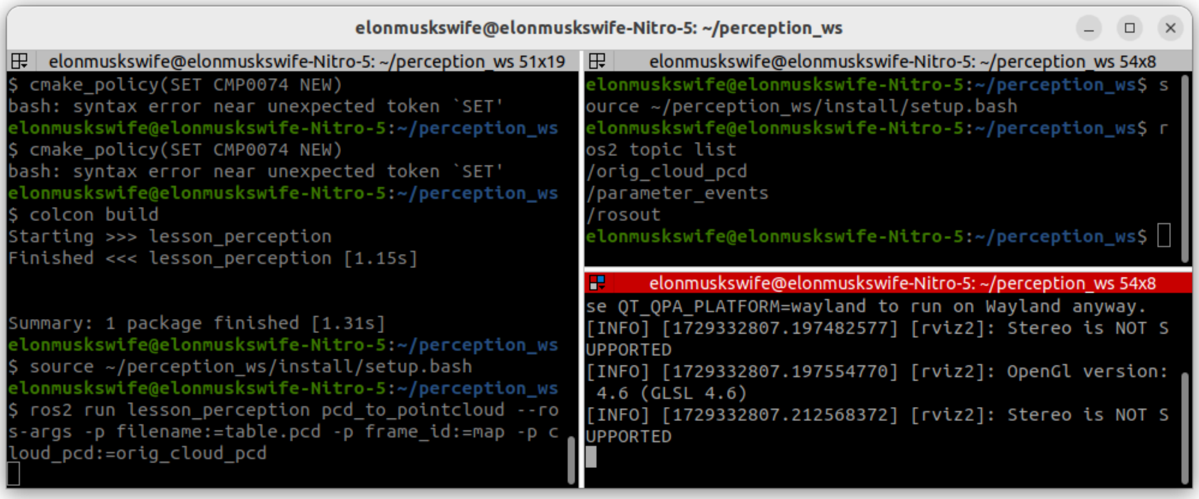
Madina Turmagambetova, Kanagat Kantoreyeva

Homework Lab Assignment #3
PERCEPTION IN ROS 2

1. Follow the instructions and complete ROS 2 tutorial exercise 4.1. – Intro to Perception.

In this exercise, we experiment with data generated from the Asus Xtion Pro (or Microsoft Kinect) sensor in order to become more familiar with processing 3D data.

