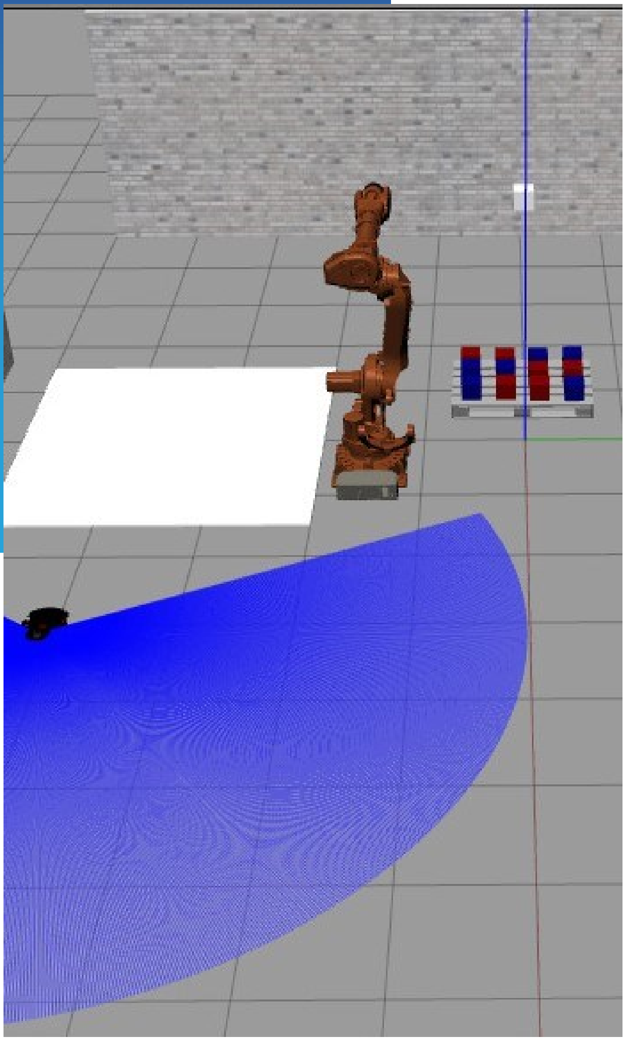


REPORT

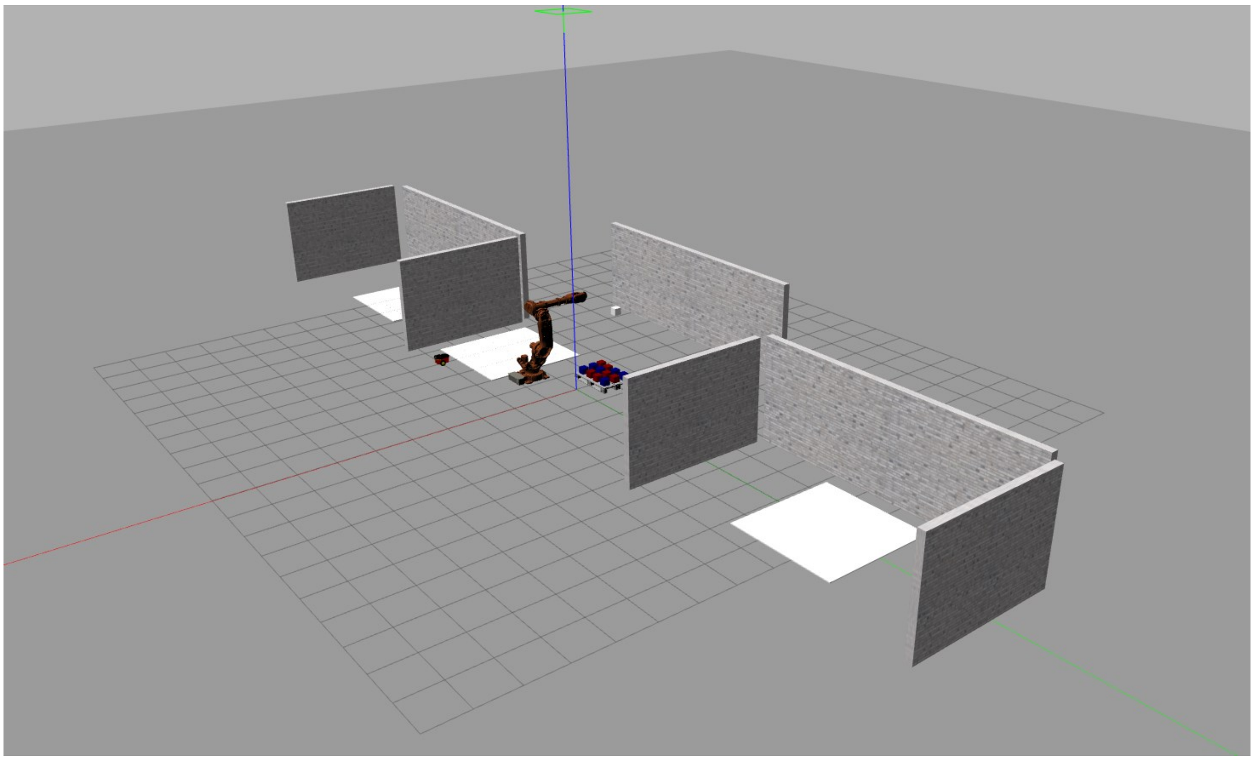
Smart Warehouse

University of Naples Federico II

23/06/2020

Eliana La Frazia

Andrea Cavaliere



Abstract

In the following report the students Andrea Cavaliere and Eliana La Frazia present the results achieved in the project "Smart Warehouse 2".
Purpose of the project is the realization of an autonomous warehouse environment where an industrial robot manipulator has to depalletize a pallet of at least 10 boxes (5 red and 5 blue) and an AGV is in charge to bring them to specific platforms.

