RIS LAB I REPORT

Authors: Irfan Karmali, Hani Alnahas & Emmanuel Mutuku

The 4 tasks given on the RIS Lab project were completed by all of us as a team and each of us contributed equally hence we would kindly request for the report to be assessed as a team effort. In other words, all team members contributed approximately 33.3% to each task. **A zip file (called lab_report) contains all our content and tasks, this folder needs to be added to the uuv_simulator directory.**

Introduction

This report is divided into 4 parts and all 4 parts of the report are elaborated and covered descriptively below.

Part 1: Nodes, Topics, Messages and Services

In this task, we launched the uuv_gazebo/rexrov_default.launch and inspected the nodes, topics, services it launches and the message/services types they use to communicate respectively.

The launch file above was launched and the nodes that were active or currently running after the launch were inspected. The command *rosnode list* was used to inspect the nodes. As the figure below depicts that only **9 nodes** were active after the launch.

```
karmali_irfan@ubuntu-ik:~$ 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
/rviz
```

Figure 1: List of active nodes

To gain more information about the nodes, the command *rosnode info <name_of_node>* can be used. This gives information on the publishers, subscribers, services, and connection of the nodes. For this example the

node *rexrov/velocity_control* was used. The command *rosnode info rexrov/velocity_control* was run. The output can be seen below. The subscriptions and Services of the topic can be seen below.

```
karmali_irfan@ubuntu-ik:~$ rosnode info /rexrov/velocity_control
--------------------------------------------------------------------------------
Node [/rexrov/velocity_control]
Publications:
 * /rexrov/cmd_accel [geometry_msgs/Accel]
 * /rexrov/velocity_control/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
 * /rexrov/velocity_control/parameter_updates [dynamic_reconfigure/Config]
 * /rosout [rosgraph_msgs/Log]

Subscriptions:
 * /rexrov/cmd_vel [geometry_msgs/Twist]
 * /rexrov/pose_gt [unknown type]

Services:
 * /rexrov/velocity_control/get_loggers
 * /rexrov/velocity_control/set_logger_level
 * /rexrov/velocity_control/set_parameters


contacting node http://ubuntu-ik:43795/ ...
Pid: 2005
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound (45439 - 127.0.0.1:38648) [17]
    * transport: TCPROS
 * topic: /rexrov/cmd_accel
    * to: /rexrov/acceleration_control
    * direction: outbound (45439 - 127.0.0.1:38636) [9]
    * transport: TCPROS
 * topic: /rexrov/cmd_vel
    * to: /rexrov/joy_uuv_velocity_teleop (http://ubuntu-ik:44393/)
    * direction: inbound
    * transport: TCPROS
```

Figure 2: Rosnode info

Moreover, the topics were also inspected. In ROS, topics are named buses over which nodes exchange messages. To see the list of topics that are currently subscribed to or published to, the command *rostopic list* was used, the result of the running the command can be seen below.

```
karmali_irfan@ubuntu-ik:~$ 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
/tf
/tf_static
```

Figure 3: List of Active Topics

Moreover, to retrieve more information about rostopics the command *rostopic info <name_of_topic>* was used. For this example the node *rexrov/velocity_control* was used. The command *rostopic info rexrov/cmd-vel* was run. The output can be seen below. The publishers and the subscribers of the topic can be seen below.

```
karmali_irfan@ubuntu-ik:~$ rostopic info /rexrov/cmd_vel
Type: geometry_msgs/Twist

Publishers:
 * /rexrov/joy_uuv_velocity_teleop (http://ubuntu-ik:44393/)

Subscribers:
 * /rexrov/velocity_control (http://ubuntu-ik:43795/)
```

