1. Denso Simulation setup

1. Introduction

This section describes how to setup Denso robot simulation in Gazebo along with Kinect sensor.

2. Installation

- Gazebo_ros_pkgs (this package is already installed on your system)
- denso_robot_ros (this package has Moveit configuration packages for different Universal robots)

```
$ cd moveit_ws/src
$ git clone https://github.com/ipa-hsd/denso_robot_ros
$ cd ..
$ catkin_make
$ source devel/setup.bash
```

3. Startup

Open each command below in a new terminal:

```
$ roslaunch denso_moveit_config_pkg denso_robot_gazebo.launch
$ roslaunch denso_moveit_config_pkg denso_moveit.launch sim:=true
```

4. RViz configuration

1. Add the “Motion Planning” type into RViz
2. Set the fixed frame to “world”

2. AR Alvar Tag Detection

1. Introduction

Object recognition is essential in higher level robotic applications. To achieve this, the goal in this tutorial is to detect the fiducial markers ("AR Tags") based on RGB data given by a simulated RGBD camera.

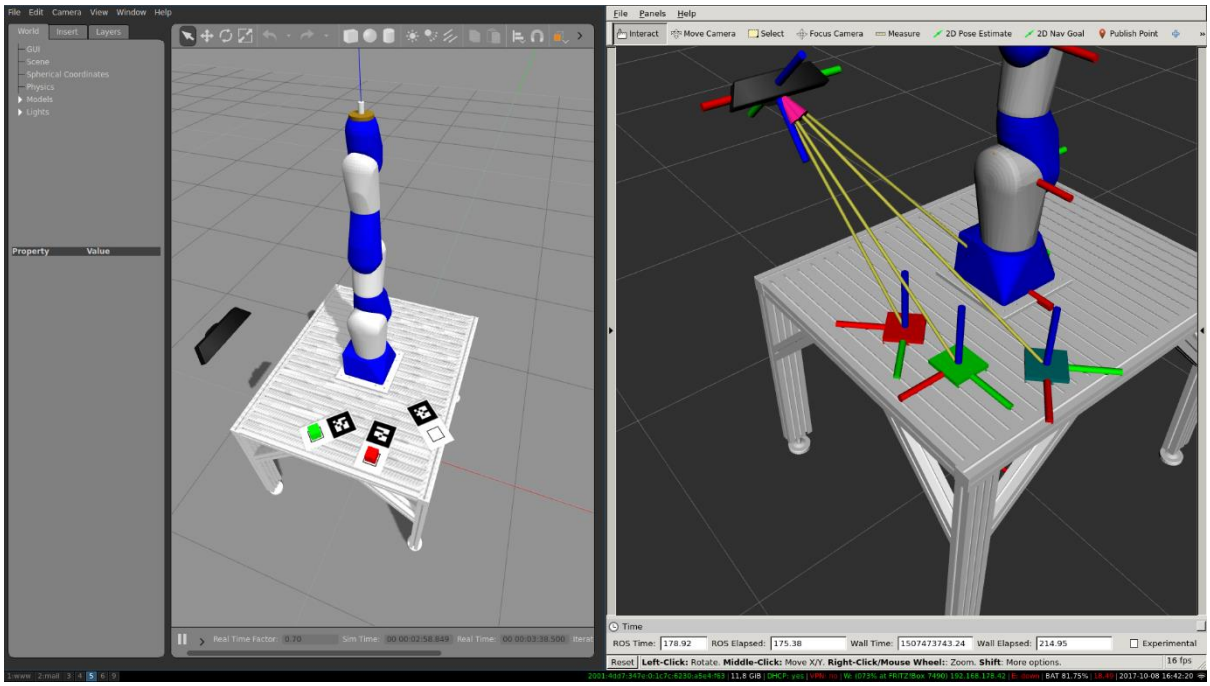

Figure 1: Simulation environment

2. Installation and Startup

AR Markers are fiducial markers (see figure 2). They provide a way of visual pose estimation. AR Markers can be detected by the *ar_track_alvar* package. This package can be installed in the following manner along with the package *openni2_launch* to read camera images:

```
$ sudo apt-get install ros-kinetic-ar-track-alvar
$ sudo apt-get install ros-kinetic-openni2-launch
```