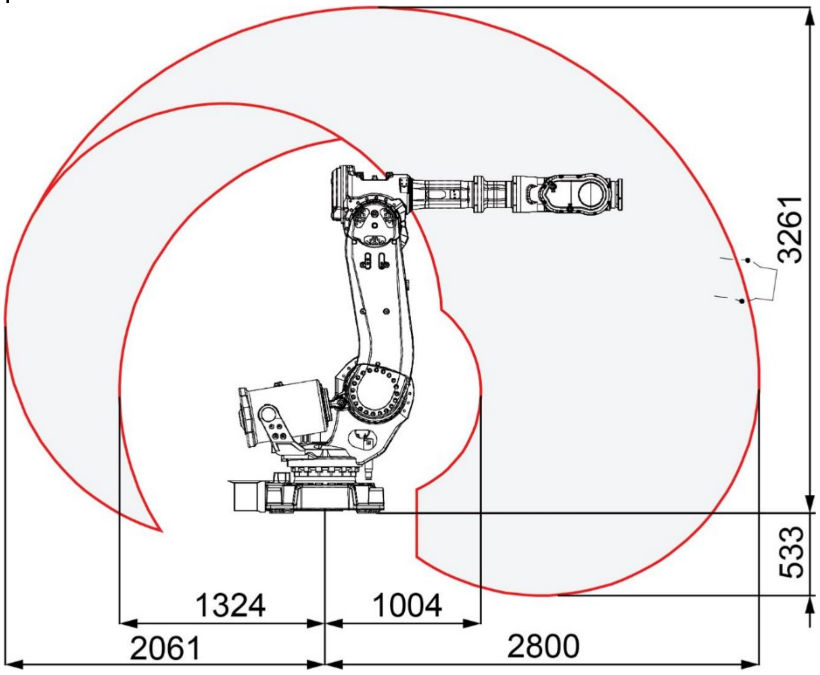
Topics

1 Robots

Various types of robots have been investigated to perform the two different tasks. The manipulator should be robust and with a large working space and the mobile robot should be efficient and versatile.

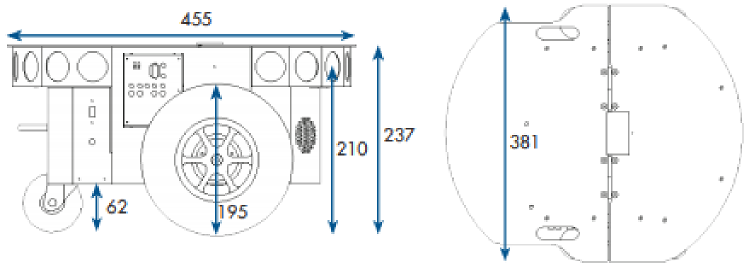
1.1 ABB IRB6640

The ABB irb 6640-185 was chosen mainly because of its wide range of action. As the robot can bend fully backwards, the working range is greatly extended allowing it to fit well into dense production lines.