Figure 4: rostopic info

After the topics were inspected, the messages were inspected. Nodes communicate with each other by publishing messages to topics. The list of messages active or running was retrieved using the command *rosmsg list.*

```
karmali_irfan@ubuntu-ik:~$ 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
beginner_tutorials/WaveParam
bond/Constants
bond/Status
```

```
controller_manager_msgs/ControllersStatistics
controller_manager_msgs/HardwareInterfaceResources
diagnostic_msgs/DiagnosticArray
diagnostic_msgs/DiagnosticStatus
diagnostic_msgs/KeyValue
dynamic_reconfigure/BoolParameter
dynamic_reconfigure/Config
dynamic_reconfigure/ConfigDescription
dynamic_reconfigure/DoubleParameter
dynamic_reconfigure/Group
dynamic_reconfigure/GroupState
dynamic_reconfigure/IntParameter
dynamic_reconfigure/ParamDescription
dynamic_reconfigure/SensorLevels
dynamic_reconfigure/StrParameter
gazebo_msgs/ContactState
gazebo_msgs/ContactsState
gazebo_msgs/LinkState
gazebo_msgs/LinkStates
gazebo_msgs/ModelState
gazebo_msgs/ModelStates
gazebo_msgs/ODEJointProperties
gazebo_msgs/ODEPhysics
gazebo_msgs/PerformanceMetrics
gazebo_msgs/SensorPerformanceMetric
gazebo_msgs/WorldState
geometry_msgs/Accel
geometry_msgs/AccelStamped
geometry_msgs/AccelWithCovariance
geometry_msgs/AccelWithCovarianceStamped
geometry_msgs/Inertia
geometry_msgs/InertiaStamped
geometry_msgs/Point
geometry_msgs/Point32
geometry_msgs/PointStamped
geometry_msgs/Polygon
geometry_msgs/PolygonStamped
geometry_msgs/Pose
geometry_msgs/Pose2D
geometry_msgs/PoseArray
geometry_msgs/PoseStamped
geometry_msgs/PoseWithCovariance
geometry_msgs/PoseWithCovarianceStamped
```

```
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
control_msgs/GripperCommandFeedback
control_msgs/GripperCommandGoal
control_msgs/GripperCommandResult
control_msgs/JointControllerState
control_msgs/JointJog
control_msgs/JointTolerance
control_msgs/JointTrajectoryAction
control_msgs/JointTrajectoryActionFeedback
control_msgs/JointTrajectoryActionGoal
control_msgs/JointTrajectoryActionResult
control_msgs/JointTrajectoryControllerState
control_msgs/JointTrajectoryFeedback
control_msgs/JointTrajectoryGoal
control_msgs/JointTrajectoryResult
control_msgs/PidState
control_msgs/PointHeadAction
control_msgs/PointHeadActionFeedback
control_msgs/PointHeadActionGoal
control_msgs/PointHeadActionResult
control_msgs/PointHeadFeedback
control_msgs/PointHeadGoal
control_msgs/PointHeadResult
control_msgs/SingleJointPositionAction
control_msgs/SingleJointPositionActionFeedback
control_msgs/SingleJointPositionActionGoal
control_msgs/SingleJointPositionActionResult
control_msgs/SingleJointPositionFeedback
control_msgs/SingleJointPositionGoal
control_msgs/SingleJointPositionResult
controller_manager_msgs/ControllerState
controller_manager_msgs/ControllerStatistics
```

