

Robotics Lab I Report

Utkrist Singh Karky and Manish Thapa

Prof. Dr. Francesco Maurelli

TA: Assylbek Sakenov

November 2022

1. Launch `uuv_gazebo/rexrov.default.launch` and inspect the nodes, topics, services it launches and the message/service types they use to communicate.
 - (a) Write a short report describing these nodes, topics, services and messages including screenshots of `rqt` or the terminal output from which you collected this information. (15 pts)

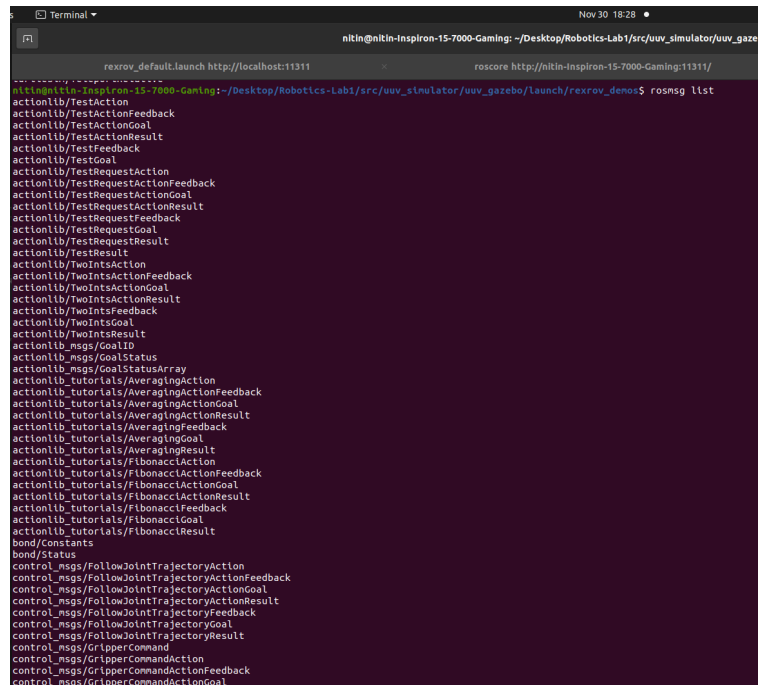
- Answer:

After we launch the `rexrov.default.launch` file, we can view the `rviz`. We can view the nodes and topics of the file that has launched from the `rqt` and `rqt_graph`(view graph) or can view all the nodes, topics, messages and services from the terminal.

If we view the `rqt_graph` to see the nodes and topics that are generated, we can see we have eight nodes which are in a spherical container. These nodes are for different purposes and functionalities. For eg. `/rexrov/urdf_spawner` is to spawn the robot as soon as we launch it and `/rexrov/joystick` is for the control of the robot. Nodes here in the graph can be seen connected but are passing through the rectangular boxes. These boxes contain the 'topics'. Nodes are communicating with the exchange of topics which basically is information. For eg. the node '`rexrov/joy_uuv_velocity_teleop`' is subscribing to the topic called '`/rexrov/joy`' and data(messages) transmitted by the node '`/rexrov/joystick`' is received by the subscribing topic.

Here the messages come in as well. There are many ros messages generated while this file is launching. We can view it in the example screenshot below as well. These messages for example `visualization_msgs/ImageMaker`, `turtlesim/color`, `tf/tfMessage` etc. are published by the nodes. These messages have descriptions about the node we have. We have services, generally defined by 'srv' files(screenshots as example attached). Some services like