1.2 Pioneer

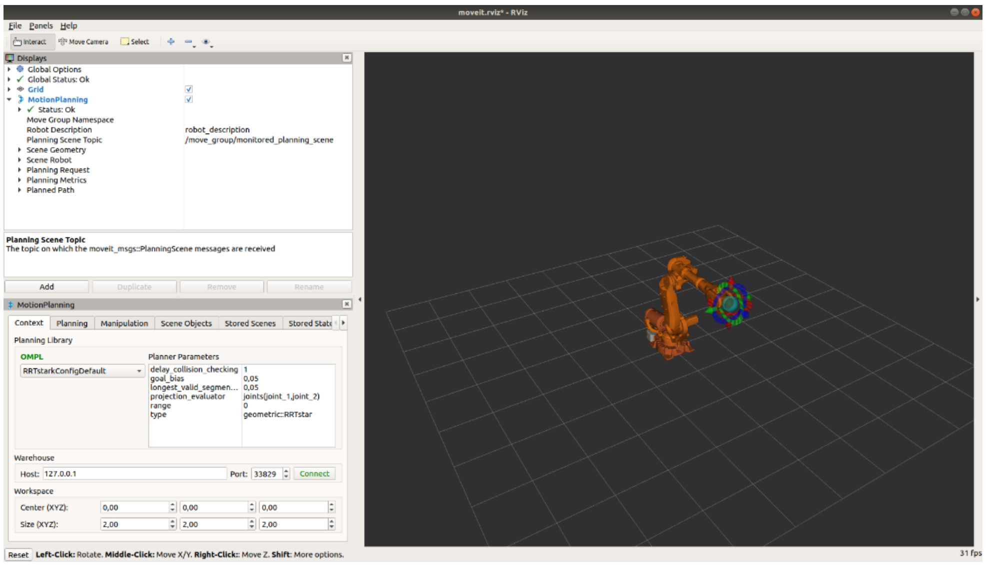
Pioneer 3-DX is a small lightweight two-wheel two-motor differential drive robot ideal for indoor environment. The model we have chosen is equipped with a 2D LiDAR sensors LMS1xx / Indoor (0.1 m - 4 m).



2 Implementation

2.1 Environment Initialization

Since we didn't find a gazebo model of the ABB IRB6640 we made an adaptation of the package abb_irb6640_support, renamed to abb_irb6640_description, changing the urdf file in order to spawn the robot in the gazebo environment. We created a new package in order to interface the Moveit! control plugin (implemented in the ROS Industrial ABB stack) with gazebo. To do so we started from a working example found on internet of the ABB IRB120 and adapted it to our robot.



```
Global Options
Global Status: Ok
Grid                          ✓
MotionPlanning                ✓
  Status: Ok
  Move Group Namespace
  Robot Description           robot_description
  Planning Scene Topic        /move_group/monitored_planning_scene
  Scene Geometry
  Scene Robot
  Planning Request
  Planning Metrics
  Planned Path

Planning Scene Topic
The topic on which the moveit_msgs::PlanningScene messages are received

Planning Library
OMPL                          Planner Parameters
RRTstarkConfigDefault         delay_collision_checking  1
                              goal_bias                 0,05
                              longest_valid_segmen...   0,05
                              projection_evaluator      joints(joint_1,joint_2)
                              range                     0
                              type                      geometric::RRTstar

Warehouse
Host: 127.0.0.1                Port: 33829

Workspace
Center (XYZ):  0,00   0,00   0,00
Size (XYZ):    2,00   2,00   2,00

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.
```

Figure 1 Rviz Moveit! window

Regarding the Pioneer, from the work of Rafael Berkvens and Mario Serna Hernández we got the Pioneer 3DX description and control packages. Then we created the p3dx_move package with the implementation of move base, AMCL and SLAM (thanks to gmapping) algorithms.