Start the launch file *ar.launch* from the *moveit_tutorial package* to detect the AR Tags in the Simulation as well. AR Track Alvar works out of the box by just running it, you

only need to determine the length of one side of the tag and the image topic to be used. This is parameterized in the launch file.

```
marker size: 13cm x 13cm
camera image topic: /camera/rgb/image_raw
camera info topic: /camera/rgb/camera_info
output frame: /camera_link
```

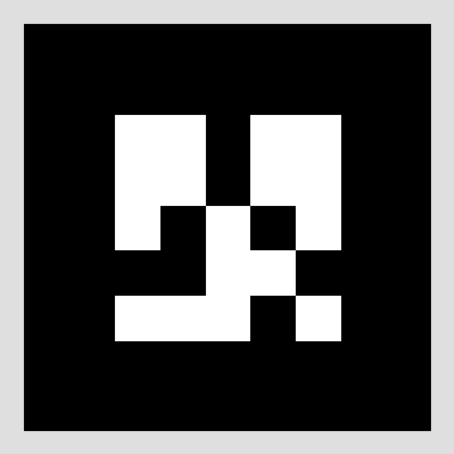

Figure 2: AR Tag showing number 1

When you start up AR-Track Alvar you can visualize the detected AR Tags by enabling the topic /visualization_marker in Rviz. You can also visualize (and make use of) tf data provided by AR-Track Alvar as it also represents detected Markers by publishing transforms.

Once you started it up for the first time and set up the appropriate visualizations in rviz you'll probably notice a screen like the following:

