RIS LAB PROJECT

Group: Pratham Nimesh Shah, Haider Qaizar Hussain, Aryans Rathi

Professor: Dr Francesco Maurelli

1. Launch uuv_gazebo/rexrov_default.launch and inspect the nodes, topics, services it launches and the message/service types they use to communicate. (Contribution 33.3% each)

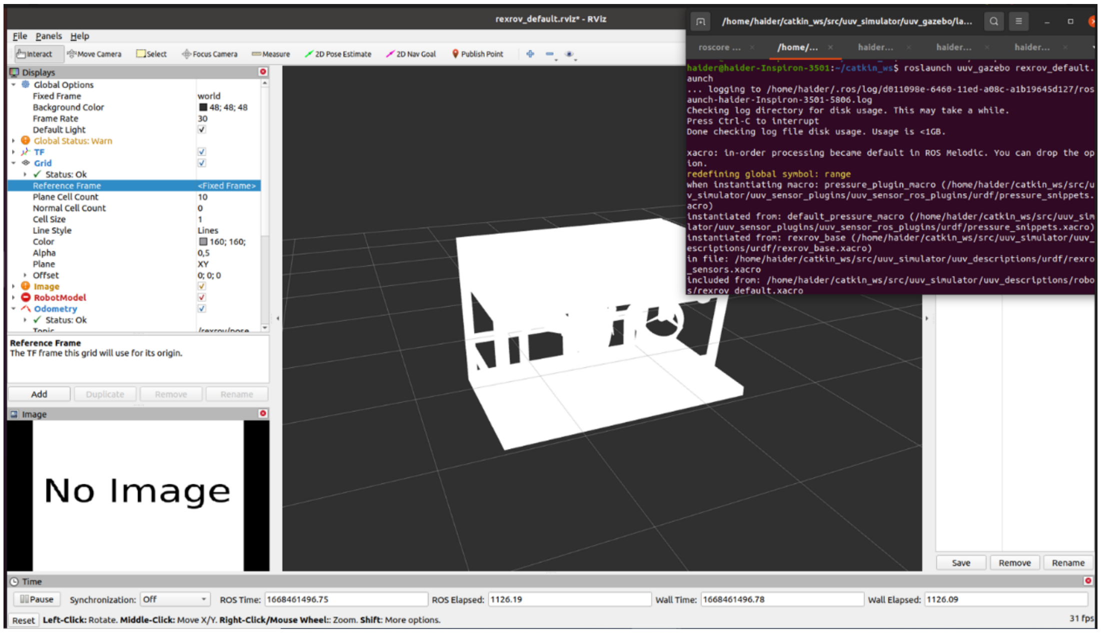
(a) Write a short report describing these nodes, topics, services, and messages including screenshots of rqt or the terminal output from which you collected this information. (15 pts)

When the rexrov_default.launch file is launched, you can see the rviz. You can see the nodes and topics of the file that has been launched from the rqt and rqt_graph or can view all the nodes, topics, messages, and services from the terminal.

If you observe the rqt_graph to see the nodes and topics that are generated, you 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 control_msgs/ QueryTrajectoryState are used here to request and reply provided by nodes

Shown below are screenshots of the terminal outputs after writing the command lines