2.2 How it works

We supposed to have a RGBD camera in warehouse. We give the opportunity to tune the colour of the boxes, just in case the user wants to redefine the exact shades of colours.
We implemented the package *smartwarehouse_opencv* which uses the library OpenCV. After the calibration, the camera detects all the objects whose colour is in the defined range and with a rectangular shape. The implemented algorithm allows the camera to detect boxes of different size (obviously the dimensions must differ slightly from the starting ones to work with the rest of the project) because the camera doesn't have to know the dimensions a priori and neither the number of the boxes but it can calculate them by itself. It would be better if the boxes are placed with a gap between them. The camera node is subscribed to a Gazebo topic in order to achieve the gazebo models names and once it detects a box position it calculates the box model nearest to this position and identifies its name. This node is also an action client to send a goal to the manipulator action server node.

We also implemented a node to manage the manipulator pick and place task that is the action server. The ABB action server node receives the goal from the camera node, consisting of the box position, colour and the gazebo model name. The ABB node provides to reach the box position using Moveit!. To make the attach between the end effector and the box we use a package in which is implemented a virtual link. This service requires to know the gazebo models

names (provided from the camera node) and the boxes to be dynamic, so we had to modify the sdf file of each model and the warehouse2.world.
We assumed to have two conveyor belts, each of them takes a different colour, so the ABB manipulator places the boxes in sequence on them. The sorting policy of boxes consists of selecting before all the red boxes and then the blue boxes. After the end of the depalletization, the manipulator is immediately ready to depalletize another pallet.

The task of the mobile robot(or robots) is to deliver the boxes to their corresponding platforms. To realize a robust infrastructure which can support a routine with one or two Pioneers we created a "simple navigation node" for each Pioneer and a sort of "manager node" that coordinates the delivering and communicates with the ABB manipulator thanks to a publish subscribe policy where every time the manipulator places a new box on one of the conveyor belts it publishes a string that will be parsed from the *pioneer_manager* in order to get model name and colour. Thanks to this information the node will update a queue with the names of the models and their colour. The first information is needed to implement the virtual link between the boxes and the top-plate of the Pioneer and the second one to set the right destination to give to each Pioneer.
At the beginning, since the robot has no knowledge of the environment we drove it through the warehouse using move base and gmapping, in order to create a map of the place.

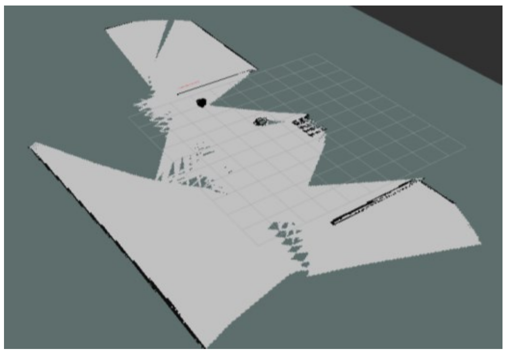


Figure 2 Resulting Map

After that we substituted the SLAM algorithm with the AMCL and let the AGV drive autonomously.

3 Testing

3.1 Depalletize: typical scenario

We tested the algorithms:

- Changing the number and positions of blue boxes and red boxes
- Adding other boxes after the end of the depalletization
- Putting all red boxes or all blue boxes on the pallet.

The position recognition algorithm and the pick and place algorithm work also if the boxes are placed one above the other but it’s hard to test it in simulation because of the boxes are dynamics.
During the testing there were some failures of trajectory planning between two distant points, these failures are related to the simulation environment because all the points the robot has to reach are in the working space and are absolutely reachable (both position and orientation assigned).

For this reason the failures are supposed to be rare so if the planning fails during a pick phase the robot try to return in start position and the action client resend the box position, instead if it fails during a place phase the robot try to return in a start position and to replan the trajectory to reach the place pose and if it fails again then it aborts.

3.2 Obstacles avoidance

Since a warehouse is a semi-structured environment where human beings can stand on the robot's navigation path we tested how the mobile robots react to unexpected obstacle(not marked on the map), and then tried to tune the planner parameters (like simulation time of the local parameters, tolerance, dimensions of the local map) to get the best trade-off between obstacle avoidance(to ensure operators safety) and efficiency.
Since the local map save the laser readings into the local map it could happen that even if the obstacle gets away from the path the robot tries again to “avoid” it. This can happen near a destination pose like the base or one of the platforms. Obviously, it can happen only one time since the local map is updated after a while.