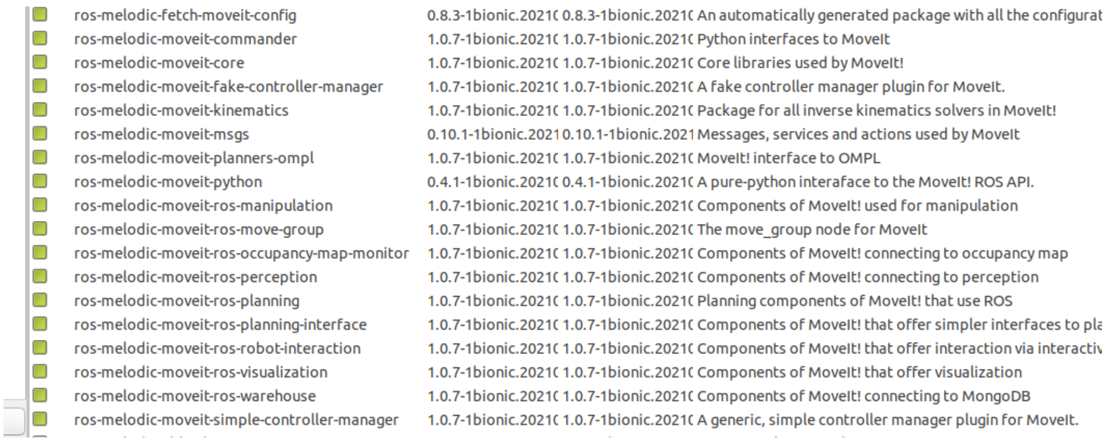
Homework-2 Robot Control

7 DOF MANIPULATOR SIMULATION ON ROS

As mentioned earlier, ROS is capable of prototyping, simulating and programming a wide variety of robots. In this homework, you are expected to simulate a 7 DOF Arm on ROS, using the Moveit control package and the related Python code. Moveit has many specific features (inverse kinematics, control, collision checking etc.) to handle this complex task. To get prepared for the process, firstly you need to follow the instructions below:

1) Download “packages.zip” folder and extract “seven_dof_arm_config” and “mastering_ros_robot_description_pkg” packeges into “catkin_ws/src” directory.

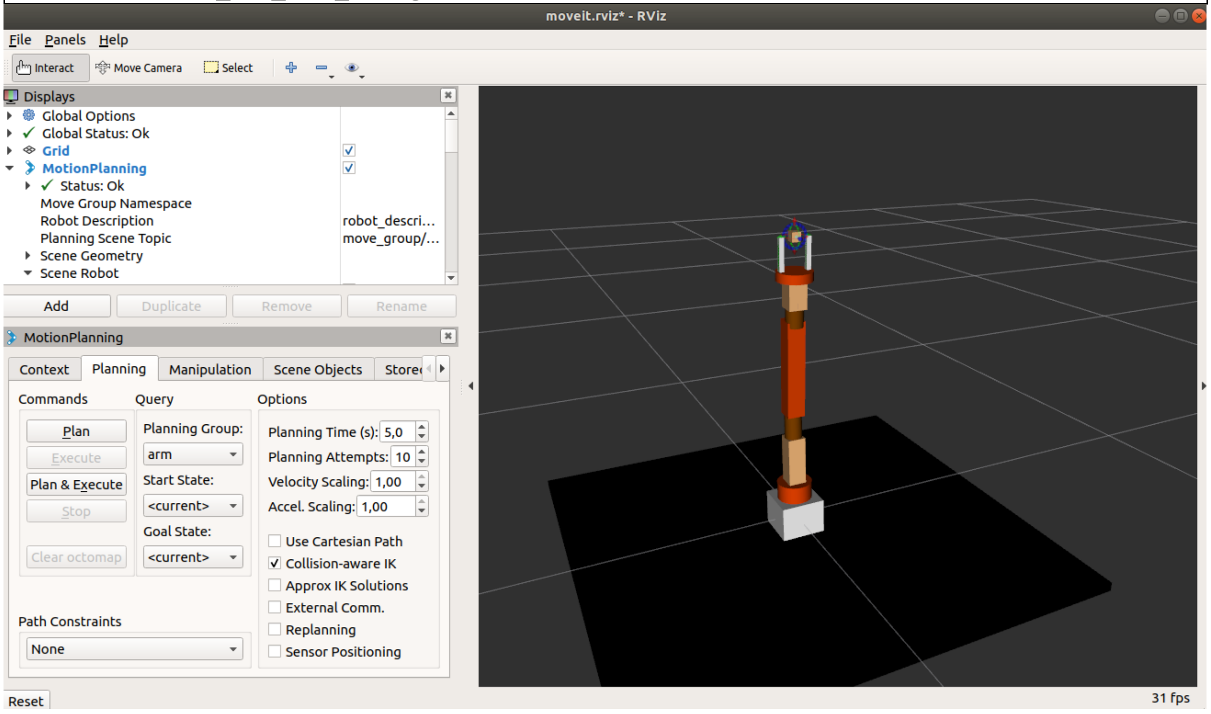
2) Our packages also depend on the Moveit packages. Please install the required packages by using Synaptic Package Manager tool. A list of my installations are given below:



🟩	ros-melodic-fetch-moveit-config	0.8.3-1bionic.2021(	0.8.3-1bionic.2021(	An automatically generated package with all the configurat
🟩	ros-melodic-moveit-commander	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Python interfaces to MoveIt
🟩	ros-melodic-moveit-core	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Core libraries used by MoveIt!
🟩	ros-melodic-moveit-fake-controller-manager	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	A fake controller manager plugin for MoveIt.
🟩	ros-melodic-moveit-kinematics	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Package for all inverse kinematics solvers in MoveIt!
🟩	ros-melodic-moveit-msgs	0.10.1-1bionic.2021	0.10.1-1bionic.2021	Messages, services and actions used by MoveIt
🟩	ros-melodic-moveit-planners-ompl	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	MoveIt! interface to OMPL
🟩	ros-melodic-moveit-python	0.4.1-1bionic.2021(	0.4.1-1bionic.2021(	A pure-python interaface to the MoveIt! ROS API.
🟩	ros-melodic-moveit-ros-manipulation	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Components of MoveIt! used for manipulation
🟩	ros-melodic-moveit-ros-move-group	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	The move_group node for MoveIt
🟩	ros-melodic-moveit-ros-occupancy-map-monitor	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Components of MoveIt! connecting to occupancy map
🟩	ros-melodic-moveit-ros-perception	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Components of MoveIt! connecting to perception
🟩	ros-melodic-moveit-ros-planning	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Planning components of MoveIt! that use ROS
🟩	ros-melodic-moveit-ros-planning-interface	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Components of MoveIt! that offer simpler interfaces to pla
🟩	ros-melodic-moveit-ros-robot-interaction	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Components of MoveIt! that offer interaction via interactiv
🟩	ros-melodic-moveit-ros-visualization	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Components of MoveIt! that offer visualization
🟩	ros-melodic-moveit-ros-warehouse	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	Components of MoveIt! connecting to MongoDB
🟩	ros-melodic-moveit-simple-controller-manager	1.0.7-1bionic.2021(	1.0.7-1bionic.2021(	A generic, simple controller manager plugin for MoveIt.

3) When the installation process is completed, running the code below will open a Rviz window as shown in the figure.

```
roslaunch seven_dof_arm_config demo.launch
```



moveit.rviz* - RViz

Displays: Global Options; Global Status: Ok; Grid ✓; MotionPlanning ✓; Status: Ok; Move Group Namespace; Robot Description: robot_descri…; Planning Scene Topic: move_group/…; Scene Geometry; Scene Robot

MotionPlanning — Planning tab

Planning Group: arm
Start State: <current>
Goal State: <current>
Planning Time (s): 5,0
Planning Attempts: 10
Velocity Scaling: 1,00
Accel. Scaling: 1,00
☐ Use Cartesian Path
☑ Collision-aware IK
☐ Approx IK Solutions
☐ External Comm.
☐ Replanning
☐ Sensor Positioning
Path Constraints: None

4) Practice manual operation using a GUI : To get familiar with this Moveit system, move the cube inside the gripper to different positions and orientations by translating and rotating it and observe the manipulator attaining different poses to achieve the desired cube poses. Practice with the MotionPlanning menu to get familiar with the system.

5) Programming for manipulation : First prepare the coding environment. Open your Pycharm IDE and create a new Python 2.7 project. You will use the python code example ("test_arm.py") with the following capabilities:

- Initialization of the Moveit API (Application Programming Interface).
- An example of performing open loop set point tracking for each manipulator joint.
- An example of attaining a desired pose for the end effector. Note: The inverse kinematics and trajectory planning are handled by the Moveit packages.
- An example of end effector trajectory tracking.
- Visualization of the planned trajectory and tracking process on RViz

This code is taken and modified from Moveit python tutorial github directory. In order to use the Moveit functions properly, you should study the Moveit python wiki page:
http://docs.ros.org/en/melodic/api/moveit_tutorials/html/index.html

Assignments:

1) Manipulate each joint position with different references and show the modifications you made to the code **clearly** with necessary explanation. (15p)

2) Manipulate the end effector to two different pose references, and show the modifications you made to the code **clearly** with necessary explanation. (15p)

3) Now, you are expected to make end effector follow a trajectory. The desired trajectory to be followed by the end effector is the last digit of your student ID. By considering robot workspace limits, create and follow the trajectory by using the "cartesian path" function. (30p)

4) For this section, you are expected to follow the same trajectory by using the "go_to_pose_goal" function. (30)

5) Briefly explain the difference between the coding process in (3) and (4). (10p)

6) Provide a short video for each step of the assignments and share its link in the end of the report. (20p)

For further questions, please mail me at ozdemirayk@itu.edu.tr.

Best wishes,

Aykut