Figure 5: List of rosmessages Part 1

```
geometry_msgs/Vector3
geometry_msgs/Vector3Stamped
geometry_msgs/Wrench
geometry_msgs/WrenchStamped
map_msgs/OccupancyGridUpdate
map_msgs/PointCloud2Update
map_msgs/ProjectedMap
map_msgs/ProjectedMapInfo
nav_msgs/GetMapAction
nav_msgs/GetMapActionFeedback
nav_msgs/GetMapActionGoal
nav_msgs/GetMapActionResult
nav_msgs/GetMapFeedback
nav_msgs/GetMapGoal
nav_msgs/GetMapResult
nav_msgs/GridCells
nav_msgs/MapMetaData
nav_msgs/OccupancyGrid
nav_msgs/Odometry
nav_msgs/Path
pcl_msgs/ModelCoefficients
pcl_msgs/PointIndices
pcl_msgs/PolygonMesh
pcl_msgs/Vertices
roscpp/Logger
rosgraph_msgs/Clock
rosgraph_msgs/Log
rosgraph_msgs/TopicStatistics
rospy_tutorials/Floats
rospy_tutorials/HeaderString
sensor_msgs/BatteryState
sensor_msgs/CameraInfo
sensor_msgs/ChannelFloat32
sensor_msgs/CompressedImage
sensor_msgs/FluidPressure
sensor_msgs/Illuminance
sensor_msgs/Image
sensor_msgs/Imu
sensor_msgs/JointState
sensor_msgs/Joy
sensor_msgs/JoyFeedback
sensor_msgs/JoyFeedbackArray
sensor_msgs/LaserEcho
```

```
sensor_msgs/LaserScan
sensor_msgs/MagneticField
sensor_msgs/MultiDOFJointState
sensor_msgs/MultiEchoLaserScan
sensor_msgs/NavSatFix
sensor_msgs/NavSatStatus
sensor_msgs/PointCloud
sensor_msgs/PointCloud2
sensor_msgs/PointField
sensor_msgs/Range
sensor_msgs/RegionOfInterest
sensor_msgs/RelativeHumidity
sensor_msgs/Temperature
sensor_msgs/TimeReference
shape_msgs/Mesh
shape_msgs/MeshTriangle
shape_msgs/Plane
shape_msgs/SolidPrimitive
smach_msgs/SmachContainerInitialStatusCmd
smach_msgs/SmachContainerStatus
smach_msgs/SmachContainerStructure
std_msgs/Bool
std_msgs/Byte
std_msgs/ByteMultiArray
std_msgs/Char
std_msgs/ColorRGBA
std_msgs/Duration
std_msgs/Empty
std_msgs/Float32
std_msgs/Float32MultiArray
std_msgs/Float64
std_msgs/Float64MultiArray
std_msgs/Header
std_msgs/Int16
std_msgs/Int16MultiArray
std_msgs/Int32
std_msgs/Int32MultiArray
std_msgs/Int64
std_msgs/Int64MultiArray
std_msgs/Int8
std_msgs/Int8MultiArray
std_msgs/MultiArrayDimension
std_msgs/MultiArrayLayout
```

```
std_msgs/String
std_msgs/Time
std_msgs/UInt16
std_msgs/UInt16MultiArray
std_msgs/UInt32
std_msgs/UInt32MultiArray
std_msgs/UInt64
std_msgs/UInt64MultiArray
std_msgs/UInt8
std_msgs/UInt8MultiArray
stereo_msgs/DisparityImage
tf/tfMessage
tf2_msgs/LookupTransformAction
tf2_msgs/LookupTransformActionFeedback
tf2_msgs/LookupTransformActionGoal
tf2_msgs/LookupTransformActionResult
tf2_msgs/LookupTransformFeedback
tf2_msgs/LookupTransformGoal
tf2_msgs/LookupTransformResult
tf2_msgs/TF2Error
tf2_msgs/TFMessage
theora_image_transport/Packet
trajectory_msgs/JointTrajectory
trajectory_msgs/JointTrajectoryPoint
trajectory_msgs/MultiDOFJointTrajectory
trajectory_msgs/MultiDOFJointTrajectoryPoint
turtle_actionlib/ShapeAction
turtle_actionlib/ShapeActionFeedback
turtle_actionlib/ShapeActionGoal
turtle_actionlib/ShapeActionResult
turtle_actionlib/ShapeFeedback
turtle_actionlib/ShapeGoal
turtle_actionlib/ShapeResult
turtle_actionlib/Velocity
turtlesim/Color
turtlesim/Pose
uuv_auv_control_allocator/AUVCommand
uuv_control_msgs/Trajectory
uuv_control_msgs/TrajectoryPoint
uuv_control_msgs/Waypoint
uuv_control_msgs/WaypointSet
uuv_gazebo_ros_plugins_msgs/FloatStamped
uuv_gazebo_ros_plugins_msgs/ThrusterConversionFcn
```

Figure 6: List of rosmessages Part 2

Lastly, the services were inspected. In simple terms, services are another way of passing data between nodes in ROS. The command *rossrv list* was used, the results of running the command can be seen below.

```
karmali_irfan@ubuntu-ik:~$ rossrv list
beginner_tutorials/AddTwoInts
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
roscpp/Empty
roscpp/GetLoggers
roscpp/SetLoggerLevel
roscpp_tutorials/TwoInts
rospy_tutorials/AddTwoInts
rospy_tutorials/BadTwoInts
rviz/SendFilePath
sensor_msgs/SetCameraInfo
std_srvs/Empty
std_srvs/SetBool
std_srvs/Trigger
tf/FrameGraph
tf2_msgs/FrameGraph
topic_tools/DemuxAdd
topic_tools/DemuxDelete
topic_tools/DemuxList
topic_tools/DemuxSelect
topic_tools/MuxAdd
topic_tools/MuxDelete
topic_tools/MuxList
topic_tools/MuxSelect
turtlesim/Kill
turtlesim/SetPen
turtlesim/Spawn
turtlesim/TeleportAbsolute
turtlesim/TeleportRelative
uuv_control_msgs/AddWaypoint
uuv_control_msgs/ClearWaypoints
uuv_control_msgs/GetMBSMControllerParams
uuv_control_msgs/GetPIDParams
uuv_control_msgs/GetSMControllerParams
uuv_control_msgs/GetWaypoints
```

```
uuv_control_msgs/GetPIDParams
uuv_control_msgs/GetSMControllerParams
uuv_control_msgs/GetWaypoints
uuv_control_msgs/GoTo
uuv_control_msgs/GoToIncremental
uuv_control_msgs/Hold
uuv_control_msgs/InitCircularTrajectory
uuv_control_msgs/InitHelicalTrajectory
uuv_control_msgs/InitRectTrajectory
uuv_control_msgs/InitWaypointSet
uuv_control_msgs/InitWaypointsFromFile
uuv_control_msgs/IsRunningTrajectory
uuv_control_msgs/ResetController
uuv_control_msgs/SetMBSMControllerParams
uuv_control_msgs/SetPIDParams
uuv_control_msgs/SetSMControllerParams
uuv_control_msgs/StartTrajectory
uuv_control_msgs/SwitchToAutomatic
uuv_control_msgs/SwitchToManual
uuv_gazebo_ros_plugins_msgs/GetFloat
uuv_gazebo_ros_plugins_msgs/GetListParam
uuv_gazebo_ros_plugins_msgs/GetModelProperties
uuv_gazebo_ros_plugins_msgs/GetThrusterConversionFcn
uuv_gazebo_ros_plugins_msgs/GetThrusterEfficiency
uuv_gazebo_ros_plugins_msgs/GetThrusterState
uuv_gazebo_ros_plugins_msgs/SetFloat
uuv_gazebo_ros_plugins_msgs/SetThrusterEfficiency
uuv_gazebo_ros_plugins_msgs/SetThrusterState
uuv_gazebo_ros_plugins_msgs/SetUseGlobalCurrentVel
uuv_sensor_ros_plugins_msgs/ChangeSensorState
uuv_thruster_manager/GetThrusterCurve
uuv_thruster_manager/GetThrusterManagerConfig
uuv_thruster_manager/SetThrusterManagerConfig
uuv_thruster_manager/ThrusterManagerInfo
uuv_world_ros_plugins_msgs/GetCurrentModel
uuv_world_ros_plugins_msgs/GetOriginSphericalCoord
uuv_world_ros_plugins_msgs/SetCurrentDirection
uuv_world_ros_plugins_msgs/SetCurrentModel
uuv_world_ros_plugins_msgs/SetCurrentVelocity
uuv_world_ros_plugins_msgs/SetOriginSphericalCoord
uuv_world_ros_plugins_msgs/TransformFromSphericalCoord
uuv_world_ros_plugins_msgs/TransformToSphericalCoord
```