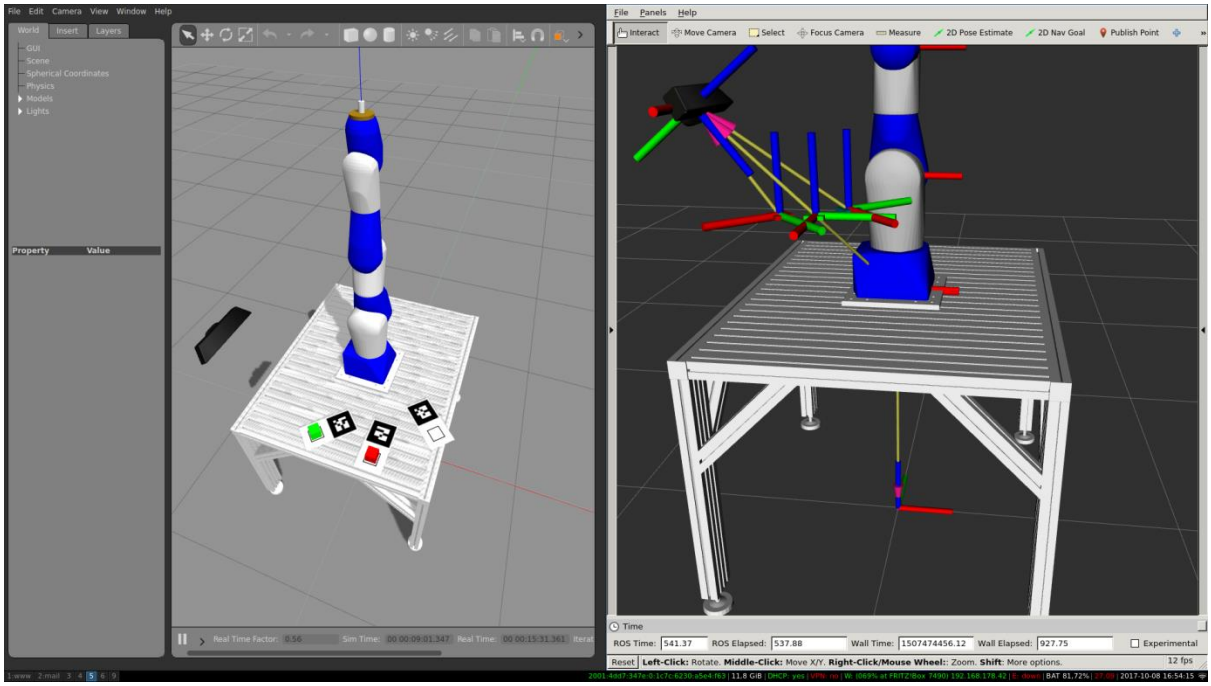

Figure 3: AR Tags at wrong size

This is caused by a wrong value of "marker_size" in the ar_track_rgb.launch file. Try to tweak this value until you have a setup as shown in figure 1.

3. Output Analysis

Running AR-Track alvar will provide a new topic called /ar_pose_marker which provides information about the detected AR Tags. These messages can easily be used in ROS nodes to further interact with the perceived AR Tags.

3. Application: "Move arm above AR Tag"

1. Introduction

This section describes the final task for this session. It is supposed to wrap up your knowledge about URDF, TF and MoveIt! and create a real-world application with it.

2. Task description

The final task for the application development is to complete a python node that interacts with the UR5 MoveIt's moveit_commander interface to move the robot's end-effector above the AR Tag. See figure 1 for further description.

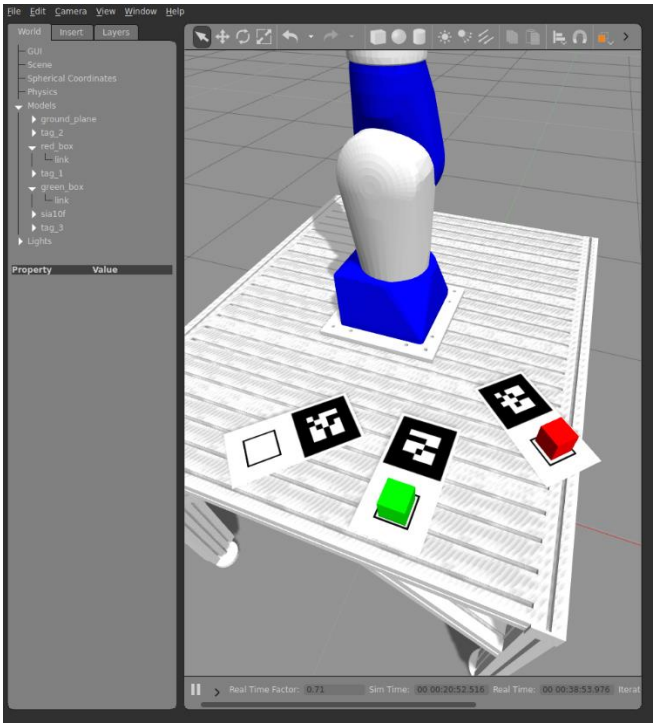

Figure 2:Final setup of boxes.

The node of interest is called arm_move.py in the moveit_tutorial package. It contains several methods to modify the position of the end-effector and to process data given by the AR Track Alvar node.

Your task is to simply complete the node to let the arm move above the AR tag in a continuous loop as shown in figure 1. Following is the list of tasks to be completed:

Task 1:
Instantiate ‘MoveGroupCommander’ with the correct ‘group name’. (line number 76)
Refer to Rviz to get the group name in the ‘Motion Planning’ tab
OR Check the SRDF file in the config folder of the respective moveit_config package

Task 2:
Add a subscriber to ar pose marker topic (line number 80)
Find out the message type of the topic from command line using rostopic

```
rostopic info topic_name
```

Task 3:
Assign position values from mPose to pose_target (line number 106)
HINT: Refer to method '*handle_marker*' to get type of mPose