Integrating mobile robot navigation control, visual object detection and manipulation using ROS and V-REP

Felipe Pierre Conter[1], João Alberto Fabro[1] and André Schneider de Oliveira[2]

Abstract— **The integration of robotic manipulators and mobile robots is a challenging task involving control of both the mobile base and the manipulator in an coordinated way. In order to study this coordination, in this paper a robotics simulation environment is used. Combining ROS (Robot Operating System) and a realistic robotics simulator, V-REP (Virtual Experimentation Platform), a KUKA youBot robot was used to integrate mobile navigation and object manipulation. The omni-directional mobile base is controlled using a fuzzy position controller, the detection of an object of interest over a table is performed using a simulated camera (with a blob detection algorithm), and then the manipulator arm is controlled for object grasping. The use of the simulator allows for a first step in the development of a complete ROS solution to manipulation/pick-and-place, and its use in industrial/commercial/residential environments.**

I. INTRODUCTION

Robotics projects can be hard to develop in the real world without prior testing and simulation. There are several simulation environments available, and each with its own characteristics. V-REP (Virtual Robotics Experimentation Platform) [1] is an example, providing a variety of examples and predefined robot models that can be used for 3D simulation and study without the need to build the real scenery with real robots. This kind of tool is essential to help in the development of various approaches within research groups that don't have access to real test environments, providing a way for testing different robots without having to physically acquire them.

Object manipulation is a common task in different robotics applications, such as those presented by the RoboCup@Home league [2]. It involves object detection, position estimation, grasp planning and execution, and different techniques can be used in each of these steps, including computer vision and artificial intelligence. Previous research indicates that participating teams use different approaches and solutions for this problems [3]. Some of them are not easy to learn and put into practice. This multidisciplinary challenge can become highly complex and intimidating for new researchers in robotics. Therefore, approaches with lower complexity can encourage the entering of new researchers in the field.

[1]J. A. Fabro is a professor with the Graduate Program on Applied Computing - PPGCA, at the Informatics Department - DAINF - at UTFPR - Federal University of Technology, Parana, Brazil - Campus Curitiba. F. P. Conter is a master student within this Graduate Program. `fabro@utfpr.edu.br, felipeconter.cc@gmail.com`

[2]A. S. de Oliveira is a professor with CPGEI - Graduate Program of Electrical Engineering and Industrial Informatics, UTFPR, Brazil - Campus Curitiba. `andreoliveira@utfpr.edu.br`

Recently, an approach based on the direct manipulation of objects by a mobile robot with a coupled manipulator arm [4] was presented. The main problem related was the precise estimation of the object position, that was solved by placing a QR-Code marker on the object. This solution has several disadvantages: it is necessary to place markers on every object to be manipulated, and the marker must be visible from the robot's point of view. In order to try to avoid such problems, this project proposes an alternative approach: using a camera positioned directly over the end-effector of the manipulator arm, and using an arm that can approach the object directly from above. With the camera directed "down", it would be possible to recognize the object by its shape/color, thus avoiding the necessity for object markers.

By using ROS (Robot Operating System) [5], an open-source, meta-operating system for robotics, it would become possible to increase code reuse when using the same simulated approach with a real robot. ROS provides services for integration and interoperability of processing nodes, as well as tools and libraries for coding. It offers a communication infrastructure that standardizes the exchange of commands and data between different components. The connection between ROS nodes and V-REP is possible because V-REP has a plugin that enables subscription and publishing to ROS topics.

Inside V-REP, several robot models are available for use in its simulation environment. The one selected for this work is a robot with an open platform and an integrated manipulation arm, already available in V-REP: youBot [6] [7]. This is a robot manufactured by automation company KUKA, designed for use in education and research. The main objective of this work is to apply fuzzy control for the navigation of the simulated youBot, approaching a table where there is an object of interest to be manipulated (in this case, a white cube with 5cm of edge size). The robot searches for the object's exact position using the camera coupled to the tip of the manipulator (positioned directly above the end-effector), using blob detection algorithms for image processing. Since the height of both the table and object is known, by finding the center of the object from above, it becomes possible to pick it up with the manipulator arm. Experiments are presented regarding every aspect of this situation: fuzzy control for robot navigation allowing approach to the table, inverse and direct kinematics to control the manipulator arm, image processing to estimate the position of the object, and the actual manipulation (picking it up and moving it to the robot's base).

