ROS-Moveit-Gazebo

Controlling UR5 Simulated Robot

Objectives / Goals

- To build a custom simulation environment with UR5 Robot
  - To learn how to Mount a UR5 Robot Model on a pedestal or a table (include other accessories such as camera / Gripper etc.)
  - Include other objects into your world environment.
- To learn how to control the simulated robot in Gazebo using Moveit.
- Write Python Client or Node files to automatically send control commands.
- Carry out a complete autonomous motion planning by using visual feedback.
- Finally, using it for generating data required for training a deep reinforcement learning model.

Part I: Getting UR5 Simulation Model to work with Moveit Planner.

Tutorials / Weblinks for more information and Help

- Moveit Tutorial: https://ros-planning.github.io/moveit_tutorials/
- Universal Robot Repository: https://github.com/ros-industrial/universal_robot
- Pick & Place Example: https://github.com/lihuang3/ur5_ROS-Gazebo
- XML URDF Link: https://wiki.ros.org/urdf/XML/link
- ROS Robot Control: https://sir.upc.edu/projects/rostutorials/10-gazebo_control_tutorial/index.html
- UR5 Gazebo Robot Control using Moveit: https://www.theconstructsim.com/control-gazebo-simulated-robot-moveit-video-answer/
- Moveit RVIZ Quickstart Tutorial: http://docs.ros.org/kinetic/api/moveit_tutorials/html/doc/quickstart_in_rviz/quickstart_in_rviz_tutorial.html

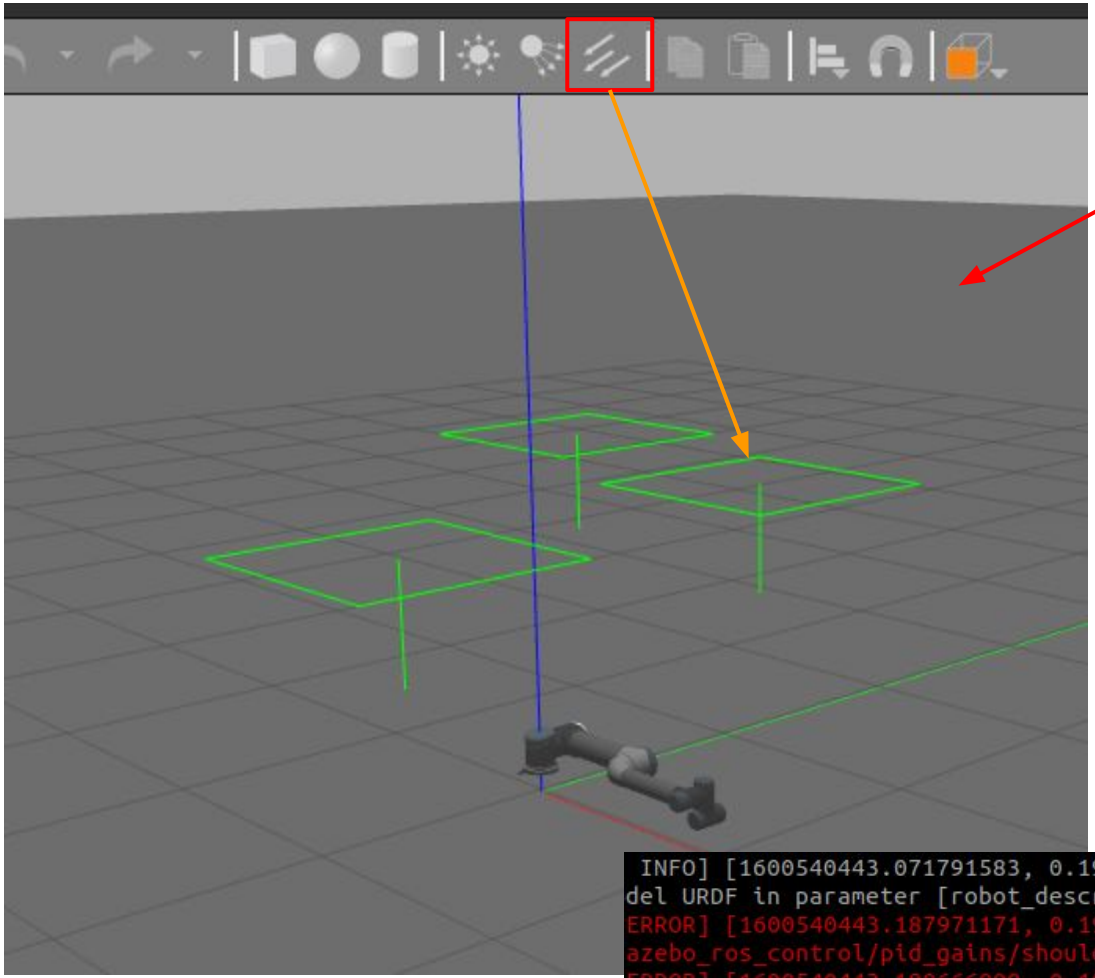
- The codes are tested with ROS-Melodic on Ubuntu 18.04

Step 1: Install Universal Robot Repository

```
$ cd catkin_ws/src
$ git clone -b melodic-devel
https://github.com/ros-industrial/universal_robot.git
$ catkin_make
$ source devel/setup.bash
$ roslaunch ur_gazebo ur5.launch
```

It shows some error on the terminal regarding something “No p gain specified for pid” which is to be ignored.