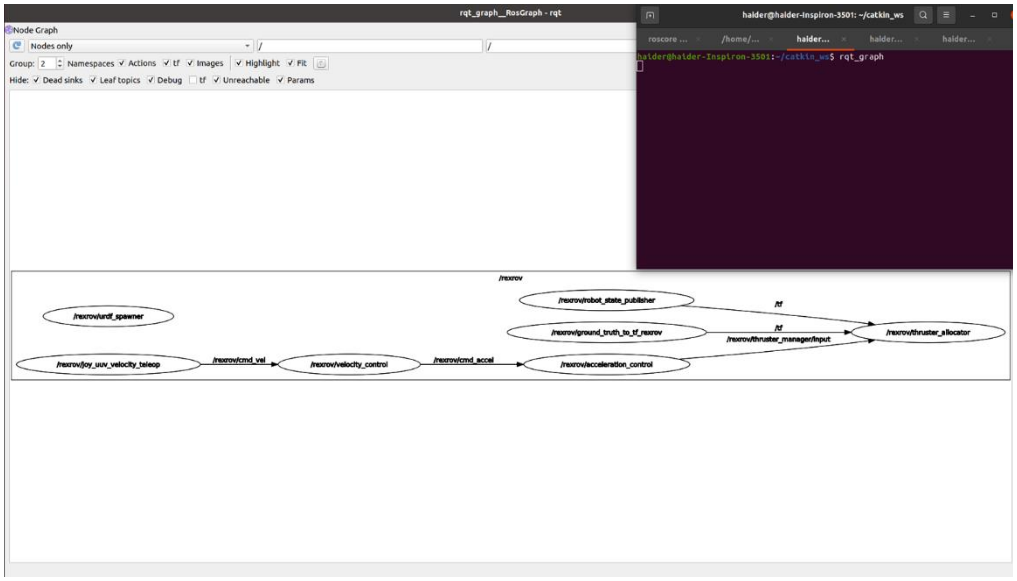
```
Global Options
  Fixed Frame          world
  Background Color     48; 48; 48
  Frame Rate           30
  Default Light        ✓
Global Status: Warn
TF                     ✓
Grid                   ✓
  Status: Ok
  Reference Frame      <Fixed Frame>
  Plane Cell Count     10
  Normal Cell Count    0
  Cell Size            1
  Line Style           Lines
  Color                160; 160;
  Alpha                0,5
  Plane                XY
  Offset               0; 0; 0
Image                  ✓
RobotModel             ✓
Odometry               ✓
  Status: Ok
  Topic                /rexrov/pose

Reference Frame
The TF frame this grid will use for its origin.
```

```
Synchronization: Off   ROS Time: 1668461496.75   ROS Elapsed: 1126.19   Wall Time: 1668461496.78   Wall Elapsed: 1126.09
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
```

```
haider@haider-Inspiron-3501:~/catkin_ws$ roslaunch uuv_gazebo rexrov_default.
aunch
... logging to /home/haider/.ros/log/d011098e-6460-11ed-a08c-a1b19645d127/ros
aunch-haider-Inspiron-3501-5806.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: in-order processing became default in ROS Melodic. You can drop the op
ion.
redefining global symbol: range
when instantiating macro: pressure_plugin_macro (/home/haider/catkin_ws/src/u
v_simulator/uuv_sensor_plugins/uuv_sensor_ros_plugins/urdf/pressure_snippets.
acro)
instantiated from: default_pressure_macro (/home/haider/catkin_ws/src/uuv_sim
lator/uuv_sensor_plugins/uuv_sensor_ros_plugins/urdf/pressure_snippets.xacro)
instantiated from: rexrov_base (/home/haider/catkin_ws/src/uuv_simulator/uuv_
escriptions/urdf/rexrov_base.xacro)
in file: /home/haider/catkin_ws/src/uuv_simulator/uuv_descriptions/urdf/rexro
_sensors.xacro
included from: /home/haider/catkin_ws/src/uuv_simulator/uuv_descriptions/robo
s/rexrov default.xacro
```