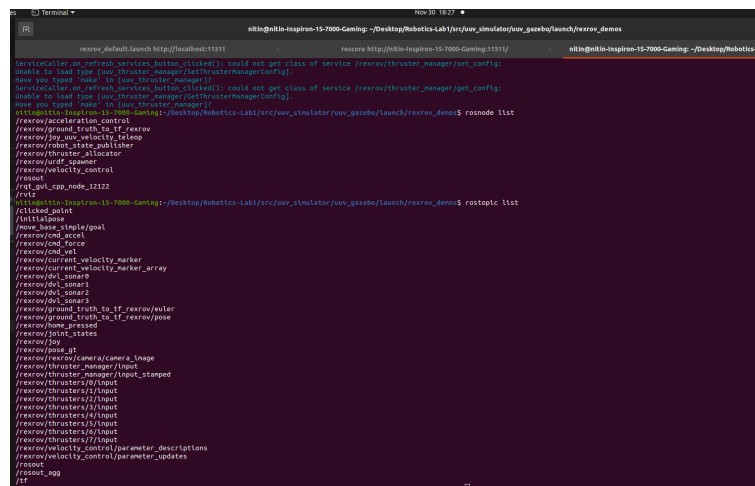
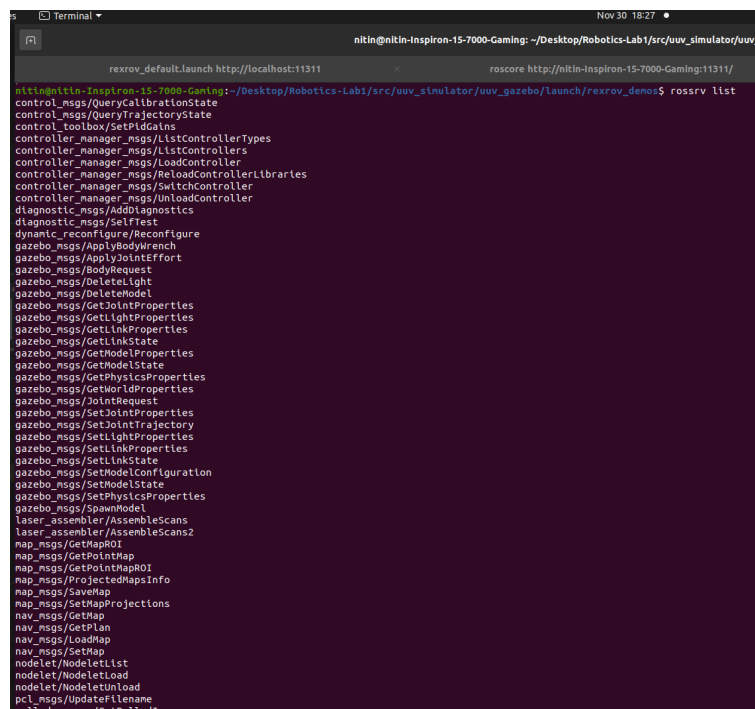
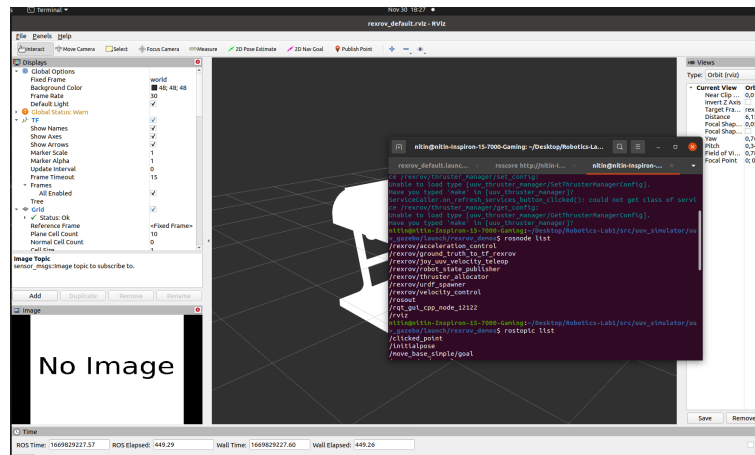
`gazebo_msgs/DeleteLight` and `control_msgs/QueryTrajectoryState` are used here to request and reply provided by nodes.