If you face problem with the building of repository, follow the instructions more closely given on the above repository page.



I add some directional lights for better visibility of the robot arm which otherwise appears very dark.

Ignore the error that you see on the terminal.

```
 INFO] [1600540443.071791583, 0.192000000]: gazebo_ros_control plugin is waiting for m
del URDF in parameter [robot_description] on the ROS param server.
ERROR] [1600540443.187971171, 0.192000000]: No p gain specified for pid.  Namespace: /
azebo_ros_control/pid_gains/shoulder_pan_joint
ERROR] [1600540443.188666999, 0.192000000]: No p gain specified for pid.  Namespace: /
azebo_ros_control/pid_gains/shoulder_lift_joint
```

Step 2: Build your own “Robot Description”

This is where you can decide to mount your UR5 robot on a pedestal or table or attach a gripper or a camera etc.

First, we will only mount the robot arm on a pedestal. Later we will include other components into our world.

So create a catkin package called “`myur5_description`” inside your `catkin_ws/src` folder.

```
$ cd catkin_ws/src
$ mkdir myur5sim
$ catkin_create_pkg myur5_description
$ cd myur5_description
$ mkdir urdf
```

- Create `myur5.urdf.xacro` file inside '`urdf`' folder that provides description of your pedestal with necessary joint and link information.

- Look into the following files for getting a better understanding of what to include in your URDF file:

```
~/catkin_ws/src/universal_robot/ur_gazebo/launch/ur5.launch
~/catkin_ws/src/universal_robot/ur_description/launch/ur5_upload.launch
~/catkin_ws/src/universal_robot/ur_description/urdf/ur5_robot.urdf.xacro
~/catkin_ws/src/universal_robot/ur_description/urdf/ur5.urdf.xacro
```

- Basically, we need to include UR5 URDF model and Gazebo related files in our own urdf file.

```xml
<?xml version="1.0"?>
<robot xmlns:xacro="http://wiki.ros.org/xacro" name="myur5">
  <link name="world"/>
  <link name="pedestal">
    <inertial>
      <origin xyz="0 0 0.5" rpy="0 0 0"/>
      <mass value="20"/>
      <inertia ixx="200" ixy="200" ixz="200" iyy="200" iyz="200" izz="200"/>
    </inertial>
    <visual>
      <origin xyz="0 0 0.5" rpy="0 0 0"/>
      <geometry>
        <cylinder radius="0.1" length="1"/>
      </geometry>
      <material name="Orange">
        <color rgba="${255/255} ${108/255} ${10/255} 1.0"/>
      </material>
    </visual>
    <collision>
      <origin xyz="0 0 0.5" rpy="0 0 0"/>
      <geometry>
        <cylinder radius="0.1" length="1"/>
      </geometry>
    </collision>
  </link>
```

Pedestal is just a cylinder of radius 0.1m and height 1m.

See the XML URDF link page on ROS wiki for more information on how to create such link.

File: ~/catkin_ws/src/myur5sim/myur5_description/urdf/myur5.urdf.xacro

File: ~/catkin_ws/src/myur5sim/myur5_description/urdf/myur5.urdf.xacro

```
26   <gazebo reference="pedestal">
27     <mu1>0.2</mu1>
28     <mu2>0.2</mu2>
29     <material>Gazebo/Orange</material>
30   </gazebo>
31   <joint name="world_joint" type="fixed">
32     <parent link="world" />
33     <child link="pedestal" />
34     <origin xyz="0 0 0" rpy="0.0 0.0 0.0"/>
35   </joint>
36   <xacro:arg name="transmission_hw_interface" default="hardware_interface/PositionJointInterface"/>
37   <!-- common stuff -->
38   <xacro:include filename="$(find ur_description)/urdf/common.gazebo.xacro" />
39   <!-- ur5 -->
40   <xacro:include filename="$(find ur_description)/urdf/ur5.urdf.xacro" />
41   <!-- arm -->
42   <xacro:ur5_robot prefix="" joint_limited="true"
43     transmission_hw_interface="$(arg transmission_hw_interface)"  />
44
45   <joint name="base_joint" type="fixed">
46     <parent link="pedestal" />
47     <child link="base_link" />
48     <origin xyz="0 0 1" rpy="0.0 0.0 0.0"/>
49   </joint>
50 </robot>
```

Gazebo related Information

Define joint between world and the pedestal.

These informations are required for loading a functional UR5 model into your environment.

Define joint between pedestal and robot base_link

Step 3: Create a Launch File to load the robot into Gazebo

- Create a file called "`myur5.launch`" inside the folder `myur5_description/launch` package.

- Please see the following files to have a better understanding of what to include in this file:
  `~/catkin_ws/src/universal_robot/ur_gazebo/launch/ur5.launch`

- Basically, we load an *empty world* file. Spawn the robot model provided through the variable "*robot_description*" and load necessary *controller files* to get the gazebo model working.

