Development of a ROS Driver and Support Stack for the KMR iiwa Mobile Manipulator*

Hatem Fakhruldeen, David Marquez-Gamez, and Andrew I Cooper

Leverhulme Research Centre for Functional Materials Design, University of Liverpool
{h.fakhruldeen, dmarquez, aicooper}@liverpool.ac.uk

Abstract. Mobile manipulators are expected to revolutionise robotics applications because they combine mobility and dexterity. Robotics middlewares such as ROS (Robot Operating System) is a key component to develop the capability of these platforms and to research their novel applications. In this paper, we present a complete ROS stack for the KMR iiwa mobile manipulator. This stack comprises of a ROS driver, with a novel architecture, running natively on the platform controller and the essential support packages that allow motion planning, navigation, visualisation and simulation using ROS standard tools and frameworks. To our knowledge, this work is the first *ROS 1*[1] package for the KMR iiwa. To demonstrate the capabilities of our work, we present example applications both in simulation and using the real robot. Finally, the proposed stack is used in a heterogeneous multi-robot system in the context of an autonomous chemistry laboratory.

1 Introduction

Mobile manipulators are becoming more common and are expected to revolutionise various sectors using robotics [12]. A mobile manipulator comprises of a robotic manipulator mounted on a mobile base, which provides both mobility and dexterity. In the coming years, it is expected for these platforms will be widely employed in different indoor and outdoor applications such as providing assistance in manufacturing [9], transporting goods in warehouses [3], performing chemistry workflows [2] and picking fruits in farms [7], among others.

The expansion in the use and development of these platforms in recent years has been accelerated by the growth of e-commerce and its aim to have fully automated warehouses run by robots [1]. This trend is reflected in the increasing number of commercially available mobile manipulators such as KUKA KMR iiwa, MiR ER-FLEX, OMRON MoMa, to name but a few. Furthermore, these platforms are still widely studied in academic research labs as their potential application domains and full capabilities are still being investigated.

The Robot Operating System (ROS) is an open source robotics middleware framework that allows to connect various hardware and software components

* We acknowledge the Leverhulme Trust via the Leverhulme Research Centre for Functional Materials Design for funding.

[1] For the purpose of this work the term ROS will refer to *ROS 1*

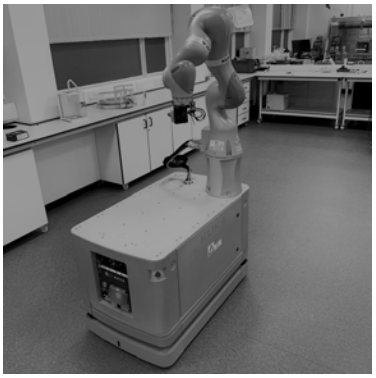

Fig. 1. KMR iiwa mobile manipulator.

together [11]. This allows for the easy addition and integration of new peripherals with robots such as sensors and grippers. Also, it provides a number of tools and frameworks for robot's control, motion planning, navigation and simulation, that also allow to develop new algorithms and test them on real and simulated robots. For these reasons, ROS has become a very popular and widely used as middleware in robotics research.

ROS support is crucial to advance research with mobile manipulators and to develop novel applications. Specifically, this includes having a ROS-based driver for the platform and supporting ROS packages that utilise that driver for planning and controlling the robot. Some mobile manipulators have ROS compatibility features out of the box by virtue of having ROS-based support for their individual components, *i.e.*, the mobile base and mounted manipulator. In that case, additional development and integration work is needed to fully use these robots with ROS. On the other hand, some mobile manipulators lack of ROS support because one or both of their components (the robotic manipulator or the mobile base) do not have developed drivers and supporting packages.