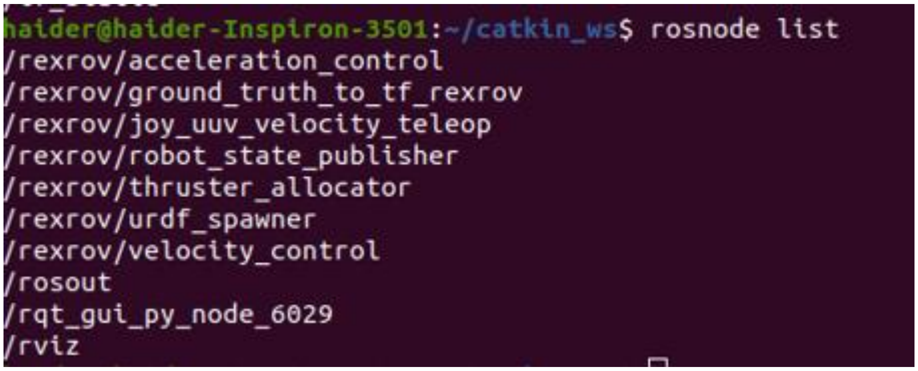

rqt_graph__RosGraph - rqt
Node Graph
haider@haider-Inspiron-3501: ~/catkin_ws
roscore ...
/home/...
haider...
haider...
haider...
/rexrov
/rexrov/robot_state_publisher
/tf
/rexrov/urdf_spawner
/rexrov/ground_truth_to_tf_rexrov
/tf
/rexrov/thruster_manager/input
/rexrov/thruster_allocator
/rexrov/joy_uuv_velocity_teleop
/rexrov/cmd_vel
/rexrov/velocity_control
/rexrov/cmd_accel
/rexrov/acceleration_control

Nodes only · / · /

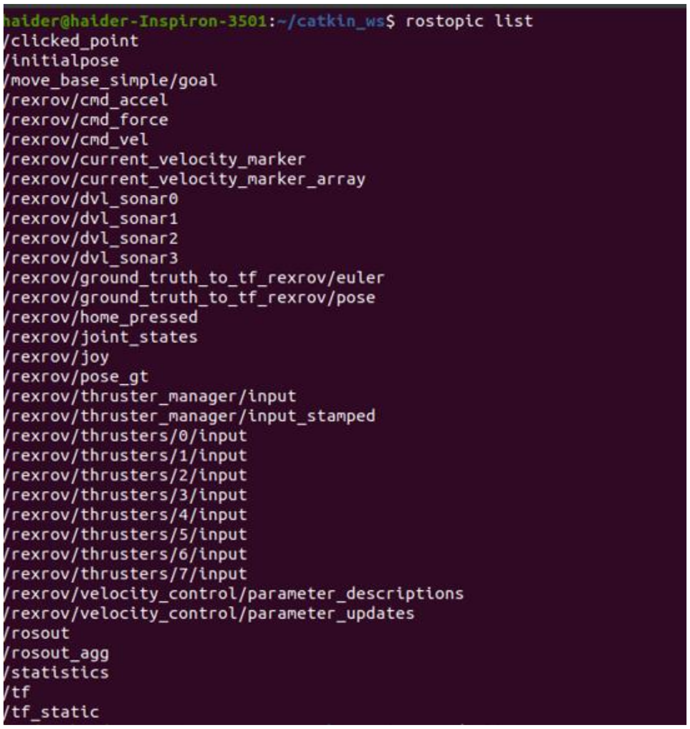
Group: 2 Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit

Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug □ tf ✓ Unreachable ✓ Params