Figure 7: List of ros services

Moreover, to gain more information on the services the command *rosservice info <name_of_node or name_of_topic> <name_of_service>* can be used. For this example the node *rexrov/velocity_control* was used. The command *rosservice info rexrov/acceleration-control/get_loggers* was run. The output can be seen below.

```
karmali_irfan@ubuntu-ik:~$ rosservice info /rexrov/acceleration_control/get_loggers
Node: /rexrov/acceleration_control
URI: rosrpc://ubuntu-ik:34703
Type: roscpp/GetLoggers
Args:
```

Figure 8: Rosservice info

Lastly, to graphically represent the active nodes and topics the command *rosrun rqt_graph rqt_graph* was used. The diagram below depicts the output.

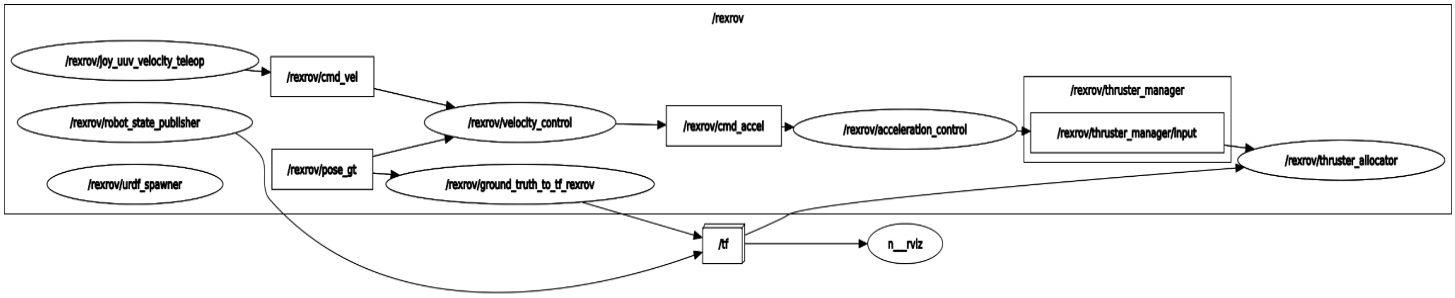


Figure 9: Rgt graph showing all the nodes and topics

Part 2: Controlling the Robot

For the first part of Part 2, a launch file was created to launch **rexrov** in the world spawned by **empty_underwater_world.launch**, the launch file also launched the node (**teleop_twist_keyboard**) which was used to control the simulated ROV in the world. The launch file used (**rov_teleop.launch)** can be found in the directory **uuv_simulator/lab_report/lab_report/launch**. The command *roslaunch lab_report rov_teleop.launch* can be used to launch the file.

