-N×M*, but both simulations perform the movements successfully. In the video[3] with the

simulation results, it can be seen that *Custom-N×M* follows the movements of the hand better.

3.2 Results with the UR10 cobot

The tests are performed with the physical UR10 cobot shown in Fig. 7. The solution tested with the cobot is the *Custom-N×M* modified to control a single robot. Before running the tests, the simulations were evaluated in Gazebo to verify that everything was working correctly. During these tests, the maximum speed at which the robot moves was limited for safety reasons. The results obtained in Fig. 7 shows the user instructing the robot, via the Leap Motion device, to pick up and put down different objects: a sole (first row), a plastic bottle (second row), and a teddy bear (third row).

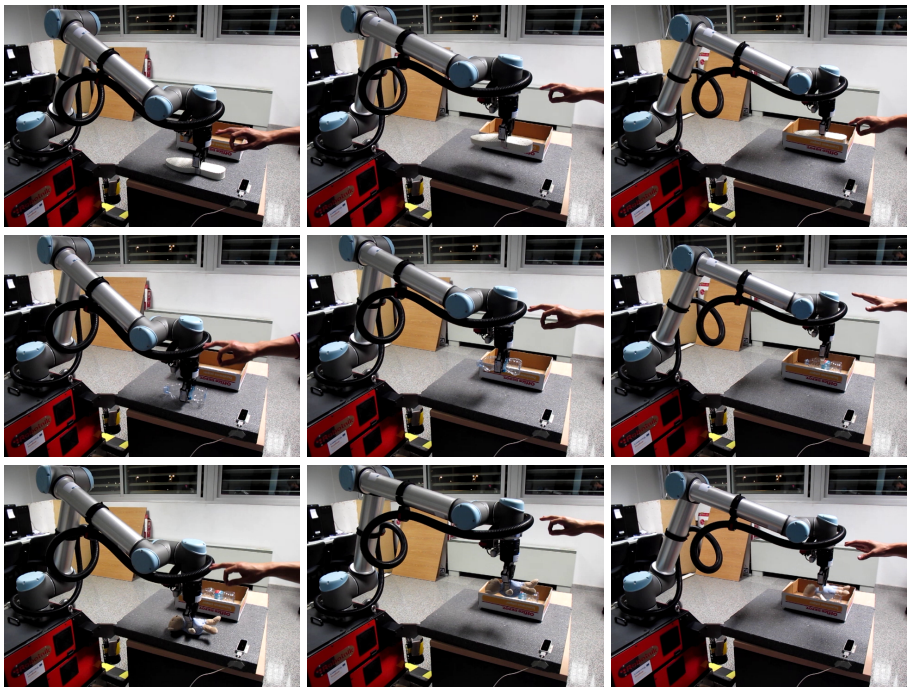


Fig. 7. Results of picking up and placing objects with Leap Motion on a robotic platform. Picking up and placing a sole (first row), a plastic bottle (second row), and a teddy bear (third row). Watch the tests in the video[3].

[3]Video link: https://github.com/Serru/MultiCobot-UR10-Gripper

4 Conclusion

This article addresses the problem of developing a multi-robot system in the ROS Kinetic environment, enabling the manipulation of objects with several cobots. For this purpose, several possible solutions were analyzed that allow the manipulation of objects and the manual control of up to two cobots through the Leap Motion device by recognizing gestures and movements of the hands. It was found difficult to properly configure the ROS package MoveIt! since it is not documented and is not able to natively support multiple cobots. Our proposed solutions (*MoveIt-N×1* and *Custom-N×M*) can scale up to N cobots that can be controlled simultaneously. In addition, the system allows independent manual control of up to two cobots by one person via the Leap Motion device. The operation of the system was validated with tests in Gazebo for one, two and four cobots in both solutions and, finally, in a real environment, in this case with a mobile manipulator to which a UR10 robot is attached. Based on the results obtained, improvements can be made to the system, such as adding artificial intelligence to predict future positions of the cobot. Also of interest is the addition of sensors to detect and avoid collisions with other cobots or humans, which would increase the safety of the system. Note that ROS 2 is currently growing and offering new features for multi-robot systems after its official launch in 2017. Nevertheless, due to its current predominance, ROS is still necessary for the correct work of many applications, and the contributions proposed in the field of multi-robot systems will continue to be of interest and use, even if new alternatives such as ROS 2 appear.

Acknowledgment

This work was supported by the Spanish Government/European Union through projects PGC2018-098719-B-I00 and PID2021-124137OB-I00 funded by MCIN/AEI/10.13039/501100011033.

References

1. Ahmed, S., Popov, V., Topalov, A., Shakev, N.: Hand gesture based concept of human - mobile robot interaction with leap motion sensor. IFAC-PapersOnLine **52**(25), 321–326 (2019). 19th IFAC Conference on Technology, Culture and International Stability TECIS 2019
2. Brito, T., Lima, J., Costa, P., Matellán, V., Braun, J.: Collision avoidance system with obstacles and humans to collaborative robots arms based on RGB-D data. In: M.F. Silva, J. Luís Lima, L.P. Reis, A. Sanfeliu, D. Tardioli (eds.) Robot 2019: Fourth Iberian Robotics Conference, pp. 331–342. Springer International Publishing, Cham (2020)
3. Chitta, S.: MoveIt!: an introduction. In: Robot Operating System (ROS): The Complete Reference (Volume 1), pp. 3–27. Springer International Publishing, Cham (2016)

4. Fischer, H., Vulliez, P., Gazeau, J.P., Zeghloul, S.: An industrial standard based control architecture for multi-robot real time coordination. In: IEEE 14th International Conference on Industrial Informatics (INDIN), pp. 207–212 (2016)
5. Gutierrez, A., Guda, V.K., Mugisha, S., Chevallereau, C., Chablat, D.: Trajectory planning in dynamics environment: Application for haptic perception in safe human-robot interaction. In: V.G. Duffy (ed.) Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Health, Operations Management, and Design, pp. 313–328. Springer International Publishing, Cham (2022)
6. Herguedas, R., López-Nicolás, G., Aragüés, R., Sagüés, C.: Survey on multi-robot manipulation of deformable objects. In: IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 977–984 (2019)
7. Kruse, D., Wen, J.T., Radke, R.J.: A sensor-based dual-arm tele-robotic system. IEEE Transactions on Automation Science and Engineering **12**(1), 4–18 (2015)
8. López-Nicolás, G., Aranda, M., Sagüés, C.: Multi-robot formations: one homography to rule them all, pp. 703–714. Springer International Publishing, Cham (2014)
9. Ma, Z., Zhu, L., Wang, P., Zhao, Y.: ROS-based multi-robot system simulator. In: 2019 Chinese Automation Congress (CAC), pp. 4228–4232 (2019)
10. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: ROS: an open-source robot operating system. In: ICRA workshop on open source software, vol. 3, p. 5. Kobe, Japan (2009)
11. Strazdas, D., Hintz, J., Khalifa, A., Abdelrahman, A.A., Hempel, T., Al-Hamadi, A.: Robot system assistant (RoSA): Towards intuitive multi-modal and multi-device human-robot interaction. Sensors **22**(3) (2022)
12. Von Drigalski, F., et al.: Team O2AS at the world robot summit 2018: An approach to robotic kitting and assembly tasks using general purpose grippers and tools. Advanced Robotics **34**(7-8), 514–530 (2020)
13. Čorňák, M., Tölgyessy, M., Hubinský, P.: Innovative collaborative method for interaction between a human operator and robotic manipulator using pointing gestures. Applied Sciences **12**(1), 258 (2022)