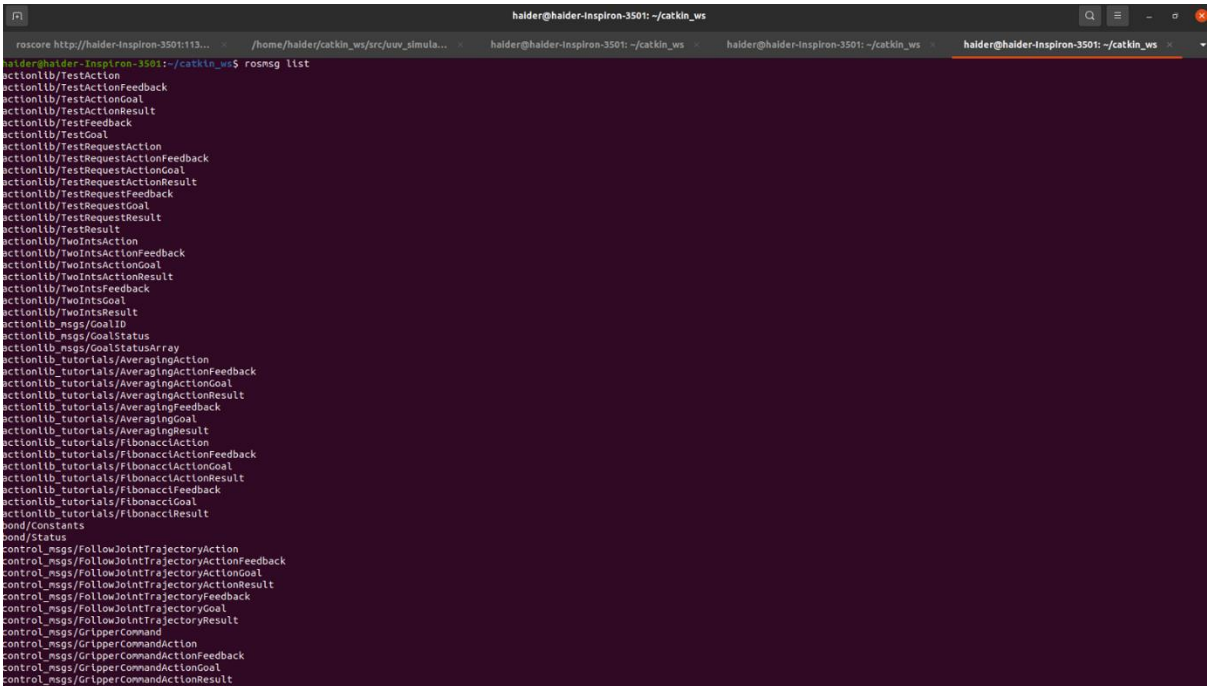
```
haider@haider-Inspiron-3501:~/catkin_ws$ rqt_graph
```



```
haider@haider-Inspiron-3501:~/catkin_ws$ rosnode list
/rexrov/acceleration_control
/rexrov/ground_truth_to_tf_rexrov
/rexrov/joy_uuv_velocity_teleop
/rexrov/robot_state_publisher
/rexrov/thruster_allocator
/rexrov/urdf_spawner
/rexrov/velocity_control
/rosout
/rqt_gui_py_node_6029
/rviz
```