For the second part of Part 2, **rosbag** was used to log the topic **teleop_twist_keyboard/cmd_vel.** The ROV was moved while **rosbag** was recording and then the node **teleop_twist_keyboard** was stopped. Then the robot was moved and controlled by replaying the generated **rosbag**. A plot of the ROV's traced trajectory as a result of **rosbag** replaying using **rqt_graph** is shown below.

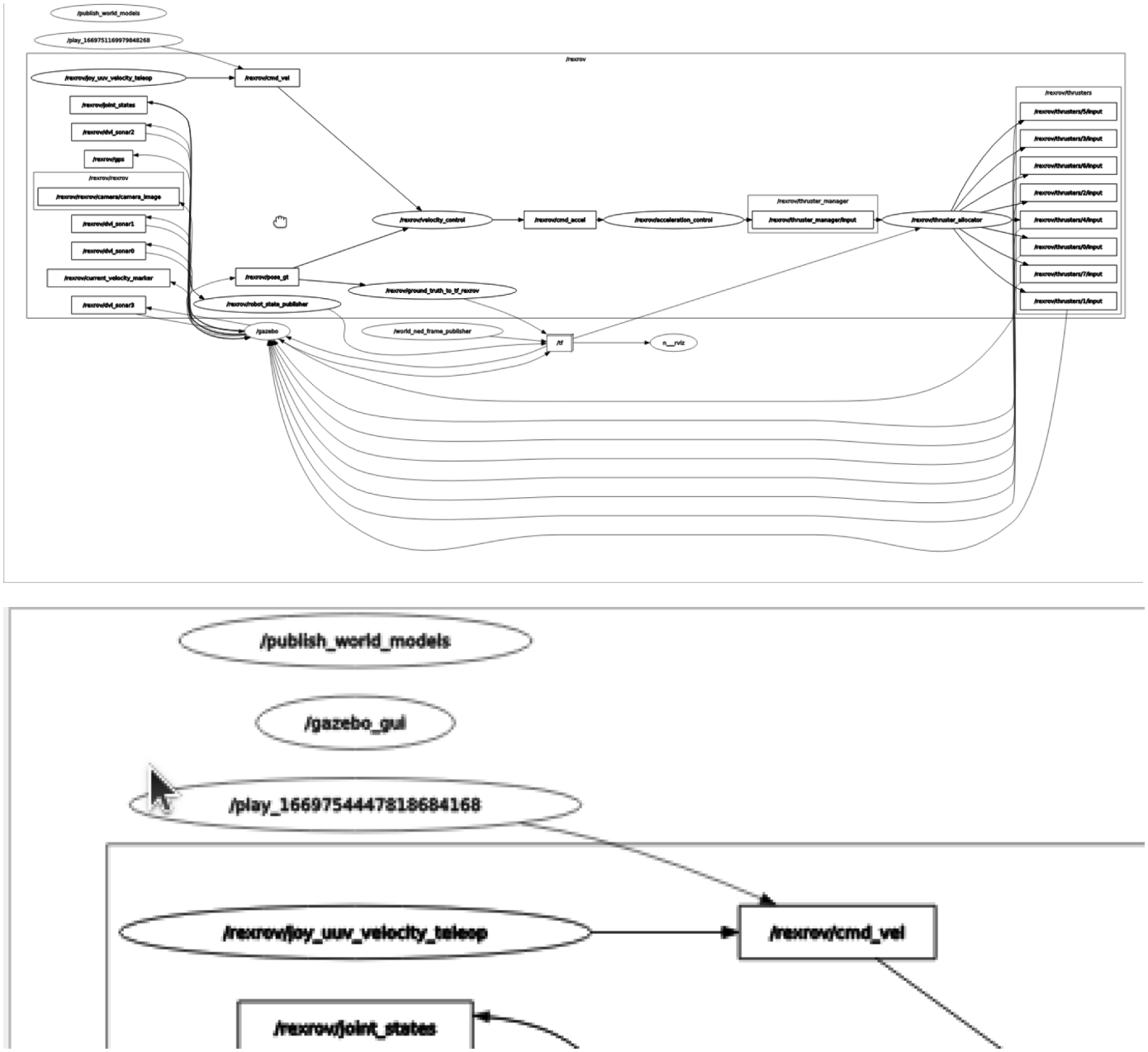


Figure 10: RQT graph showing the rosbag playing and moving the ROV

The plot of the trajectory using the recorded rosbag can be found below.

The command *rosrun rqt_bag rqt_bag* was run in the terminal to generate the plot from the rosbag.