- Now you can load the gazebo model using the following command:
  `$ roslaunch myur5_description myur5.launch`

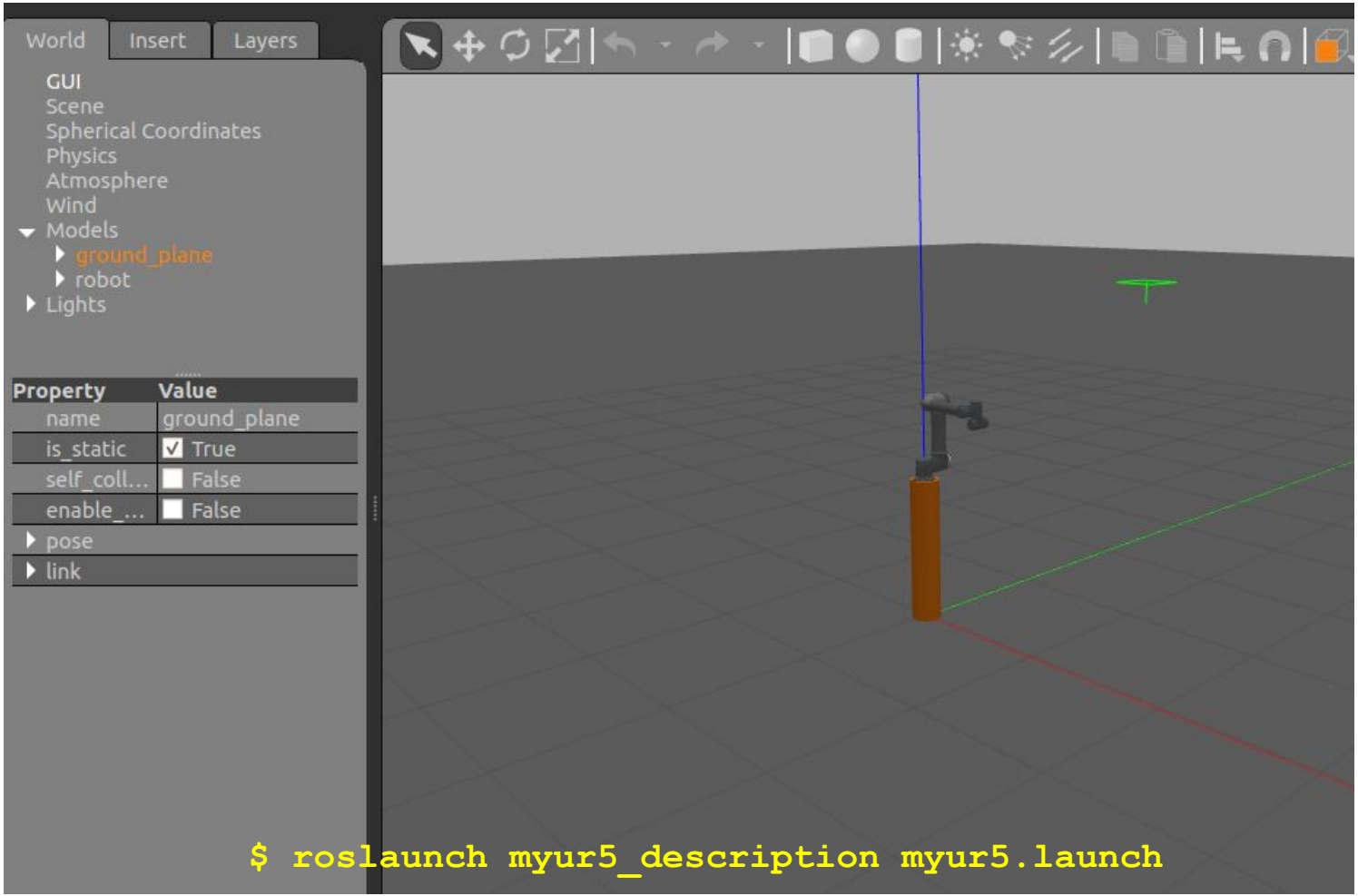
File: ~/catkin_ws/src/myur5sim/myur5_description/launch/myur5.launch

```
1  <?xml version="1.0"?>
2  <launch>
3    <arg name="limited" default="true"  doc="If true, limits joint range [-PI, PI] on all joints." />
4    <arg name="paused" default="false" doc="Starts gazebo in paused mode" />
5    <arg name="gui" default="true" doc="Starts gazebo gui" />
6    <arg name="transmission_hw_interface" default="hardware_interface/PositionJointInterface" />
7
8    <!-- startup simulated world -->
9    <include file="$(find gazebo_ros)/launch/empty_world.launch">
10     <arg name="world_name" default="worlds/empty.world"/>
11     <arg name="paused" value="$(arg paused)"/>
12     <arg name="gui" value="$(arg gui)"/>
13   </include>
14
15   <param name="robot_description" command="$(find xacro)/xacro '$(find myur5_description)/urdf/myur5.urdf.xacro'
   transmission_hw_interface:=$(arg transmission_hw_interface)" />
16
17   <!-- push robot_description to factory and spawn robot in gazebo -->
18   <node name="spawn_gazebo_model" pkg="gazebo_ros" type="spawn_model" args="-urdf -param robot_description -model robot -z 0.0"
   respawn="false" output="screen" />
19
20   <include file="$(find ur_gazebo)/launch/controller_utils.launch"/>
21
22   <!-- start this controller -->
23   <rosparam file="$(find ur_gazebo)/controller/arm_controller_ur5.yaml" command="load"/>
24   <node name="arm_controller_spawner" pkg="controller_manager" type="controller_manager" args="spawn arm_controller"
   respawn="false" output="screen"/>
25
26 </launch>
```

