INTRODUCTION

Factories of Industry 4.0 will be populated by human operators and robots able to cooperate and collaborate in order to fulfill complex, non-repetitive tasks. An example is the storage of goods and their subsequent assembly: mobile manipulator robots will help humans with the retrieval of goods for shelves and their transportation within the factory warehouse. This means that they will be able to autonomously navigate without the need of following electrified paths or lines drawn on the ground. They will be able to autonomously compute the optimal path from their current pose to a target one while avoiding fixed or moving obstacles. They will be able to recognize and manipulate those goods that need to be transported.

Figure 1 – The simulated and real workspace

PROJECT’S DESCRIPTION

Preamble:

Suppose to have a set of \( N \) goods in your warehouse, each of them identified by a unique marker. These goods, or subsets of \( n \) of them, can be exploited in order to fulfill different assembly tasks.

Task:

Suppose to have a subset of \( X \leq N \) goods on a table. A human operator will ask for \( x \) of them, according to the desired assembly task (from command line, \( 0 < x \leq X \)). A camera is mounted on that table, together with a UR10 manipulator robot. This visual sensor has to be exploited to detect the correct goods (via Apriltag) and communicate their poses (position and orientation) to the UR10. The UR10 has to pick up the objects, one by one, and place them on the top of a Marrtino mobile robot, that has previously docked
itself near the manipulator. Martino will bring the pieces to a second docking station, near a human assembler. During its path, the mobile robot will face a narrow passage delimited by fixed barriers and an open space populated by fixed and movable obstacles. The human assembler will pick up the objects and assemble them. Keep on until all useful pieces have been carried on the human workstation or a prefixed timeout is reached (see Figure 1).

**CONSTRAINTS**

- Total time: 30 min;
- Area on the top of the mobile robot is limited: an optimal policy must be implemented letting Martino bring to the largest number of pieces to the assembly station;
- Docking stations’ poses are fixed with respect to the pose of the UR10 and the pose of the assembly station;
- The open area is populated by fixed and movable obstacles;
- The mobile robot always starts from a prefixed pose;
- Only desired objects have to be manipulated;
- During the manipulation, the UR10 does not have to collide with the objects on the table;
- Objects on the table can be shift by a human operator while Martino is navigating.

**USEFUL INSTRUCTIONS**

**ROS Kinetic installation:**


**Make your own workspace:**

Make a workspace named ros_ws: [http://wiki.ros.org/catkin/Tutorials/create_a_workspace](http://wiki.ros.org/catkin/Tutorials/create_a_workspace)

**Lab packages installation:**

To install the lab environment, register to BitBucket and send us your username. You will be registered as IAS-lab team member! Then, you will be able to download and use the following script:


To be sure you can execute the script, grant execution permission to the file:

```
cd [download_directory]
chmod +x install.sh
```

Finally, source the script:
source install.sh

!!! Check the path to your ROS workspace in /arena_collision/pysdf/src/pysdf/parse.py

Build your code:
When you want to build your code, use catkin_make: [http://wiki.ros.org/catkin/commands/catkin_make](http://wiki.ros.org/catkin/commands/catkin_make) (install.sh already calls catkin_make)

**BASIC INSTRUCTIONS FOR THE LAB PACKAGES – SIMULATION**

The simulated environment can be easily launched by using the command:

```
roslaunch challenge_arena challenge.launch sim:=true
```

**Ur10 MoveIt! setup:**

1) **Launch the RViz MoveIt! plugin:**

```
roslaunch ur10_platform_challenge_moveit_config ur10_platform_challenge_moveit_planning_execution.launch sim:=true
roslaunch ur10_platform_challenge_moveit_config moveit_rviz.launch config:=true
```