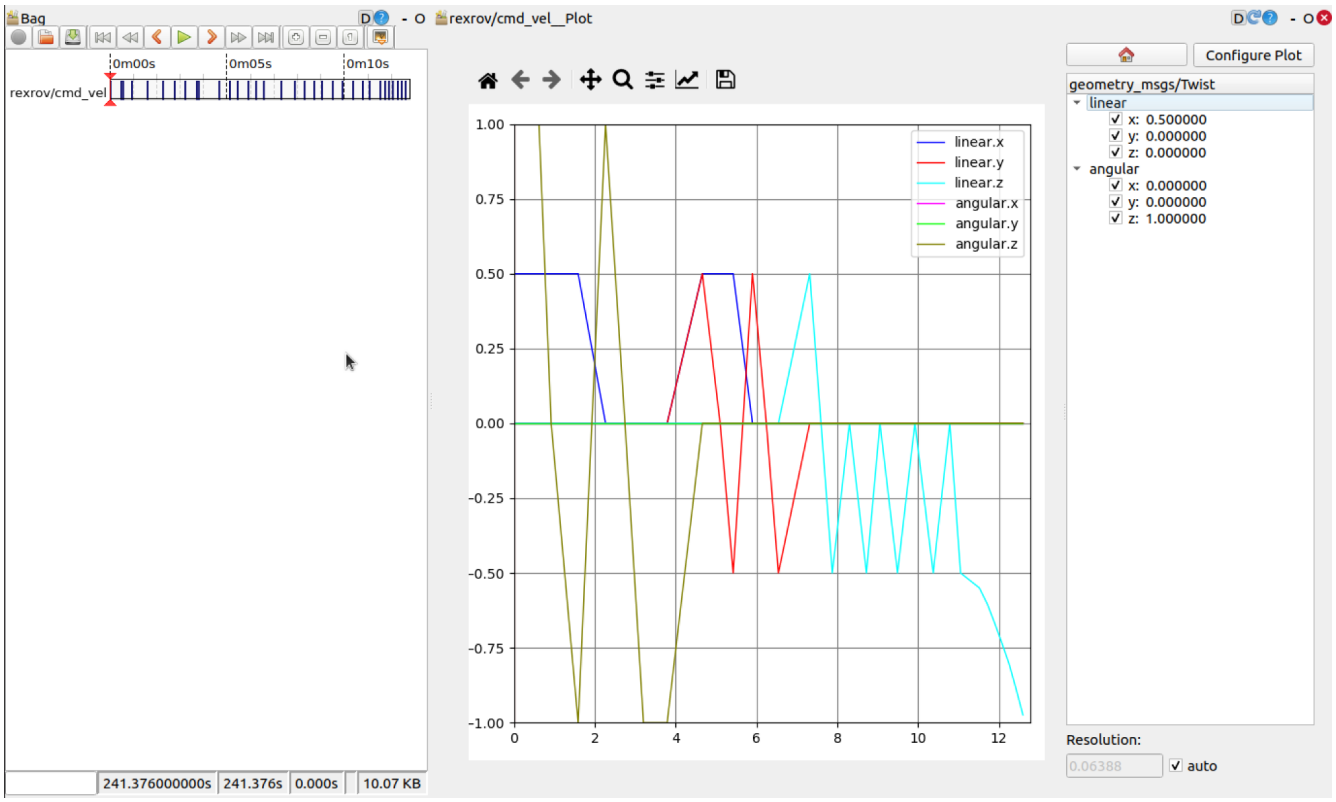


Figure 11: Plot showing the traced trajectory

<u>Part 3: Forces and Torque Output</u>

For the first part of Part 3, a node was created that publishes forces and torques to control the AUV, this specific node publishes to **rexrov/thruster_manager/input.** The node, **force_keyboard_control.py**, can be found in the directory **uuv_simulator/lab_report/lab_report/scripts.**

For the second part of Part 3, a launch file was created that launches the AUV and the node above. The launch file, **new_wrench_control.launch**, can be found in the directory **uuv_simulator/lab_report/lab_report/launch.**