The remaining of this paper is organized as follows:

Section II presents the methodology, section III details the experiments (regarding the fuzzy control of the robot's base towards the table for manipulation, object detection and the manipulator control in order to pick the object over the table). Section IV presents an evaluation of the results of the simulated experiments, and Section V presents some conclusions and suggestions of future work.

II. METHODOLOGY

The scene in V-REP includes a 5x5 meters floor, with a youBot robot that was modified by adding a laserscan Hokuyo URG-04LX-UG01 sensor [8] and V-REP vision sensor (camera) called "Blob detection camera" (available in the model browser, inside components, sensors). The Hokuyo sensor was placed on the front side of the youBot base, providing information that can be used for the navigation. The camera was placed on the tip of the arm, just before the grippers, as can be seen on Figure 1. The "Blob detection camera" is a container component, including the camera and a image processing filter that constantly search for areas based on a given threshold. In the proposed approach, the object with a predefined color and known shape is positioned in the pick-up area, and the blob-detector filter allows for the precise localization of this object in the manipulation area.

The problem set in V-REP is the following: youBot has to move to a predefined location, next to a table, with a predefined orientation (positioned towards the table). The table has a white cube in top of it. Once arriving to its destination, the robot has to position its camera on top of the table and detect the cube, using the blob detection camera filters to segment the object. Once the cube position is known, the last step is to grasp it with the arm and move it to a flat space located on top of the robot's base.

Robot movement and positioning is managed by a ROS node responsible for the robot control, written in C++ running outside V-REP. It makes use of the FLIE fuzzy library [10] for fuzzy control, based on the proposal of [9]. In V-REP, the youBot robot model has a script written in Lua language, that is responsible for the logic directly associated with the robot's low level calculations. It was adapted to communicate with ROS.

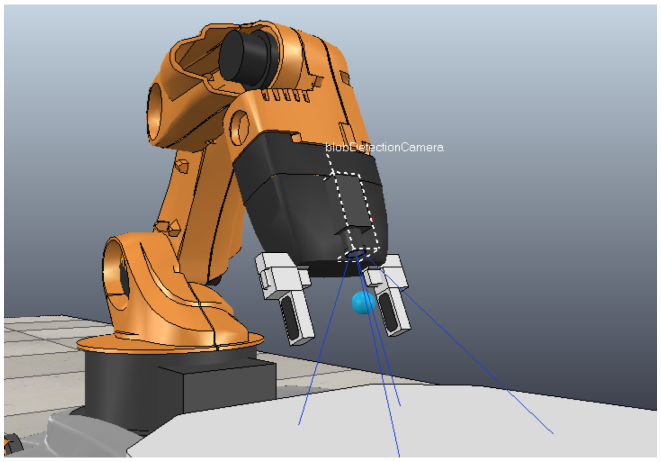

Fig. 1. Blob detection camera position on youBot arm

III. EXPERIMENTS

The problem has been divided into 4 steps: fuzzy control for youBot navigation, arm positioning control, camera positioning for object detection and object manipulation.

A. *Fuzzy Control youBot Mobile Base*

The first part of the experiment consists in making the robot move to the desired position (in front of the table), with the desired orientation (rotated, so it's facing the table). The movement control was reduced to two variables: angular velocity and linear velocity. Although the robot is capable of omni-directional movement, only one dimension of linear motion (forward/backward) was used. In this way, the code is less coupled to this robot in particular and can be easily adapted to non-omnidirectional robots.

The V-REP youBot model has a script that handles its movement and positioning. The robot has 4 wheels, and the script supports 3 types of velocities: forward/backward velocity (x), left/right velocity for sideways movement (y) and clockwise/counter-clockwise rotation velocity (z). The x, y and z velocities are then combined in a single value that is distributed to each wheel (Δ). The front left wheel velocity is obtained by (1). Equations (2), (3) and (4) represent the rear left wheel, rear right wheel and front right wheel velocities, respectively. Since we want the code to be compatible with non-omnidirectional robots, the only velocities ROS controls are the angular and linear velocities. For this reason, the left/right velocity (y) is always set to zero.

$$\Delta_{frontleft} = -x - y - z \tag{1}$$

$$\Delta_{rearleft} = -x + y - z \tag{2}$$

$$\Delta_{rearright} = -x - y + z \tag{3}$$

$$\Delta_{frontright} = -x + y + z \tag{4}$$

V-REP publishes the robot reference point odometry in a ROS topic (/odom). That topic is read by the ROS node responsible for the robot control. Knowing the desired position (a predefined goal in front of the table), the angular and linear errors are calculated. The linear error is the distance between the robot reference point and the goal, and the angular error is the difference between the robot orientation angle and the desired orientation, which is pointing towards the goal. To generate the angular and linear velocities, two fuzzy controllers, of the Mamdani type, were created:

- The first, responsible for the linear velocity (a.k.a. linear velocity fuzzy controller), receives as input the linear and angular errors, and outputs the linear velocity, between 0 and 1.
- The second (a.k.a. angular velocity fuzzy controller) receives the linear and angular errors as input and delivers the angular velocity, varying from -1 to 1.