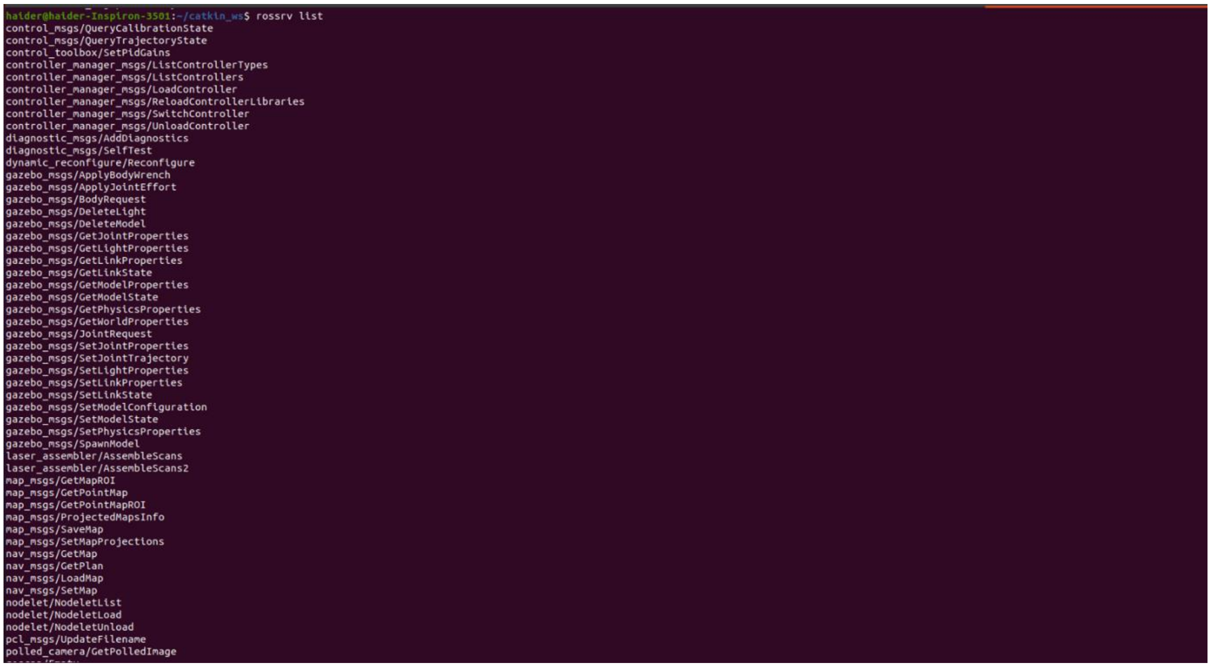


```
haider@haider-Inspiron-3501:~/catkin_ws$ rostopic list
/clicked_point
/initialpose
/move_base_simple/goal
/rexrov/cmd_accel
/rexrov/cmd_force
/rexrov/cmd_vel
/rexrov/current_velocity_marker
/rexrov/current_velocity_marker_array
/rexrov/dvl_sonar0
/rexrov/dvl_sonar1
/rexrov/dvl_sonar2
/rexrov/dvl_sonar3
/rexrov/ground_truth_to_tf_rexrov/euler
/rexrov/ground_truth_to_tf_rexrov/pose
/rexrov/home_pressed
/rexrov/joint_states
/rexrov/joy
/rexrov/pose_gt
/rexrov/thruster_manager/input
/rexrov/thruster_manager/input_stamped
/rexrov/thrusters/0/input
/rexrov/thrusters/1/input
/rexrov/thrusters/2/input
/rexrov/thrusters/3/input
/rexrov/thrusters/4/input
/rexrov/thrusters/5/input
/rexrov/thrusters/6/input
/rexrov/thrusters/7/input
/rexrov/velocity_control/parameter_descriptions
/rexrov/velocity_control/parameter_updates
/rosout
/rosout_agg
/statistics
/tf
/tf_static
```


haider@haider-Inspiron-3501: ~/catkin_ws
roscore http://haider-Inspiron-3501:113...
/home/haider/catkin_ws/src/uuv_simula...
haider@haider-Inspiron-3501: ~/catkin_ws
haider@haider-Inspiron-3501: ~/catkin_ws
haider@haider-Inspiron-3501: ~/catkin_ws

```
haider@haider-Inspiron-3501:~/catkin_ws$ rosmsg 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/GripperCommandActionGoal
control_msgs/GripperCommandActionResult
```

(b)



```
haider@haider-Inspiron-3501:~/catkin_ws$ rossrv list
control_msgs/QueryCalibrationState
control_msgs/QueryTrajectoryState
control_toolbox/SetPidGains
controller_manager_msgs/ListControllerTypes
controller_manager_msgs/ListControllers
controller_manager_msgs/LoadController
controller_manager_msgs/ReloadControllerLibraries
controller_manager_msgs/SwitchController
controller_manager_msgs/UnloadController
diagnostic_msgs/AddDiagnostics
diagnostic_msgs/SelfTest
dynamic_reconfigure/Reconfigure
gazebo_msgs/ApplyBodyWrench
gazebo_msgs/ApplyJointEffort
gazebo_msgs/BodyRequest
gazebo_msgs/DeleteLight
gazebo_msgs/DeleteModel
gazebo_msgs/GetJointProperties
gazebo_msgs/GetLightProperties
gazebo_msgs/GetLinkProperties
gazebo_msgs/GetLinkState
gazebo_msgs/GetModelProperties
gazebo_msgs/GetModelState
gazebo_msgs/GetPhysicsProperties
gazebo_msgs/GetWorldProperties
gazebo_msgs/JointRequest
gazebo_msgs/SetJointProperties
gazebo_msgs/SetJointTrajectory
gazebo_msgs/SetLightProperties
gazebo_msgs/SetLinkProperties
gazebo_msgs/SetLinkState
gazebo_msgs/SetModelConfiguration
gazebo_msgs/SetModelState
gazebo_msgs/SetPhysicsProperties
gazebo_msgs/SpawnModel
laser_assembler/AssembleScans
laser_assembler/AssembleScans2
map_msgs/GetMapROI
map_msgs/GetPointMap
map_msgs/GetPointMapROI
map_msgs/ProjectedMapsInfo
map_msgs/SaveMap
map_msgs/SetMapProjections
nav_msgs/GetMap
nav_msgs/GetPlan
nav_msgs/LoadMap
nav_msgs/SetMap
nodelet/NodeletList
nodelet/NodeletLoad
nodelet/NodeletUnload
pcl_msgs/UpdateFilename
polled_camera/GetPolledImage
```