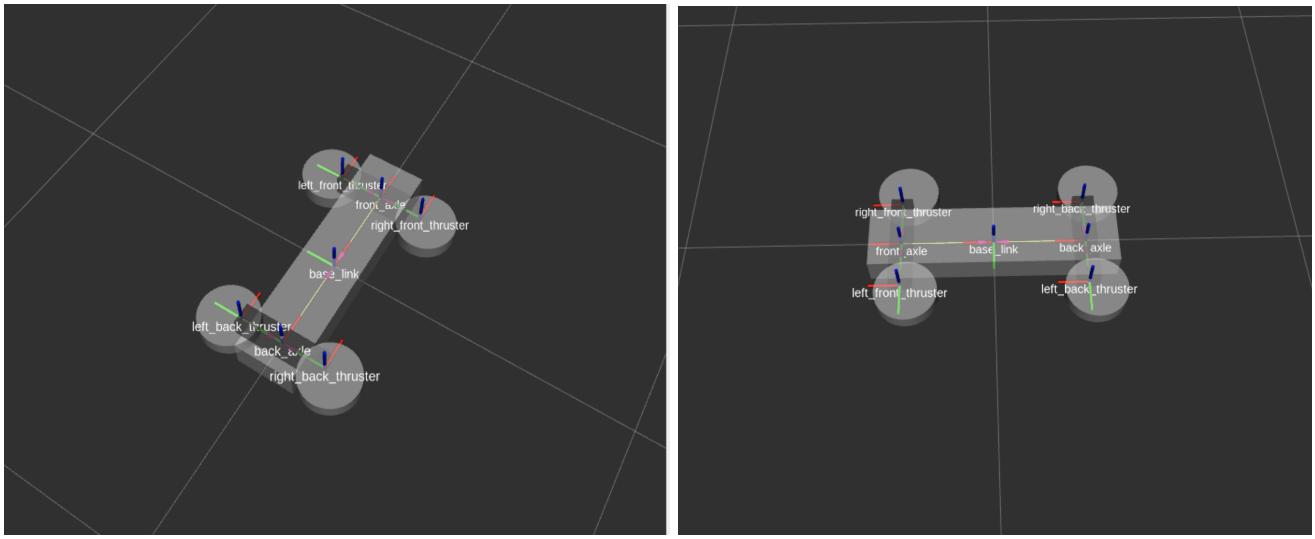
Part 4a: Creating your own Robot

This task involves creating and controlling a robot of our own design. A urdf file *(submarine.urdf)* that describes our simple ROV can be found in the directory path *urdf_tutorial/urdf*. The geometric shapes used include a box size for the base link and 4 cylinders for the two front thrusters and two back thrusters on both the left and right side of the base link. Below is a sample code that describes the robot:

```xml
<?xml version="1.0"?>

<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="submarine">

    <material name="white">
        <color rgba="1 1 1 1"/>
    </material>

    <material name="gray">
        <color rgba="0.5 0.5 0.5 1"/>
    </material>

    <material name="black">
        <color rgba="0 0 0 1"/>
    </material>


<xacro:property name="axis_o" value="0 1 0"/>
<xacro:property name="axle_s" value="0.07 0.3 0.07"/>


    <link name="base_link">
        <visual>
            <geometry>
                <box size="0.8 0.2 0.1"/>
            </geometry>
            <material name="white"/>
        </visual>
    </link>

    <link name="front_axle">
        <visual>
            <geometry>
                <box size="${axle_s}"/>
            </geometry>
            <material name="black"/>
        </visual>
    </link>

    <joint name="base_to_front_axle" type="fixed">
        <parent link="base_link"/>
        <child link="front_axle"/>
        <origin xyz="0.3 0 0"/>
    </joint>

    <link name="back_axle">
        <visual>
            <geometry>
                <box size="${axle_s}"/>
            </geometry>
            <material name="black"/>
        </visual>
    </link>

    <joint name="base_to_back_axle" type="fixed">
        <parent link="base_link"/>
        <child link="back_axle"/>
        <origin xyz="-0.3 0 0"/>
    </joint>

    <link name="left_front_thruster">
        <visual>
            <geometry>
                <cylinder length="0.05" radius="0.1"/>
            </geometry>
            <origin xyz="-0.02 0.03 0"/>
            <material name="white"/>
            <origin xyz="-0.02 0.03 0"/>
            <material name="white"/>
        </visual>
    </link>
        <parent link="front_axle"/>
        <child link="left_front_thruster"/>
        <axis xyz="${axis_o}"/>
        <origin xyz="0 0.15 0"/>
    </joint>

    <link name="right_front_thruster">
        <visual>
            <geometry>
                <cylinder length="0.05" radius="0.1"/>
            </geometry>
            <origin xyz="-0.02 -0.03 0"/>
            <material name="white"/>
        </visual>
    </link>
```

Figure 12: Sample Code of "wakanda_marine" robot in urdf file

Model of the Wakanda_marine in Rviz:



Figures 13 & 14: Top and side view of the model in Rviz

<u>Part 4b: Reasoning of the positioning of the thrusters</u>

The ROV model was based off an underwater car from a cartoon. The four thrusters are strategically placed to allow for maximum degrees of freedom for the ROV. All four thrusters can rotate freely hence also allowing for the ROV to move freely underwater depending on the angles of the thrusters. In addition, thrusters joints are continuous and not fixed and each of them has its own axis of rotation.

<u>Part 4c: Writing a Node that takes in forces and torques as input and publishes thrust commands for our ROV.</u>

Unfortunately, we failed to complete this task and write a node to perform the desired task.