The KMR iiwa mobile manipulator is one such platform that lacks ROS support and compatibility. It was developed by the German company KUKA (Keller und Knappich Augsburg) as a collaborative robot (Cobot) for the purpose of meeting the requirements of Industry 4.0. It is a highly flexible and mobile platform that is intended for handling manufacturing automation tasks. This mobile manipulator is shown in Fig. 1. It comprises of the KMP200 omniMove mobile base and a LBR iiwa14 R820 robotic arm. The robot is programmed using KUKA's proprietary Sunrise.OS that utilises the Java programming language. Currently, there is no official KUKA support to provide ROS packages for the KMR iiwa mobile manipulator, the work presented in [3] was one of the first attempts to produce a ROS compatible software to control the first generation of the KUKA mobile platform, however that was explored to a limited extent and the produced source code was not made public.

The installed manipulator, LBR iiwa14, has a ROS-based driver and accompanying support packages that allow to control and interact with the arm [5]. The KMP200 mobile base does not have any public ROS-based driver at the moment. On the other hand, there exists a *ROS 2* (the second generation of ROS)[2] driver for the KMR iiwa platform that allows to control the mobile base and receive its sensors' data [4]. However, *ROS 2* nodes and topics are not compatible with *ROS 1*. There exists a special *ROS 1 - ROS 2* bridge[3] that can facilitate their interaction but is of limited use because it only offers restricted functionality and is cumbersome to set up and use. Furthermore, *ROS 1* is predominately used in research and few robots and peripherals currently offer *ROS 2* support, which limits the applicability of the driver in [4] in the context of heterogeneous robotic applications and use with other peripherals.

In this paper, we present a novel complete ROS stack for the KMR iiwa mobile manipulator that allows the control of its arm and mobile base, and the gathering of sensory data. This stack is composed of a ROS driver running natively on the robot controller and accompanying support packages that collectively allow to perform motion planning, navigation, visualisation and simulation using ROS standard tools. To the best of our knowledge this is the first public ROS package for the KMR iiwa mobile manipulator that provides an interface and complete integration between the platform, and the Robot Operating System (ROS). Furthermore, this paper details our contribution of the novel design of the driver architecture and describes the various tests performed to validate the stack operation with the robot. This will allow other robotic researchers to utilise this platform with ROS in their research and development endeavours. Our aim is to use this ROS-based mobile manipulator in the context of an autonomous chemistry laboratory and develop hardware and software architectures for chemist robots (robots that research chemistry); but, we envisage that there will be other real-world applications that would also benefit from this architecture.

This paper is structured as follows. A description of the KMR iiwa platform is presented in section 2. Section 3 describes the developed software stack and its components. In section 4, the results and tests performed to validate the software architecture are detailed. Finally, section 5 summarises the key conclusions.

2 KMR iiwa Robot

2.1 Platform Description

The KMR iiwa mobile manipulator (see Fig. 1) comprises of a LBR iiwa14 R820 robotic manipulator, a KMP200 omniMove mobile base and two SICK S300 safety laser scanners. The LBR iiwa14 arm is a 7-DOF (Degrees Of Freedom) robotic manipulator with 14 Kg payload. It is designed to be used in human robot collaboration settings.

[2] https://docs.ros.org/en/galactic/index.html
[3] https://github.com/ros2/ros1_bridge

The KMP200 is an omnidirectional mobile base that has four mecanum wheels. This base is equipped with two SICK S300 safety laser scanners mounted diagonally opposite to each other. These scanners emit laser beams at a height of 15 cm above the floor. Each scanner covers an area spanning 270° and thus covers one long and one short side of the base. The platform controller, named The Sunrise cabinet, along with the drive battery, are located inside the KMP200 base.

2.2 Operation and Safety

KUKA Sunrise.OS is the current operating system software for the KMR iiwa mobile manipulator. It is used by all KUKA robots that are controlled by the Sunrise cabinet. It provides tools for the development, deployment and configuration of robotic applications. Moreover, these applications are developed in the Sunrise Workbench programming environment using Java programming language and Sunrise.OS software packages. These collectively represent Sunrise.OS API (Application Programming Interface) that allows to interact and control the robot's components. This API allows to command and access the robot resources locally via the Sunrise cabinet or the platform's teach pendant, named SmartPAD. Consequently, in contrary to using a middleware such as ROS, this makes the system harder to integrate in heterogeneous robotic experiments and interface with other systems.