Set planning request to the “manipulator” group

**Robotiq 3-Finger Gripper:**

1) **Open:**

```
rostopic pub --once /robotiq_hands/l_hand/SModelRobotOutput robotiq_s_model_control/SModel_robot_output "{rACT: 1, rMOD: 0, rGTO: 1, rATR: 0, rGLV: 0, rICF: 0, rICS: 0, rPRA: 0, rSPA: 0, rFRA: 0, rPRB: 0, rSPB: 0, rFRB: 255, rPRC: 0, rSPC: 0, rFRC: 0, rPRS: 0, rSPS: 0, rFRS: 0}"
```

2) **Close:**

```
rostopic pub --once /robotiq_hands/l_hand/SModelRobotOutput robotiq_s_model_control/SModel_robot_output "{rACT: 1, rMOD: 0, rGTO: 1, rATR: 0, rGLV: 0, rICF: 0, rICS: 0, rPRA: 255, rSPA: 255, rFRA: 150, rPRB: 0, rSPB: 0, rFRB: 0, rPRC: 0, rSPC: 0, rFRC: 0, rPRS: 0, rSPS: 0, rFRS: 0}"
```

3) **Attach (example):**
rosservice call /link_attacher_node/attach "model_name_1: 'robot'
link_name_1: 'wrist_3_link'
model_name_2: 'cube3'
link_name_2: 'cube3_link'"

4) Detach (example):

rosservice call /link_attacher_node/detach "model_name_1: 'robot'
link_name_1: 'wrist_3_link'
model_name_2: 'cube3'
link_name_2: 'cube3_link'"

BASIC INSTRUCTIONS FOR THE LAB PACKAGES – REAL

LOCAL:
- Kinect activation:
  roslaunch kinect2_bridge kinect2_bridgw.launch sensor_name:=camera

- Apriltag:
  roslaunch apriltags_ros challenge.launch

- Connection to the manipulator robot:
  roslaunch challenge_arena challenge.launch sim:=false

- Check that you are receiving the Kinect information
  rqt_image_view

- Launch your node

!!! The workstation pcrob14 is set to be the MASTER! (MASTER_IP:= 192.168.80.100) Have a look at .bashrc! Every laptop that you decide to connect to the network MUST have the ROS MASTER set to this IP address!

Ur10 MoveIt! setup:

1) Launch the RViz MoveIt! plugin:

roslaunch ur10_platform_challenge_moveit_config ur10_platform_challenge_moveit_planning_execution.launch sim:=false
roslaunch ur10_platform_challenge_moveit_config moveit_rviz.launch config:=true

Set planning request to the “manipulator” group

**Robotiq 3-Finger Gripper:**

1) **Open:**

```bash
rostopic pub --once /robotiq_hands/l_hand/SModelRobotOutput robotiq_s_model_control/SModel_robot_output "{rACT: 1, rMOD: 0, rGTO: 1, rATR: 0, rGLV: 0, rICF: 0, rICS: 0, rPRA: 0, rSPA: 0, rFRA: 0, rPRB: 0, rSPB: 0, rFRB: 255, rPRC: 0, rSPC: 0, rFRC: 0, rPRS: 0, rSPS: 0, rFRS: 0}"
```

2) **Close:**

```bash
rostopic pub --once /robotiq_hands/l_hand/SModelRobotOutput robotiq_s_model_control/SModel_robot_output "{rACT: 1, rMOD: 0, rGTO: 1, rATR: 0, rGLV: 0, rICF: 0, rICS: 0, rPRA: 255, rSPA: 255, rFRA: 150, rPRB: 0, rSPB: 0, rFRB: 0, rPRC: 0, rSPC: 0, rFRC: 0, rPRS: 0, rSPS: 0, rFRS: 0}"
```

**Temporary 3d printed gripper**

1) **Connection:**

```bash
rosrun rosserial_python serial_node.py _port=/dev/ttyACM0 _baud=9600
```

2) **Open:**

```bash
rostopic pub /angle_motor std_msgs/UInt16 "data: 10"
```

3) **Close:**

```bash
rostopic pub /angle_motor std_msgs/UInt16 "data: 150"
```

!!! Check port permission: sudo chmod 777 /dev/ttyACM0