<u>Part 4d: Writing a launch file</u>

A launch file was created to launch our ROV in gazebo and can be found in the directory **lab_report/urdf_tutorial/launch.** The launch file is called **display.launch**

An image of wakanda_marine that was launched in the gazebo using the launch file is below.

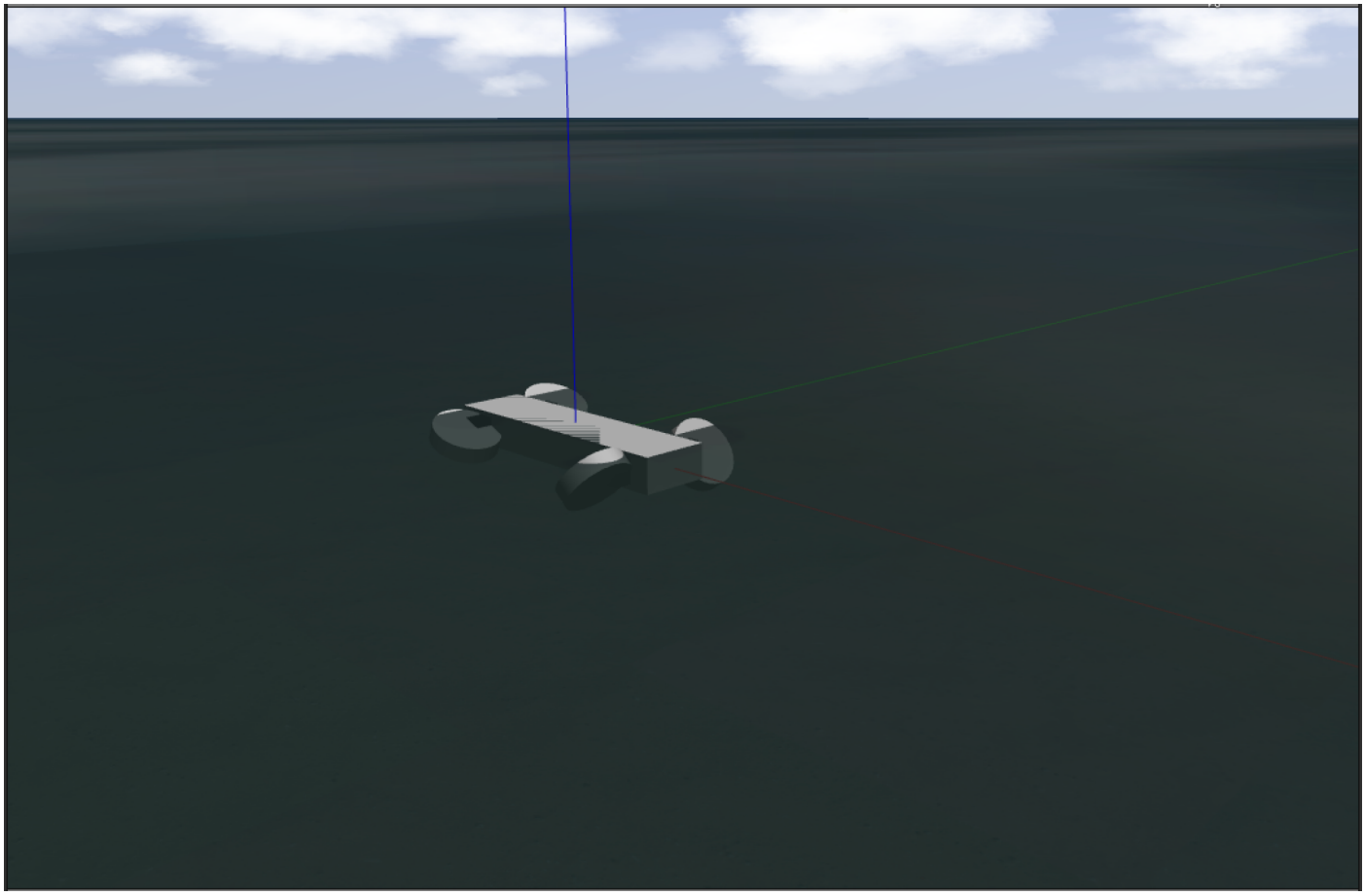


Figure 15: wakanda_marine in the Gazebo simulation

Citations

1. http://wiki.ros.org/urdf/Tutorials/Building%20a%20Visual%20Robot%20Model%20with%20URDF%20from%20Scratch
2. https://github.com/uuvsimulator/uuv_simulator/wiki
3. https://admantium.com/blog/ros03_visualize_with_rviz/
4. https://github.com/bluerobotics/bluerov-ros-pkg/blob/master/bluerov/robots/bluerov.urdf