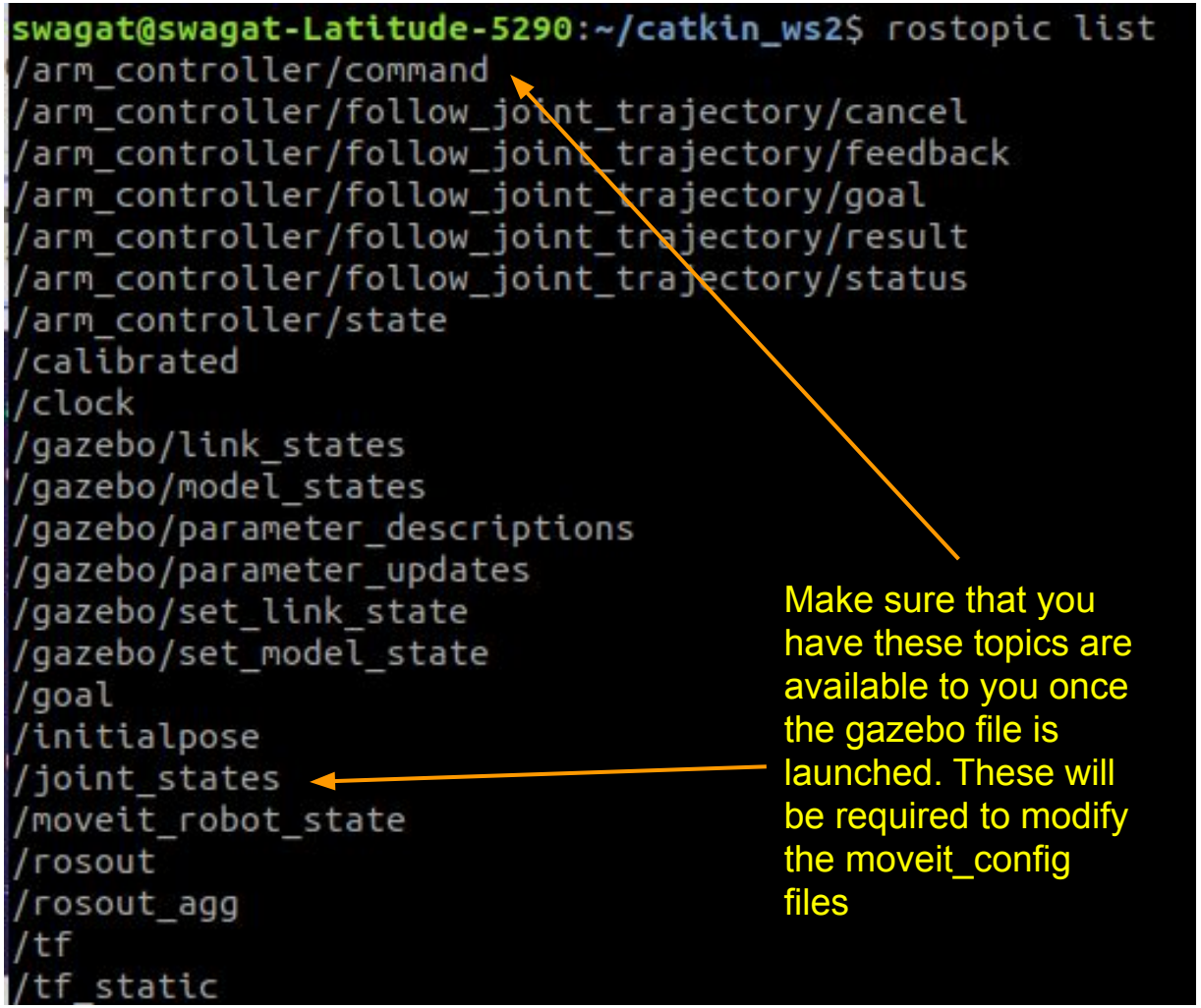
Load gazebo with empty world

Load the robot model into Gazebo

Load controllers



It still gives error saying “no p gain specified for pid” on the terminal. You can safely ignore this error.


swagat@swagat-Latitude-5290:~/catkin_ws2$ rostopic list
/arm_controller/command
/arm_controller/follow_joint_trajectory/cancel
/arm_controller/follow_joint_trajectory/feedback
/arm_controller/follow_joint_trajectory/goal
/arm_controller/follow_joint_trajectory/result
/arm_controller/follow_joint_trajectory/status
/arm_controller/state
/calibrated
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/goal
/initialpose
/joint_states
/moveit_robot_state
/rosout
/rosout_agg
/tf
/tf_static
Make sure that you have these topics are available to you once the gazebo file is launched. These will be required to modify the moveit_config files