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

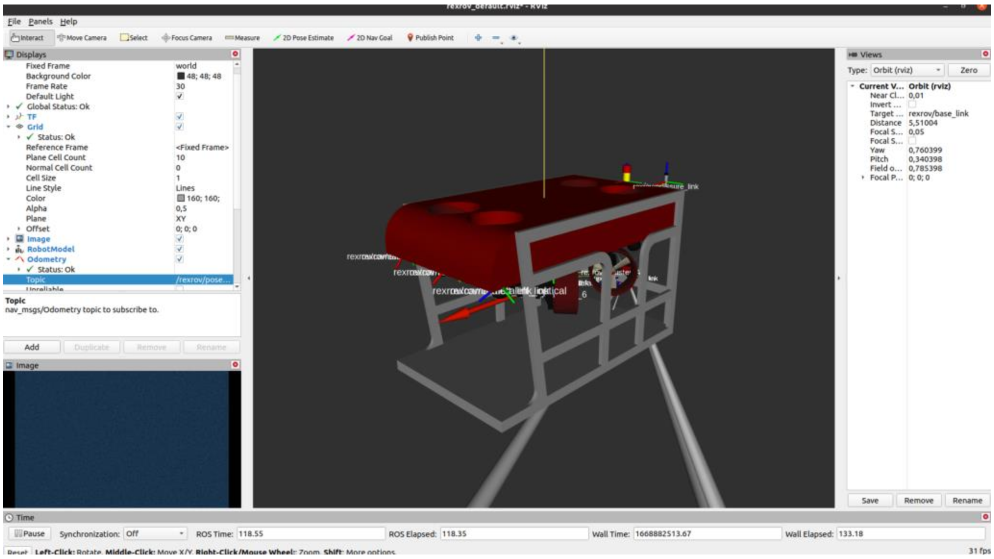
Following are the command lines in order after we have git cloned the uuv_simulator package in our workspace:

1.	catkin_make	Here, we built the packages using the 'catkin_make' command.
2.	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
3.	roscore	To open a new terminal and run the 'roscore' command. We make sure run this before we running any commands as it enables communication
4.	roslaunch uuv_gazebo rexrov_default.launch	To get into the directory where the rexrov_default.launch file is located and launch the file to view it in rviz using the 'roslaunch uuv_gazebo rexrov_default.launch' command line.
5.	rqt	To obtain the rqt screen where we can view all the nodes, topics, services, and messages and msg/srv types. We just type 'rqt'.
6.	rqt_graph	To view the detailed graph of launched file with 'rqt_graph' command. We got to see nodes, tf and topics and how they were connected.
7.	rosnode list	To see all the nodes in the launch file while its running from the command using this command line.
8.	rostopic list	To see topic from the command line
9.	rosmsg list	To see the messages involved in the launch
10.	rossrv list	To see the services running while the file launches

2. Controlling the robot usingteleop_twist_keyboard: (Contribution 33.3% each)

(a) Write a launch file that launches the rexrov in uuv_gazebo/rexrov_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)

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



Fixed Frame	world
Background Color	48; 48; 48
Frame Rate	30
Default Light	☑
Global Status: Ok	
TF	☑
Grid	☑
Status: Ok	
Reference Frame	<Fixed Frame>
Plane Cell Count	10
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Color	160; 160;
Alpha	0,5
Plane	XY
Offset	0; 0; 0
Image	☑
RobotModel	☑
Odometry	☑
Status: Ok	
Topic	/rexrov/pose...
Unreliable	

Topic
nav_msgs/Odometry topic to subscribe to.