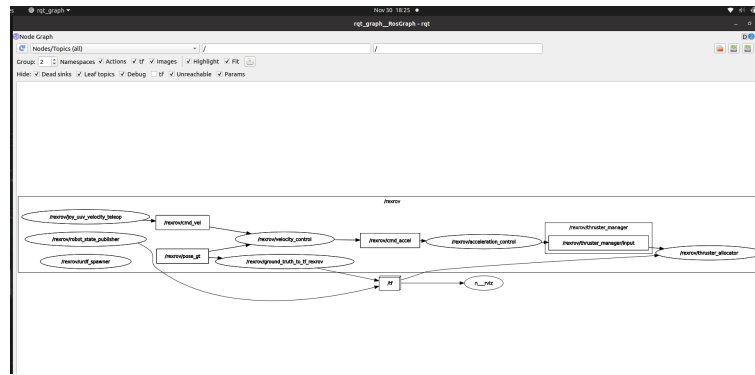


```

Terminal
nitin@nitin-Inspiron-15-7000-Gaming: ~/Desktop/Robotics-Lab1/src/uuv_simulator/uuv_gazebo
rexrov_default.launch http://localhost:11311
roscmp http://nitin-Inspiron-15-7000-Gaming:11311/
nitin@nitin-Inspiron-15-7000-Gaming:~/Desktop/Robotics-Lab1/src/uuv_simulator/uuv_gazebo/launch/rexrov_demos$ roscmp list
actionlib/TestAction
actionlib/TestActionFeedback
actionlib/TestActionGoal
actionlib/TestActionResult
actionlib/TestFeedback
actionlib/TestGoal
actionlib/TestRequestAction
actionlib/TestRequestActionFeedback
actionlib/TestRequestActionGoal
actionlib/TestRequestActionResult
actionlib/TestRequestFeedback
actionlib/TestRequestGoal
actionlib/TestRequestResult
actionlib/TestResult
actionlib/TwoIntsAction
actionlib/TwoIntsActionFeedback
actionlib/TwoIntsActionGoal
actionlib/TwoIntsActionResult
actionlib/TwoIntsFeedback
actionlib/TwoIntsGoal
actionlib/TwoIntsResult
actionlib_msgs/GoalID
actionlib_msgs/GoalStatus
actionlib_msgs/GoalStatusArray
actionlib_tutorials/AveragingAction
actionlib_tutorials/AveragingActionFeedback
actionlib_tutorials/AveragingActionGoal
actionlib_tutorials/AveragingActionResult
actionlib_tutorials/AveragingFeedback
actionlib_tutorials/AveragingGoal
actionlib_tutorials/AveragingResult
actionlib_tutorials/FibonacciAction
actionlib_tutorials/FibonacciActionFeedback
actionlib_tutorials/FibonacciActionGoal
actionlib_tutorials/FibonacciActionResult
actionlib_tutorials/FibonacciFeedback
actionlib_tutorials/FibonacciGoal
actionlib_tutorials/FibonacciResult
bond/Constants
bond/Status
control_msgs/FollowJointTrajectoryAction
control_msgs/FollowJointTrajectoryActionFeedback
control_msgs/FollowJointTrajectoryActionGoal
control_msgs/FollowJointTrajectoryActionResult
control_msgs/FollowJointTrajectoryFeedback
control_msgs/FollowJointTrajectoryGoal
control_msgs/FollowJointTrajectoryResult
control_msgs/GripperCommand
control_msgs/GripperCommandAction
control_msgs/GripperCommandActionFeedback
control_msgs/GripperCommandActionResult

```





- (b) Include the commands you used to retrieve this information and write down what these commands do. (10 pts)

- Answer:

Following are the command lines in order after we have git cloned the uuv_simulator package in our workspace:

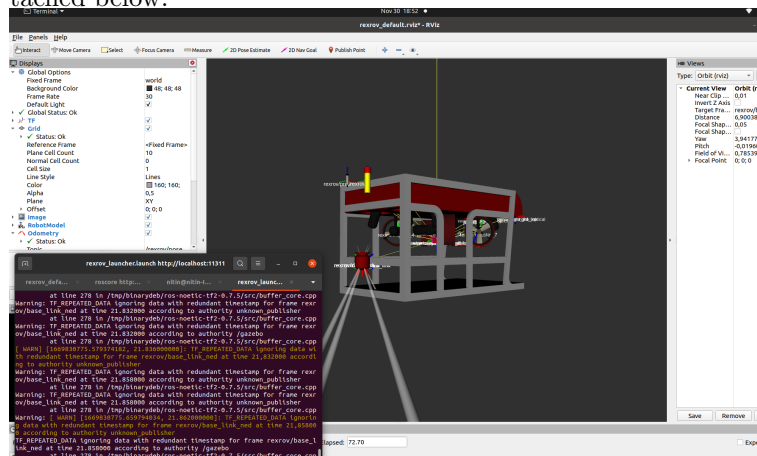
- i. **catkin_make**
Here, we built the packages using the ‘catkin_make’ command.
- ii. **source devel/setup.bash**
After we build the packages, we need to make sure the ros path is active in the terminal we are working on and for that we source using the ‘source devel/setup.bash’ command.
- iii. **roscore**
We open a new terminal and run the ‘roscore’ command. We make sure run this before we running any commands as it enables communication
- iv. **roslaunch rextrover_default.launch**
Finally, we get into the directory where the rextrover_default.launch file is located and launch the file to view it in rviz using the ‘roslaunch rextrover_default.launch’ command line.
- v. **rqt**
We obtain the rqt screen where we can view all the nodes, topics, services and messages and msg/srv types. We just type ‘rqt’.
- vi. **rqt_graph**
We can view the detailed graph of launched file with ‘rqt_graph’ command. We got to see nodes, tf and topics and how they were connected.
- vii. **rostopic list**
We can view all the nodes in the launch file while its running from the command using this command line.
- viii. **rostopic list**
View topic directly from the command line
- ix. **rostopic list**
View the messages involved in the launch
- x. **rostopic list**
View the services running while the file launches

2. Controlling the robot using teleop_twist_keyboard:

- (a) Write a launch file that launches the rextrov in `uuv_gazebo/rextrov_default.launch` into the world spawned by `uuv_gazebo_worlds/empty_underwater_world.launch` and the `teleop_twist_keyboard` node in such a way that you can use the teleop node to control the simulated ROV. Please include comments in this launch file to explain what each line is doing. (15 pts)

- Answer:

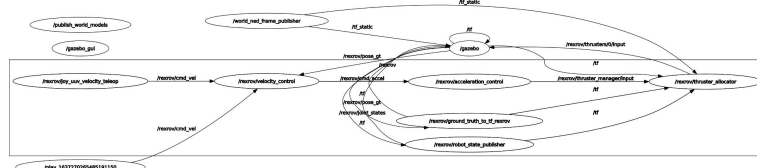
The launch file is `uuv_simulator/rov/launch/rexrov_launcher.launch`. Also screenshot of rviz attached below:

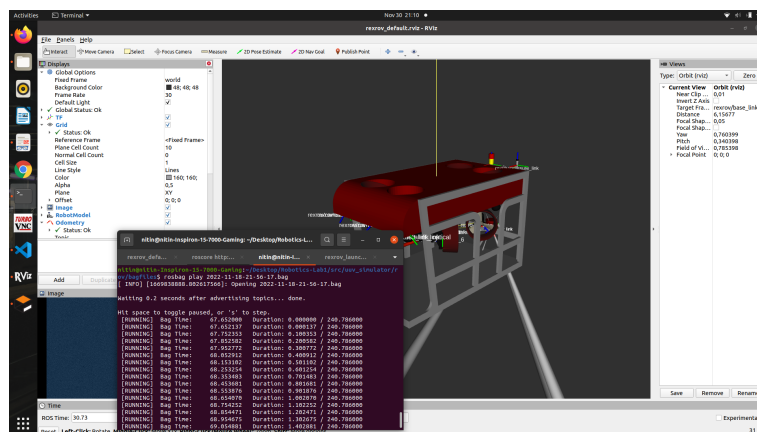


- (b) Log the topic `teleop_twist_keyboard/cmd_vel` using `roscat`, move the vehicle around with your key-board while `roscat` is recording. Stop `teleop_twist_keyboard` and control the robot by replaying the generated `roscat`. Create a plot of the traced trajectory of the ROV, and attach a screenshot of `rqt_graph` while the bag is playing (5 pts)