The linear error has 3 fuzzy classes: arrived, near, far and the representing fuzzy sets can be seen in Figure 2.

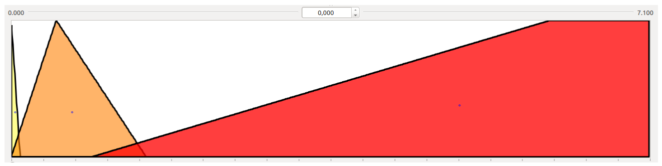

Fig. 2. Fuzzy sets for linear error. From left to right: class “arrived” is a triangle (0.0, 0.0, 0.1), class “near” is a triangle (0.0, 0.5, 1.5), and class “far” is a trapezoid (0.9, 6.0, 7.1, 7.1).

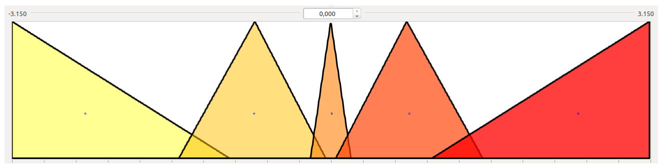

Fig. 3. Fuzzy sets for angular error. From left to right: class “high-negative” is a triangle (-3.15, -3.15, -1.0), class “low-negative” is a triangle (-1.5, -0.75, -0.05), class “straight” is a triangle (-0.2, 0.0, 0.2), class “low-positive” is a triangle (0.05, 0.75, 1.5), and class “high-positive” is a triangle (1.0, 3.15, 3.15).

The angular error has 5 fuzzy classes: high-negative, low-negative, straight, low-positive, high-positive and the representing fuzzy sets can be seen in Figure 3.

These fuzzy sets are used by the fuzzy controllers when calculating their output values in the defuzzification step, which follow the “centroid” calculation method. The fuzzy sets for the linear and angular velocity outputs can be seen in Figures 4 and 5, respectively. The rules considered for the linear velocity are presented in Table I, while the rules for the angular velocity are presented in Table II.

The output of both fuzzy controllers is combined into a single data type and written to a ROS topic (/cmd_vel). That topic is read by the V-REP script and the velocities are scaled and applied. V-REP scales these signals and the range goes from -1.0 to 1.0, to the broader range of $-240 \times \pi/180$ to $240 \times \pi/180$, to be applied individually in each wheel according to the rules defined in equations (1-4). This cycle goes on until the robot has reached its destination (linear error is smaller than 0.05). For this project, it is assumed that the odometry errors of the robot are negligible, and this is not the case for most real robots, but since the localization of the robot is not critical, any robot with some kind of SLAM (Simultaneous Localization and Mapping) navigation algorithm should be able to position the robot correctly in relation to the manipulation area (the table).

Once the goal is reached, the robot’s orientation is adjusted in relation to the table. The fuzzy control for angular velocity

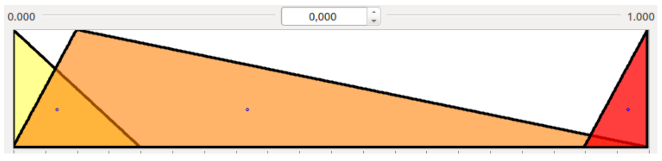

Fig. 4. Fuzzy sets for linear velocity output. From left to right: class “low” is a triangle (0.0, 0.0, 0.2), class “medium” is a triangle (0.0, 0.1, 1.0), and class “high” is a triangle (0.9, 1.0, 1.0).