Step 4: Create a Moveit_Config Package

- Follow the instructions available at this link (watch the video) to create a moveit package for the robot.
- This basically involves using `moveit_setup_assistant` to create a package called "`myur5_moveit_config`" inside the `~/catkin_ws/src/myur5sim/` folder.
- While defining the planning group, create a kinematic chain from pedestal to tool0.
- Create / Modify the following files as per the instruction provided in the above video:

```
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/controllers.yaml
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/joint_names.yaml
~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_moveit_controller_manager.launch.xml
~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_planning_execution.launch
```

Define Virtual Joints

Create a virtual joint between a robot link and an external frame of reference (considered fixed with respect to the robot).

	Virtual Joint Name	Child Link	Parent Frame	Type
1	FixedBase	pedestal	world	fixed
2	RobotBase	base_link	pedestal	fixed

Main settings on
moveit_setup_assistant manager

```
$ roslaunch
moveit_setup_assistant
setup_assistant.launch
```

Define Planning Groups

Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of (joint, link) pairs considered for planning and collision checking. Define individual groups for each subset of the robot you want to plan for.Note: when adding a link to the group, its parent joint is added too and vice versa.

Current Groups

- **manipulator**
  - *Joints*
  - *Links*
  - *Chain*
    - pedestal -> tool0
  - *Subgroups*

Define Robot Poses

Create poses for the robot. Poses are defined as planning groups. This is useful for things like *hor* each robot will be its initial pose in simulation.

	Pose Name	Group Name
1	AllZeros	manipulator
2	HomePose	manipulator

File:
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/controllers.yaml

```yaml
controller_list:
  - name: arm_controller
    action_ns: "follow_joint_trajectory"
    type: FollowJointTrajectory
    joints:
      - shoulder_pan_joint
      - shoulder_lift_joint
      - elbow_joint
      - wrist_1_joint
      - wrist_2_joint
      - wrist_3_joint

```

These are taken from “rostopic list” output while Gazebo is running.

This information is available in ur5.urdf.xacro file present in the universal_robot package.

File:
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/joint_names.yaml

```yaml
controller_joint_names: [shoulder_pan_joint,
                         shoulder_lift_joint,
                         elbow_joint,
                         wrist_1_joint,
                         wrist_2_joint,
                         wrist_3_joint]
```

The joint names could also be provided in the form of a list as mentioned in the video.

Modify the following File:

~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_moveit_controller_manager.launch.xml

```
<launch>

  <!-- loads moveit_controller_manager on the parameter server which is taken as argument
    if no argument is passed, moveit_simple_controller_manager will be set -->
  <arg name="moveit_controller_manager" default="moveit_simple_controller_manager/MoveItSimpleControllerManager" />
  <param name="moveit_controller_manager" value="$(arg moveit_controller_manager)"/>
  <param name="use_controller_manager" value="false"/>
  <param name="trajectory_execution/execution_duration_monitoring" value="false"/>

  <!-- loads ros_controllers to the param server -->
  <!--rosparam file="$(find myur5_moveit_config)/config/ros_controllers.yaml"/-->
  <rosparam file="$(find myur5_moveit_config)/config/controllers.yaml"/>
</launch>
```

We include the “controllers.yaml” in this file. Probably, it is also possible to modify the ros_controllers.yaml file as well.

Create the following File:
~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_planning_execution.launch

```
<?xml version="1.0"?>
<launch>
  <rosparam command="load" file="$(find myur5_moveit_config)/config/joint_names.yaml" />
  <include file="$(find myur5_moveit_config)/launch/planning_context.launch" >
    <arg name="load_robot_description" value="true" />
  </include>

  <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
    <param name="/use_gui" value="false"/>
    <rosparam param="/source_list">[/joint_states]</rosparam>
  </node>

  <include file="$(find myur5_moveit_config)/launch/move_group.launch">
    <arg name="publish_monitored_planning_scene" value="true"/>
  </include>

  <include file="$(find myur5_moveit_config)/launch/moveit_rviz.launch">
    <!--arg name="config" value="true"/--> <!-- this gives error -->
  </include-->
</launch>
```

Notice these components.