The platform has three operating modes: T1 mode, in which the platform is manually operated in reduced speed mode for the purposes of testing and debugging the developed application, T2 mode that is the same as T1 without the speed reduction and AUT mode where the platform executes its program autonomously.

The operation safety of this mobile platform is monitored by means of the SICK S300 laser scanners. Each scanner monitors a predefined area around the mobile base, which is divided into warning and protective fields. The size of these fields depends on the vehicle's velocity, where higher velocities translates to larger fields. Any violation of the protective fields result in an emergency stop in all modes except for T1 and T2 modes when the velocity is less than 0.13 m/s. Violations of the warning fields depends on the mode but mostly result in maximum speed reduction. Moreover, the installed robotic arm, by virtue of it being a Cobot, has built-in force/torque sensors in the joints that ensure safe and compliant robotic manipulation.

2.3 Interfacing with ROS

To interface the Sunrise.OS with ROS middleware and consequently make the mobile platform compatible with it, two approaches in the literature were discussed. One approach to achieve that, suggested by [5], was to run ROS nodes natively on the Sunrise cabinet as part of the robotic application. This was achieved by using ROSJava[4] libraries, which provided a complete native Java

[4] https://wiki.ros.org/rosjava

implementation of the ROS framework. This allowed these nodes to directly access the robot's data and commands on the controller, and at the same time to interact with the outside "ROS ecosystem" without restrictions. This approach required the installation of third-party libraries on the robot controller to run RosJava nodes.

Another approach suggested by [10] and [4], which did not require any external libraries, was to have a *daemon thread* running on the Sunrise cabinet that allowed access to the robot data and commands over TCP/IP or UDP sockets. This thread would be exchanging string messages with an external ROS node that would parse inbound and outbound messages and exchange them accordingly with the ROS ecosystem. However, this approach, when compared to the first one, is less versatile and robust because: i) raw string messages are prone to parsing errors and difficult to handle when transmitting complicated data, ii) programming with low level sockets instead of established frameworks leads to reliability issues and corner-case errors and iii) having the ROS node running on a separate machine increases the delay of interaction with the robot and adds an extra point of failure to the system. For these reasons, we have opted to utilise the first approach when developing our proposed ROS driver and use it as as a starting point for our architecture.

3 KMR iiwa ROS Stack

Robotic applications developed with ROS are usually composed of multiple nodes that are running on different machines/robots communicating and interacting with each other in order to execute their tasks. This is only possible because ROS provides the communication middleware. As a result, in this framework, everything is a node that exposes certain interfaces and expects other nodes' interface in return in order to operate.

We utilised this philosophy in our work when designing the ROS driver for the KMR iiwa robot. As a result, the developed stack is composed of: i) a robot driver that creates a ROS node running natively on the robot controller, which exposes various standard ROS interfaces, and ii) a number of accompanying support ROS packages that utilise these interfaces to control the robot and get its sensors data, such as MoveIt[5], Navigation stack[6], RViz[7], Gazebo [8]. The following subsections describe the software design, the developed support ROS packages of the stack and their relationship with the robot safety system.

The source code of the work presented in this paper, the ROS driver and the related support packages for the KMR iiwa mobile platform are available in the following links:

- KMR iiwa ROS driver: https://github.com/stoic-roboticist/kmriiwa_ros_java
- Support packages: https://github.com/stoic-roboticist/kmriiwa_ros_stack

[5] https://moveit.ros.org
[6] https://wiki.ros.org/navigation
[7] https://wiki.ros.org/rviz
[8] http://gazebosim.org