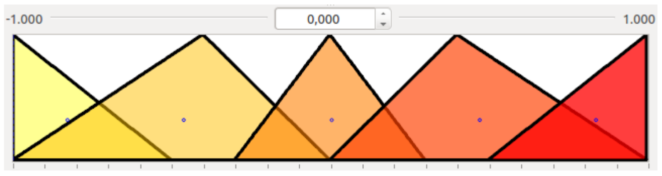

Fig. 5. Fuzzy sets for angular velocity output. From left to right: class “strong-left” is a triangle (-1.0, -1.0, -0.5), class “left” is a triangle (-1.0, -0.4, 0.0), class “straight” is a triangle (-0.3, 0.0, 0.3), class “right” is a triangle (0.0, 0.4, 1.0), and class “strong-right” is a triangle (0.5, 1.0, 1.0).

is once again activated, with the desired orientation redefined, keeping the linear velocity in 0. A small modification is made in this point, to ensure a faster convergence: the output is always lower than -0.1 or higher than 0.1, avoiding low acceleration levels. The cycle goes on until the robot has reached the desired orientation (angular error is smaller than 0.05).

B. Arm Positioning Control

The V-REP script that handles youBot movement also controls the joints of the arm. There are 5 joints and a gripper in the arm, and the joint values vary in their ranges [7]. The script supports forward kinematics (FK), where the 5 joint values have to be provided, but also supports inverted kinematics (IK), where the position of the arm’s tip can be defined in 3 dimensions (*x*, *y* and *z* axis), and the algorithm calculates the joint values to try to reach that destination. ROS code will control the arm writing in 10 topics (the full topic list can be seen in Table III):

- One topic (is_fk_mode), with message type std_msgs/Int32, defines the arm control mode: 0 means deactivated, 1 means FK and 2 means IK.
- Five topics (joint_1 up to joint_5), with message type std_msgs/Float32, define the values to be applied in FK mode, one for each joint.
- Three topics (ik_desired_x, ik_desired_y, ik_desired_z), with message type std_msgs/Float32, define the values

TABLE I
LINEAR VELOCITY FUZZY CONTROLLER RULES

Angular Error	Linear Error	Linear velocity
high-negative OR high-positive	far OR near	low
low-negative OR low-positive	far OR near	medium
straight	far OR near	high
*	arrived	low

TABLE II
ANGULAR VELOCITY FUZZY CONTROLLER RULES

Angular error	Angular velocity
high-negative	strong-left
low-negative	left
straight	straight
low-positive	right
high-positive	strong-right

TABLE III
ROS TOPIC LIST

Topic name	ROS message type
/cmd_vel	geometry_msgs/Twist
/cube_x	std_msgs/Float32
/cube_y	std_msgs/Float32
/ik_desired_x	std_msgs/Float32
/ik_desired_y	std_msgs/Float32
/ik_desired_z	std_msgs/Float32
/is_gripper_closed	std_msgs/Int32
/is_fk_mode	std_msgs/Int32
/joint_1	std_msgs/Float32
/joint_2	std_msgs/Float32
/joint_3	std_msgs/Float32
/joint_4	std_msgs/Float32
/joint_5	std_msgs/Float32
/odom	geometry_msgs/PoseStamped
/tip_x	std_msgs/Float32
/tip_y	std_msgs/Float32
/tip_z	std_msgs/Float32

to be applied in IK mode, one for each axis.

- One topic (is_gripper_closed), with message type std_msgs/Int32, defines the gripper state: 0 means open and 1 means closed. The gripper is used to grasp the object.

When V-REP script reads anything different than 0 from the arm control mode topic, it starts the arm control. If it reads 1, it starts FK control, reads the 5 values from the joint value topics and applies them. If it reads 2, it starts IK control, reads the 3 axis values from the respective topics and passes them to the IK algorithm. Therefore ROS code can control not only the position of youBot base, but also the position of its arm.

C. Camera Positioning For Object Detection

The goal of the first phase of the arm movement is to point the camera "down", towards the cube, so that the filters defined in the blob detection camera can successfully detect the cube. In the simulations, it was difficult to position the arm in the desired position and orientation using IK mode, since the algorithm is prepared to receive only the desired position of the arm's tip (x, y and z axis), not taking orientation in consideration. With this limitation, and considering that the table height is known, the FK mode was chosen. That way, the joint values for the 5 joints are individually applied, and the joints can be moved as desired. A good position for the joints was found through empirical tests, putting the arm's tip over the table, with the tip itself and the camera facing downwards, as seen in Figure 6.

The blob detection is implemented as a filter in V-REP, and it takes two parameters: threshold and minimum size of blobs (the filter will ignore blobs smaller than the size informed). The filter was tested with different threshold levels, and finally set to 0.8, detecting the white cube as soon as it completely enters the camera image frame. A V-REP API function (`simReadVisionSensor`) can be accessed in Lua, returning a variety of information about the detected blobs, including the blob count, the blob sizes, orientations and positions. With this function, it is possible to retrieve the object position within the image. An example of the detected blob can be seen in Figure 6.