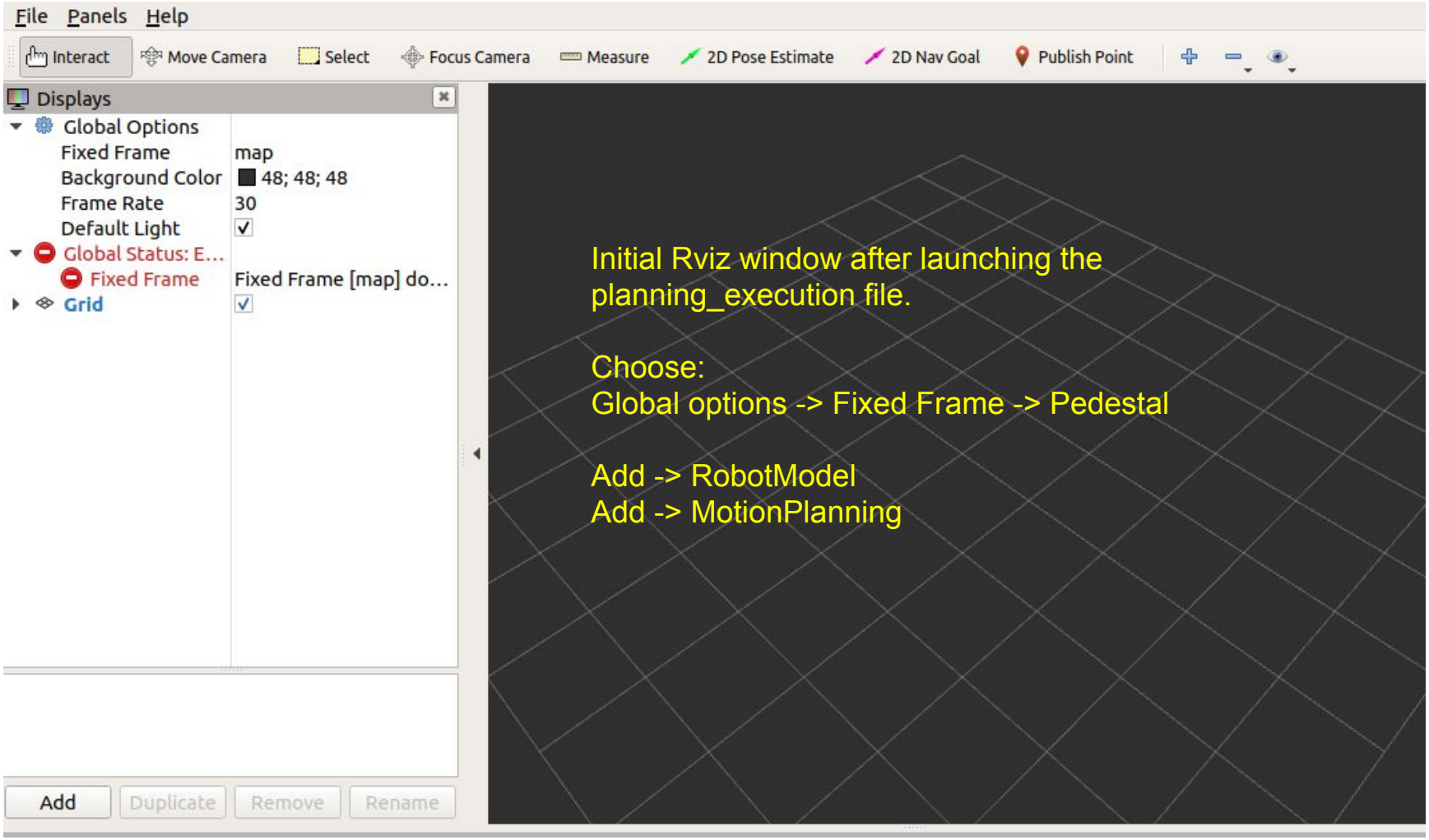
- Run the following command on a separate terminal while the Gazebo is running.