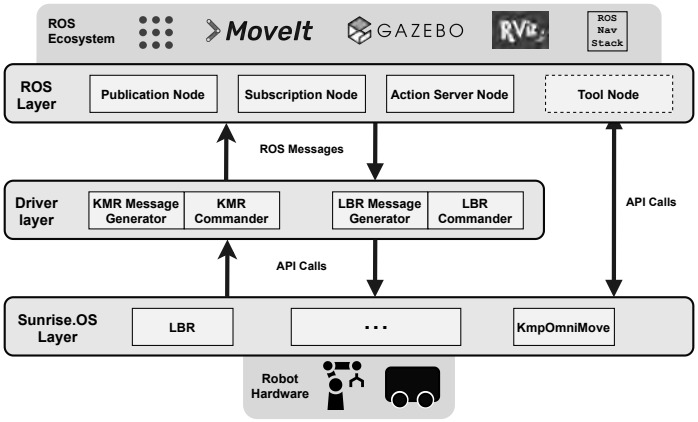

Fig. 2. KMR iiwa ROS driver architecture.

3.1 KMR iiwa Driver Design

Architecture: In order to interact with the robot and its controller, the KUKA's Sunrise.OS API need to be wrapped using the Java programming language. Moreover, to fully leverage ROS middleware capabilities and not develop custom communication solutions like in [4,10], the software need to run ROS libraries natively on the controller as described in [5]. For these reasons, the driver was developed as a robotic application running on Sunrise.OS that utilised RosJava libraries. The developed software architecture and interactions is illustrated in Fig. 2. The driver is composed of three principal layers:

- **ROS Layer**: Is an abstract layer that represents the driver's interface with the outside ROS ecosystem, which includes ROS core and all the other nodes. This layer is composed of three main nodes and one optional node:
  - **Publication Node**: that allows the driver to publish messages to different ROS topics.
  - **Subscription Node**: that allows the driver to receive messages from different ROS topics.
  - **Action Server Node**: that provides the driver with an action server that can interact with external action clients.
  - **Tool Node**: is an optional node that is provided such that it can be utilised to interact with any tools attached to the robot, such as grippers, that expose their functionality via Sunrise.OS API. In order to utilise this node, the user needs to provide the implementation of this node's methods. Table 1 lists the different topics exposed by this layer's components.
- **Driver Layer**: Is a communication layer that represents the core layer of the driver that transforms inbound ROS messages to robot's actions and at the same time transforms robot's sensors readings to outbound ROS messages.

This layer interacts with the ROS layer via generated and received ROS messages; while on the other end, it interacts with the Sunrise.OS layer via its Sunrise.OS API calls, which translate to actuating the robot and reading its sensors' data. This layer is composed of four components:

- **KMR Message Generator**: creates ROS messages from the readings of the base's laser scanners, odometry, emergency status and battery information. Note that, the Sunrise.OS API calls used to retrieve the sensors' data were based on those described in [4] since they were not readily available in the documentation.
- **KMR Commander**: uses the appropriate Sunrise.OS API calls to issue velocity commands that moves/jogs the KMP200 mobile base from the received ROS *Twist* messages.
- **LBR Message Generator**: creates ROS messages from the LBR iiwa14 arm's joints' states, emergency and calibration status information.
- **LBR Commander**: utilises the appropriate Sunrise.OS API calls from the received ROS messages to issue joint position commands to actuate the LBR iiwa14 arm. It also creates motion trajectory commands for the arm to follow from the received *follow joint trajectory* action goals.

– **Sunrise.OS Layer**: Is a physical layer that represents all the native Sunrise.OS classes such as *LBR* for the robot arm and *KmpOmniMove* for the base that are utilised to call its API and consequently interact with the robot system.

Table 1. The different topics exposed by the ROS layer components.