- Answer:

Here is the screenshot of the `rqt_graph` while the bag is playing. This picture location is in `uuv_simulator/rov/pictures/rtq_graph_play.png`





3. In this exercise you need to create a node that outputs forces and torques as input to control the AUV in `uuv_gazebo/rexrov_wrench_control.launch` based on this input. Please download this launch file from the page linked into `catkin_ws/src/uuv_simulator/uuv_gazebo/launch/rexrov_demos` and launch it into the world spawned by `uuv_gazebo_worlds/empty_underwater_world.launch`

- (a) Write a node similar to `teleop_twist_keyboard` that publishes forces and torques to control the AUV. Your node must publish to the topic `/rexrov/thruster_manager/input` (15pts)

- Answer:

The node `teleop_forces_torques.py` to publish forces and torques to control the AUV. The location is `uuv_simulator/rov/scripts/teleop_forces_torques.py`

- (b) Write a launch file similar to the one in exercise 2 that launches the AUV and your node. (5 pts)

- Answer:

The location is `uuv_simulator/rov/launch/AUV_launcher.launch`

4. Creating and controlling your own robot.

- (a) Create a urdf file that describes a simple ROV of your own design using the default geometric shapes (or design your own robot in Blender) (10 pts)

- Answer:

We have created an urdf file of our ROV model along with its launch file and have attached the file link here. The urdf model code is described in order below. The name of our robot is ‘ROV’. We have 3 links(body) and 2 joints to connect them. The description of every link and joints are under their respective tags. The position, geometry and inertial attributes of links are in the respective tags. The visual part under `visual` tag has the shapes or geometry of the links and joints. Mass and inertia for the link is under the `inertia` tag. For the geometry part, the shape of link1 which is our base link is spherical in shape while the other side links are boxes of given sizes. Links are given a color which is black under the `material` tag. Links have been given masses and inertia under the `inertia` tag. For joints, we have shown to which link is each of two joints connected to. The parent link to both joints is joint1, meaning that the other two links(link2 and link3) are child links for the joints. The `origin` and `axis` tag defines the location of either it be the links or joints.

- (b) Decide on the placement of thrusters for this ROV and give your reasoning. (5 pts)

- Answer:

So we will have eight thrusters in our ROV model to provide good controllability to our ROV. The four thrusters are placed in the base that helps in lateral movement. They are placed at an angle of 45 degrees to make the movement more effective and there is also no disturbance between the thrusters in between which can impact the movement of ROV as well. The other four thrusters are placed on top of ROV. This helps in the vertical movement of the ROV and the number of thrusters installed makes the movement powerful.

- (c) Write your own node that takes in forces and torques as input and publishes thrust commands for your ROV. The conversion from forces/torques to thrust commands has to be called as a service. (10 pts)

- Answer:

The location for the node file is `uuv_simulator/rov/scripts/thrusters.py`

- (d) Write a launch file that (10 pts)

i. load this robot into `uuv_gazebo_worlds/empty_underwater_world.launch` in gazebo

ii. starts the node you created in exercise 3 to publish force/torque data

iii. starts the node you created in part c that publishes thrust commands to your vehicle

- Answer:

The location for the launch file is `uuv_simulator/rov/launch/ROV_launcher.launch`. And we also had to modify `teleop_forces_torques` node for our ROV, so the location for this node is `uuv_simulator/rov/scripts/teleop_forces_torques_rov.py`

Sources:

1. https://github.com/my-name-is-D/uuv_simulator
2. <http://wiki.ros.org/>
3. <https://github.com/manishthapa99/RisLab1>