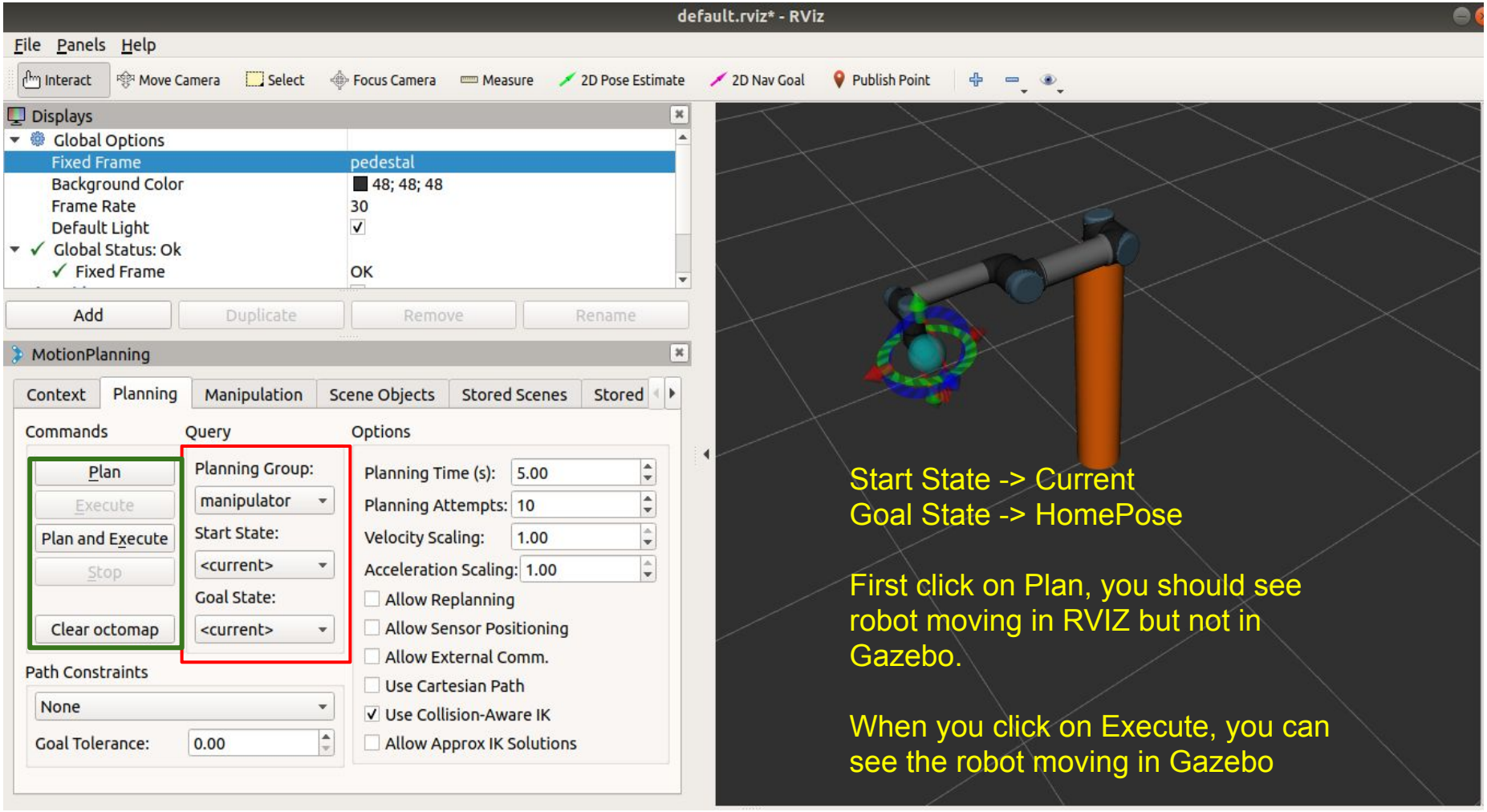
```
$ roslaunch myur5_moveit_config myur5_planning_execution.launch
```

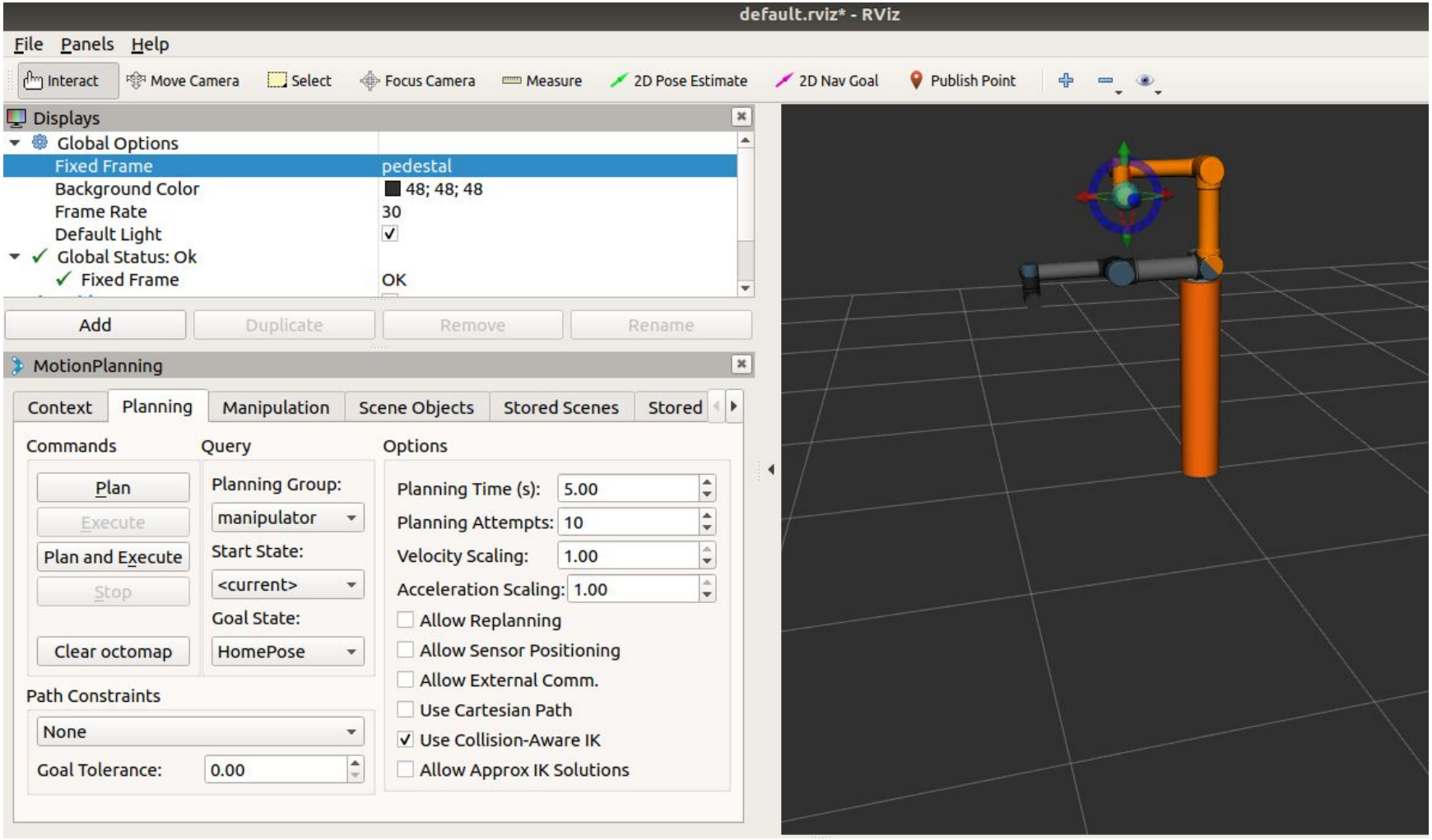
It might give an error as follows, which can be ignored:

```
[ERROR] [1600601614.145766164, 3039.711000000]: Could not find the planner configuration 'None' on the param server
```