Topic Name	Message Type	Description
Publication Node Topics		
arm/joint_states	sensor_msgs.JointState	LBR iiwa14 joint states
arm/state/RobotStatus	kmriiwa_msgs.LBRStatus	LBR iiwa14 general status
base/state/LaserB1Scan	sensor_msgs.LaserScan	Front SICK S300 laser readings
base/state/LaserB4Scan	sensor_msgs.LaserScan	Back SICK S300 laser readings
base/state/odom	nav_msgs.Odometry	KMP200 odometry readings
Subscription Node Topics		
/arm/command/JointPosition	kmriiwa_msgs.JointPosition	LBR iiwa14 joint position motion target
/base/command/cmd_vel	geometry_msgs.Twist	KMP200 velocity twist jog target
Action Server Node Actions		
/arm/manipulator_controller	control_msgs.FollowJointTrajectoryAction	LBR iiwa14 joint trajectory controller

Operation: The driver is running on the controller as a native Sunrise.OS robotic application. This application is composed of two running threads. A main thread that synchronously polls the *Subscription Node* and *Action Server Node* for any newly received ROS messages or action goals and subsequently execute them using the appropriate driver layer commander. The second thread is a publication thread, that utilises the ROS messages generated from the driver layer and publish them using *Publication Node* at a constant rate.

3.2 Support ROS Packages

The ROS ecosystem provides a standard set of tools/frameworks for robot motion planning, navigation, visualisation and simulation. These respectively include Moveit,

ROS navigation stack, Rviz and Gazebo. The developed stack provides fully configured and ready to use packages to run these tools with the KMR iiwa mobile manipulator. These packages utilise the developed driver and its exposed interfaces to interact with the robot. All these packages are name-spaced by default, which make them readily usable in multi-robot applications. A short description of these packages follows:

- **KMR iiwa Description Package**: contains the robot's description and its URDF models.
- **KMR iiwa MoveIt Package**: allows Moveit to control the robot's arm by utilising the *FollowJointTrajectory* action server running on the robot to execute the planned trajectories. This is achieved by using the interfaces provided by *LBR Commander* and *Action Server Node*. In the current implementation, ros_control, which is a generic controller interface package, is not supported because we do not currently have access to the official KUKA software packages that allow low level control of the arm.
- **KMR iiwa Navigation Package**: allows to control the robot's base and navigate it in a known or unknown environments using SLAM[6], localisation and path planning algorithms. To do so, it relies on the interfaces provided by *KMR Commander* and *KMR Message Generator*.
- **KMR iiwa Visualisation Package**: contain different Rviz configurations to visualise the robot in different contexts.
- **KMR iiwa Gazebo Package**: allows to simulate the robot in Gazebo. It exposes the same interfaces as the robot driver and allows to control both the robot's base and the arm.

3.3 Robot Safety

The KMR iiwa safety system is managed by a PLC (Programmable Logic Controller) that monitors safety signals received from the robot's various devices, *i.e.*, laser scanners, mobile base and arm, and based on their values and its programmed logic determines the safety state of robot operation. These signals and their combinations can be configured using a special safety configuration file, which is part of every Sunrise.OS project running on the robot controller. Consequently, every application running on the controller will be subjected to these settings and would stop executing if any safety rules are violated.

The developed driver, being a native Sunrise.OS application, is also subject to these safety rules and would stop executing if they are violated. As a result, using our developed stack does not violate the robot safety rules and rely on them for safe operation. Furthermore, all the stack components were tested with the default safety settings of the robot with no modifications.

4 System Testing and Applications

The proposed ROS-based driver is used in two example applications. This is not intended to be a particular challenging application or example, it is simply used to take the reader through the functionalities and capabilities of the developed software presented in this paper.

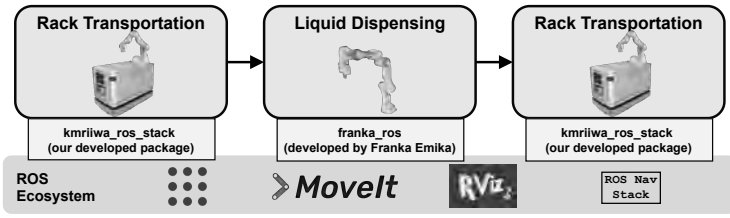

Fig. 3. The demonstrated sample preparation workflow. Note that ROS is used as a communication and execution layer for the robots.