Type: Orbit (rviz)

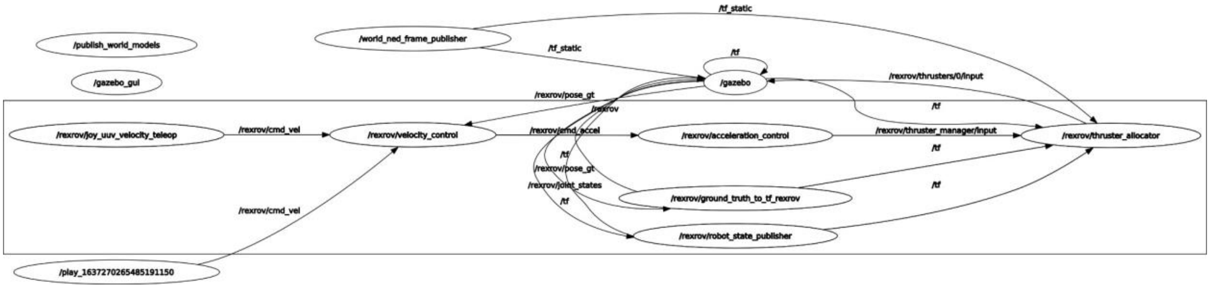
Current V...	Orbit (rviz)
Near Cl...	0,01
Invert ...	□
Target ...	rexrov/base_link
Distance	5,51004
Focal S...	0,05
Focal S...	□
Yaw	0,760399
Pitch	0,340398
Field o...	0,785398
Focal P...	0; 0; 0

Synchronization: Off ROS Time: 118.55 ROS Elapsed: 118.35 Wall Time: 1668882513.67 Wall Elapsed: 133.18

Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options. 31 fps

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

Shown below is the screenshot of the rqt_graph while the bag is playing


/tf_static
/publish_world_models
/world_ned_frame_publisher
/tf_static
/tf
/gazebo_gui
/gazebo
/rexrov/thrusters/0/input
/rexrov/pose_gt
/rexrov
/tf
/rexrov/joy_uuv_velocity_teleop
/rexrov/cmd_vel
/rexrov/velocity_control
/rexrov/cmd_accel
/rexrov/acceleration_control
/rexrov/thruster_manager/input
/rexrov/thruster_allocator
/tf
/tf
/rexrov/pose_gt
/rexrov/joint_states
/tf
/tf
/rexrov/ground_truth_to_tf_rexrov
/rexrov/cmd_vel
/rexrov/robot_state_publisher
/play_1637270265485191150

3. In this exercise you need to create a node that outputs force and torques as input to control the AUV in uuv_gazebo/rexrov_wrench_control.launch based on this input. download this launch file catkin_ws/src/uuv_simulator/uuv_gazebo/launch/rexrov_demos and launch in the world spawned by uuv_gazebo_worlds/empty_underwater_world.launch (Contribution 33.3% each)

(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)

The node teleop_forces_torques.py is used 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)

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

4. Creating and controlling your own robot. (Contribution 33.3% each)

(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)

We have created an urdf file of our ROV model along with its launch file is in /uuv_simulator/rov/urdf/ROV.urdf and /uuv_simulator/rov/launch/ROV_rviz.launch.

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

We will have eight thrusters in our ROV model so that we can have controllability to our ROV. There are 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 effective and such that 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)

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

The location for the launch file is uuv_simulator/rov/launch/ROV_launcher.launch. We 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