- Now you should be able to plan motion in Rviz using *MotionPlanner* tool and execute them on the Gazebo Robot.
- Go through the Moveit RVIZ tutorial to know more about this topic.
- You can save the modified RVIZ environment (after including robot and tools) as a separate file “moveit.rviz” file.


File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
map
Background Color
48; 48; 48
Frame Rate
30
Default Light
Global Status: E...
Fixed Frame
Fixed Frame [map] do...
Grid
Add
Duplicate
Remove
Rename
Initial Rviz window after launching the planning_execution file.
Choose:
Global options -> Fixed Frame -> Pedestal
Add -> RobotModel
Add -> MotionPlanning


default.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
pedestal
Background Color
48; 48; 48
Frame Rate
30
Default Light
Global Status: Ok
Fixed Frame
OK
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Manipulation
Scene Objects
Stored Scenes
Stored
Commands
Plan
Execute
Plan and Execute
Stop
Clear octomap
Query
Planning Group:
manipulator
Start State:
<current>
Goal State:
<current>
Options
Planning Time (s):
5.00
Planning Attempts:
10
Velocity Scaling:
1.00
Acceleration Scaling:
1.00
Allow Replanning
Allow Sensor Positioning
Allow External Comm.
Use Cartesian Path
Use Collision-Aware IK
Allow Approx IK Solutions
Path Constraints
None
Goal Tolerance:
0.00
Start State -> Current
Goal State -> HomePose
First click on Plan, you should see robot moving in RVIZ but not in Gazebo.
When you click on Execute, you can see the robot moving in Gazebo


default.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
pedestal
Background Color
48; 48; 48
Frame Rate
30
Default Light
Global Status: Ok
Fixed Frame
OK
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Manipulation
Scene Objects
Stored Scenes
Stored
Commands
Plan
Execute
Plan and Execute
Stop
Clear octomap
Query
Planning Group:
manipulator
Start State:
<current>
Goal State:
HomePose
Options
Planning Time (s):
5.00
Planning Attempts:
10
Velocity Scaling:
1.00
Acceleration Scaling:
1.00
Allow Replanning
Allow Sensor Positioning
Allow External Comm.
Use Cartesian Path
Use Collision-Aware IK
Allow Approx IK Solutions
Path Constraints
None
Goal Tolerance:
0.00

Problem:

```
[ INFO] [1600424548.415571735, 0.193000000]: gazebo_ros_control plugin is waiting for m
odel URDF in parameter [robot_description] on the ROS param server.
[ERROR] [1600424548.539552398, 0.193000000]: No p gain specified for pid.  Namespace: /
gazebo_ros_control/pid_gains/shoulder_pan_joint
[ERROR] [1600424548.540490124, 0.193000000]: No p gain specified for pid.  Namespace: /
gazebo_ros_control/pid_gains/shoulder_lift_joint
[ERROR] [1600424548.541040967, 0.193000000]: No p gain specified for pid.  Namespace: /
gazebo_ros_control/pid_gains/elbow_joint
```

Solution: Add the following lines to the file:
`universal_robot/ur_gazebo/controller/arm_controller_ur5.yaml`

```
33 gazebo_ros_control:
34   pid_gains:
35     shoulder_pan_joint: {p: 100.0, i: 0.01, d: 1.0}
36     shoulder_lift_joint: {p: 100.0, i: 0.01, d: 1.0}
37     elbow_joint: {p: 100.0, i: 0.01, d: 1.0}
38     wrist_1_joint: {p: 100.0, i: 0.01, d: 1.0}
39     wrist_2_joint: {p: 100.0, i: 0.01, d: 1.0}
40     wrist_3_joint: {p: 100.0, i: 0.01, d: 1.0}
```

This prevents the above error messages. However, it does not work.

So, Don't do this.

Summary for Part 1

- We saw how to add a “pedestal” to our UR5 robot. This paves the way to include other components into our Gazebo simulation which will be demonstrated in the next part.

- Use Moveit to control robot motion in the Gazebo Simulation.

- To do list:
  - How to use Gazebo’s model editor to build models and include them in the simulation?
  - Include a mesh file for the robot pedestal which will be visible in RVIZ.