4.1 Manipulation, Navigation and Simulation with ROS

A series of demonstrations was carried out to illustrate the stack's individual components' readiness for operation and showcase their utility. These demonstrations included: i) controlling the LBR iiwa14 arm using MoveIt, ii) controlling the KMP200 base using ROS navigation stack and iii) simulating and controlling the platform in Gazebo simulation environment. The demonstrations for MoveIt and navigation stack, which involved the real robot, were carried out by running the developed driver alongside the relevant individual package; where the user commanded the platform via RViz to different target poses. For the Gazebo demonstration, two KMR iiwa robots were simulated and then commanded using a script that utilised their available interfaces. A result video of these demonstrations can be viewed on the following link: https://youtu.be/ODOPMoMAK-o

4.2 Example Robotic Application in a Chemistry Laboratory

An example robotic application in the context of an autonomous chemistry laboratory was implemented. Specifically, a sample preparation workflow, this task is very common in manual and automated chemistry workflows [2]. This example application, besides illustrating the stack's navigation and manipulation capabilities, also demonstrates an heterogeneous system where our package is able to interact with other robots and interfacing with external sensors operating in the ROS ecosystem.

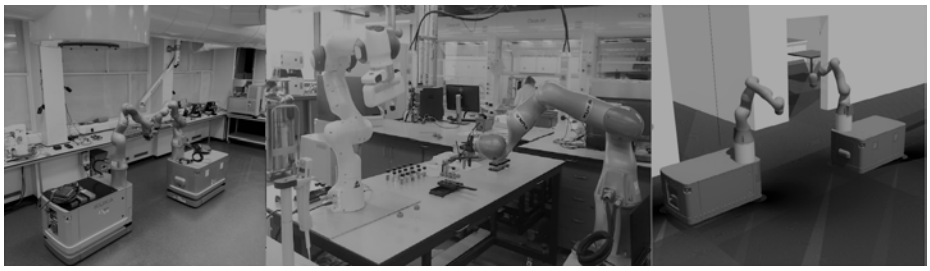

Fig. 4. (Left): Two KMR iiwa robots interacting with each other. (Centre): KMR iiwa placing the rack of vials in order for the Frank Emika robot to dispense liquid. (Right): two KMR iiwa robots interacting in Gazebo simulation environment.

The sample preparation workflow example can be described as follow: (1) the KMR iiwa navigates autonomously to a rack station. In the rack station, the robot use the information provided by a camera (Intel® Realsense™ D435) positioned on the mobile base to align the robot with the station and arrive at the correct manipulation position. The visual information use an AprilTag fiducial marker [8] to calculate the pose correction, compensate the navigation and localisation errors and correct the final position based on the sensor reading at the target rack-station-position. (2) the robot picks the rack of vials from the station and place it on top of its base. (3) the platform navigates to the liquid dispensing station, as in the first step, the robot use the the camera to align the mobile base before placing the rack on the liquid station table. (4) a Franka Emika Panda robot dispenses liquids by operating a pipette. (5) after finishing the liquid dispensing process, the KMR iiwa robot picks the rack of vials and transports it back to the rack station thus completing the task. Fig. 3 shows the described example and the interaction between the components of the system with our developed ROS-based package. In this task, all the manipulation and navigation operations were handled by Moveit and the ROS navigation stack, respectively. Moreover, ROS topics were used to coordinate the KMR iiwa and the Franka Emika Panda robot work as well as to publish the attached camera pose correction information. The task was repeated eight times. In all of the runs, the developed stack performed reliably with no performance issues. There were two failed runs due to the camera failing to detect the AprilTag and the robot navigating too close to an obstacle due to the path planner choosing a sub-optimal path that triggered an emergency stop. Fig. 4 shows screenshots of the example applications presented in this section and the package running in simulation and real robots as described in the previous section. A video result of the robotic application in a chemistry laboratory example can be viewed in the following link: https://youtu.be/psyFOOgRlyE