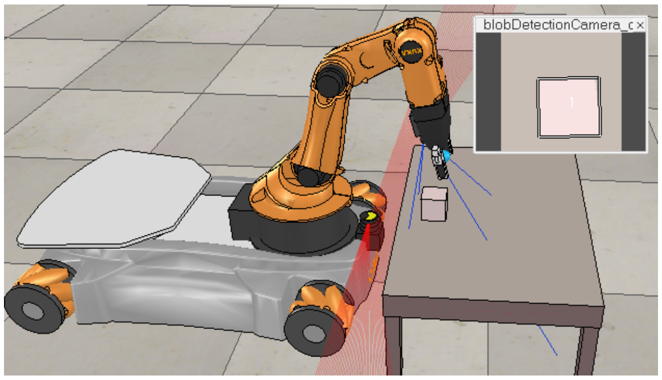

Fig. 6. Arm position and blob detection. The window shows the camera view, with the object correctly segmented, as can be seen by the highlighted edges.

D. Object Manipulation

After detecting the cube's position in the image, the goal is to pick it up and put it in the robot's base. The V-REP simulator publishes the cube's position in a topic, while ROS provides the information to the other topics related to arm positioning. After receiving the cube's position, a ROS node calculates and sets the position to be applied in IK mode. V-REP, checking that the arm control topic has been set to '2', activates IK mode and moves the arm to the desired position. The ROS code then orders the gripper to close and grab the cube.

The next step is to move the arm to a position where the cube can be released in the robot's base. This is a fixed position, relative to the arm. Setting the joint values in FK mode, the arm can be moved to the desired position, and then the gripper is opened, delivering the object to the robot's base, as seen in Figure 7.

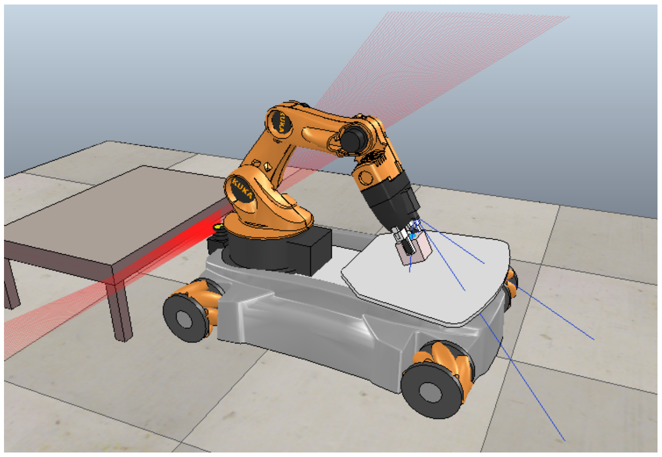
Fig. 7. Arm delivering the cube to the robot's base

IV. RESULTS

Integrating ROS and V-REP for controlling position and movement of the youBot robot involved the application of a variety of concepts, including fuzzy logic controllers and image processing for object detection. The arm control also introduces the concepts of forward kinematics and inverted kinematics, and ROS code controlled the robot, providing the coordinates to the V-REP simulation script.

One execution was made with the robot starting at position *(x,y)*=(-1.5,-1.5), with 180° rotation away from the table (facing directly the opposite direction). The fuzzy controllers output and the errors progression over time from this execution can be seen in Figure 8. It is clear that, at first, the angular velocity is high, and diminishes slowly up to the moment when the angular error approaches 0, while the linear velocity rises progressively as the angular error lowers. When the goal is reached, linear velocity goes down to 0, and the only thing left is to fix the robot's rotation.

The blob detection filter, implemented inside V-REP, was able to successfully detect and segment an object, the white cube, allowing successful manipulation in 100% out of 10 execution attempts, with the robot starting at different initial positions. Since the experiments were executed on a simulation environment, there is not much difference in the environmentally dependant variables from one execution to the other (which is not the case in real world applications). Once the robot reaches the table and rotates, there is little variety from this point onward between different executions.

Another attempt was made replacing the cube with the model of a cup (a predefined V-REP model). In this case, the blob detection filter struggled to define the cup as a whole, and instead, identified a variety of blobs over the cup.

As for the fuzzy controllers for robot navigation, the results were satisfying, taking the robot to the correct location in 100% of the execution attempts. The simulation environment made the task easier, providing the odometry for the robot. In real world scenarios, odometry can accumulate error over time, because of slipping wheels or irregular floor.

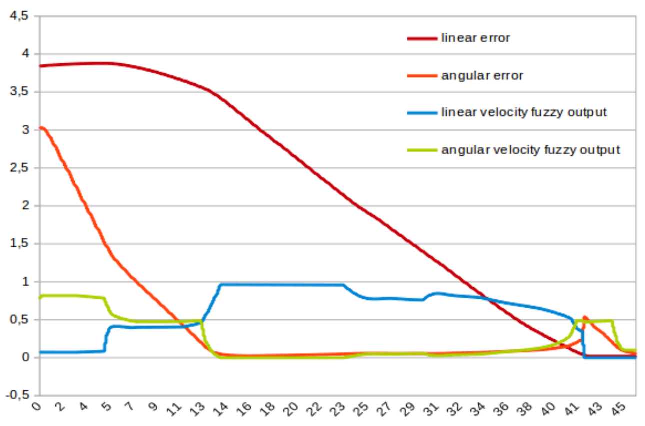

Fig. 8. Linear error, angular error, linear velocity fuzzy output and angular velocity fuzzy output progression over time.

V. CONCLUSIONS

To what concerns robotics simulation, ROS code and concepts can be easily tested in a V-REP environment. There are several predefined models and if you can't find one that suits your needs, you can define your own. Fuzzy controllers proved to be efficient for controlling the robot's position and movement. The blob detection from V-REP is a simple way to detect and segment smooth and clear objects, but complex scenarios and objects demand a more sophisticated approach (with deep learning or approaches like using OpenCV [11] being good candidates). The presented approach avoids the need for a QR-Code marker on the object, as proposed by Santos [4], opening new possibilities for object detection and solving real manipulation problems in future work.

Since the focus was to build a simple environment that encourages learning for new researchers in the field, and ROS has a rich variety of solutions available, each component can be adapted or replaced by different approaches, with different libraries. Future work could address navigation challenges that could be put into place, like obstacles, collision-avoidance and SLAM. The search for the object position without knowing it in advance could involve more sophisticated computer vision techniques. Different objects and object recognition approaches could be put into place. The final challenge would be to put a robot to act in the real world, identifying and manipulating different kinds of objects.

The complete source code of this project is available at the following github repository: https://github.com/laser-utfpr/vrep_youbot.git.

REFERENCES

[1] Rohmer, E.; Singh, S. P. N.; Freese, M.; "V-REP: a Versatile and Scalable Robot Simulation Framework", in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013.
[2] "RoboCup@Home". https://www.robocupathome.org. Accessed July 2019.
[3] Matamoros M.; Seib V.; Paulus D. "Trends, Challenges and Adopted Strategies in RoboCup@Home", in 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). https://ieeexplore.ieee.org/document/8733622. Accessed July 2019.
[4] Santos, A. G. D.; Santos, D. H.; Palar, P. S.; Oliveira, A. S.; Fabro, J. A. "Métodos de Controle de Movimento de Veículo-Manipuladores: Acoplado e Nâo-Acoplado", in Proceedings of the II Brazilian Humanoid Robot Workshop (BRAHUR) and III Brazilian Workshop on Service Robotics (BRASERO), pp. 96-101, 2019.
[5] "ROS.org — Powering the world's robots". http://www.ros.org. Accessed June 2019.
[6] "KUKA youBot. Mobile manipulator for research and education". https://bit.ly/2r1uyql. Accessed June 2019.
[7] "KUKA youBot. Research & Application Development in Mobile Robotics". https://bit.ly/2HR9ZF0. Accessed June 2019.
[8] "Scanning Rangefinder Distance Data Output/URG-04LX-UG01 — HOKUYO AUTOMATIC CO., LTD.". https://www.hokuyo-aut.jp/search/single.php?serial=166. Accessed June 2019.
[9] Koslosky, E.; Oliveira, A. S.; Wehrmeister, M.; Fabro, J. A. "Designing Fuzzy Logic Controllers for ROS-Based Multirotors", in Robot Operating System (ROS), The Complete Reference (Volume 2), Part of Studies in Computational Intelligence Series, vol. 707, p. 41-82, DOI: 10.1007978-3-319-54927-9_2, 2017.
[10] "FLIE - Fuzzy Logic Inference Engine". https://github.com/joaofabro/FLIE/. Accessed June 2019.
[11] "OpenCV". https://opencv.org/. Accessed June 2019.