5 Conclusion

Mobile manipulators and their application are becoming more common, new platforms are being released and novel application domains are being explored. Providing ROS compatibility to these platforms will allow research and development of novel applications. In this paper, we presented a fully developed ROS stack for the KMR iiwa mobile manipulator. This stack is composed of a ROS driver running natively on the controller and a number of essential accompanying ROS packages. The driver has a novel architecture with three layers that interact with each other to allow robot control and compatibility with ROS. The stack was tested in an real-world robotic application in the context of an autonomous chemistry laboratory where the use of heterogeneous hardware platforms: robotics arms, mobile manipulators and standard lab equipment (*e.g.*, pipette) was presented. Future work includes the integration of the Sunrise.OS safety configuration into ROS planning frameworks as constraints.

References

1. Bogue, R.: Growth in e-commerce boosts innovation in the warehouse robot market. Industrial Robot: An International Journal **43**, 583–587 (Oct 2016)
2. Burger, B., Maffettone, P.M., Gusev, V.V., Aitchison, C.M., Bai, Y., Wang, X., Li, X., Alston, B.M., Li, B., Clowes, R., Rankin, N., Harris, B., Sprick, R.S., Cooper, A.I.: A mobile robotic chemist. Nature **583**(7815), 237–241 (Jul 2020)

3. Dömel, A., Kriegel, S., Kaßecker, M., Brucker, M., Bodenmüller, T., Suppa, M.: Toward fully autonomous mobile manipulation for industrial environments. International Journal of Advanced Robotic Systems **14**(4) (Jul 2017)
4. Heggem, C., Wahl, N.M., Tingelstad, L.: Configuration and Control of KMR iiwa Mobile Robots using ROS2. In: 2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS). pp. 1–6 (Jun 2020)
5. Hennersperger, C., Fuerst, B., Virga, S., Zettinig, O., Frisch, B., Neff, T., Navab, N.: Towards MRI-Based Autonomous Robotic US Acquisitions: A First Feasibility Study. IEEE Transactions on Medical Imaging **36**(2), 538–548 (Feb 2017)
6. Jaulin, L.: Range-only slam with occupancy maps: A set-membership approach. IEEE Transactions on Robotics **27**(5), 1004–1010 (2011)
7. Johan From, P., Grimstad, L., Hanheide, M., Pearson, S., Cielniak, G.: RASberry - Robotic and Autonomous Systems for Berry Production. Mechanical Engineering **140**(06), S14–S18 (Jun 2018)
8. Krogius, M., Haggenmiller, A., Olson, E.: Flexible layouts for fiducial tags. In: International Conference on Intelligent Robots and Systems (October 2019)
9. Meng, J., Wang, S., Li, G., Jiang, L., Zhang, X., Liu, C., Xie, Y.: Iterative-learning error compensation for autonomous parking of mobile manipulator in harsh industrial environment. Robotics and Computer-Integrated Manufacturing (Apr 2021)
10. Mokaram, S., Aitken, J.M., Martinez-Hernandez, U., Eimontaite, I., Cameron, D., Rolph, J., Gwilt, I., McAree, O., Law, J.: A ROS-integrated API for the KUKA LBR iiwa collaborative robot. IFAC-PapersOnLine **50**(1), 15859–15864 (Jul 2017)
11. 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)
12. Štibinger, P., Broughton, G., Majer, F., Rozsypálek, Z., Wang, A., Jindal, K., Zhou, A., Thakur, D., Loianno, G., Krajník, T., Saska, M.: Mobile Manipulator for Autonomous Localization, Grasping and Precise Placement of Construction Material in a Semi-Structured Environment. Robotics and Automation Letters **6**, 2595–2602 (Apr 2021)