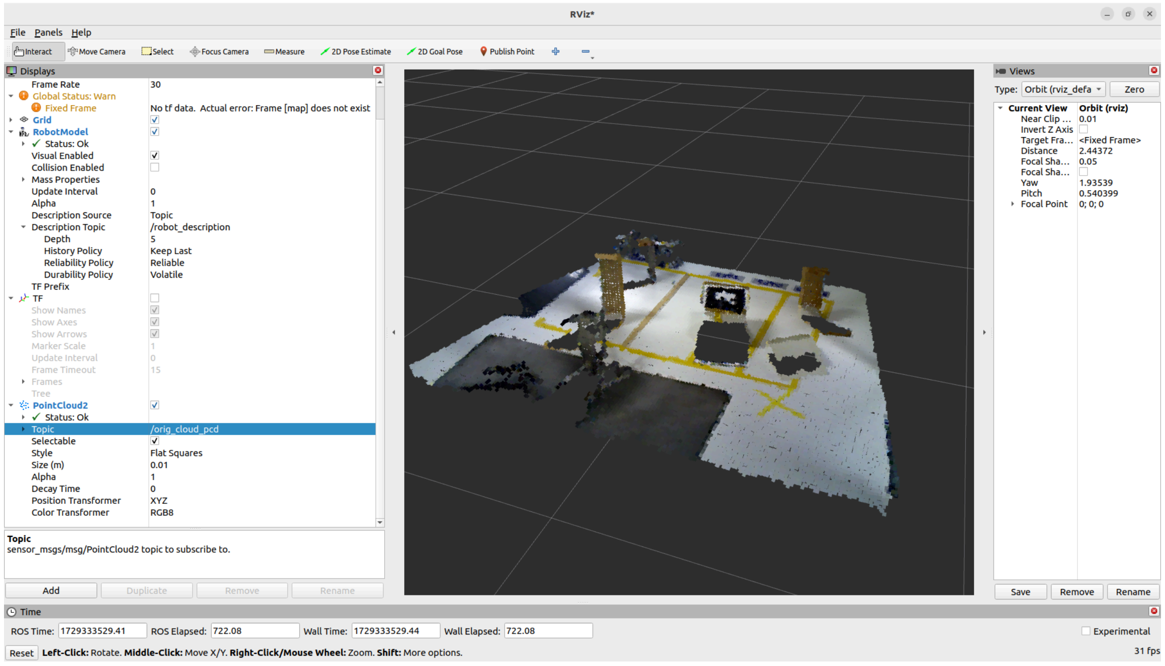


```
elonmuskswife@elonmuskswife-Nitro-5: ~/perception_ws

elonmuskswife@elonmuskswife-Nitro-5: ~/perception_ws 51x19
$ cmake_policy(SET CMP0074 NEW)
bash: syntax error near unexpected token `SET'
elonmuskswife@elonmuskswife-Nitro-5:~/perception_ws
$ cmake_policy(SET CMP0074 NEW)
bash: syntax error near unexpected token `SET'
elonmuskswife@elonmuskswife-Nitro-5:~/perception_ws
$ colcon build
Starting >>> lesson_perception
Finished <<< lesson_perception [1.15s]


Summary: 1 package finished [1.31s]
elonmuskswife@elonmuskswife-Nitro-5:~/perception_ws
$ source ~/perception_ws/install/setup.bash
elonmuskswife@elonmuskswife-Nitro-5:~/perception_ws
$ ros2 run lesson_perception pcd_to_pointcloud --ro
s-args -p filename:=table.pcd -p frame_id:=map -p c
loud_pcd:=orig_cloud_pcd
```

```
elonmuskswife@elonmuskswife-Nitro-5: ~/perception_ws 54x8
elonmuskswife@elonmuskswife-Nitro-5:~/perception_ws$ s
ource ~/perception_ws/install/setup.bash
elonmuskswife@elonmuskswife-Nitro-5:~/perception_ws$ r
os2 topic list
/orig_cloud_pcd
/parameter_events
/rosout
elonmuskswife@elonmuskswife-Nitro-5:~/perception_ws$
```

```
elonmuskswife@elonmuskswife-Nitro-5: ~/perception_ws 54x8
se QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[INFO] [1729332807.197482577] [rviz2]: Stereo is NOT S
UPPORTED
[INFO] [1729332807.197554770] [rviz2]: OpenGl version:
 4.6 (GLSL 4.6)
[INFO] [1729332807.212568372] [rviz2]: Stereo is NOT S
UPPORTED
```

We view its data stream and visualize the data in Rviz: /orig_cloud_pcd



```
Frame Rate                 30
Global Status: Warn
  Fixed Frame              No tf data.  Actual error: Frame [map] does not exist
Grid                       ✓
RobotModel                 ✓
  Status: Ok
  Visual Enabled           ✓
  Collision Enabled        ☐
  Mass Properties
  Update Interval          0
  Alpha                    1
  Description Source       Topic
  Description Topic        /robot_description
    Depth                  5
    History Policy         Keep Last
    Reliability Policy     Reliable
    Durability Policy      Volatile
  TF Prefix
TF                         ☐
  Show Names               ✓
  Show Axes                ✓
  Show Arrows              ✓
  Marker Scale             1
  Update Interval          0
  Frame Timeout            15
  Frames
  Tree
PointCloud2                ✓
  Status: Ok
  Topic                    /orig_cloud_pcd
  Selectable               ✓
  Style                    Flat Squares
  Size (m)                 0.01
  Alpha                    1
  Decay Time               0
  Position Transformer     XYZ
  Color Transformer        RGB8

Topic
sensor_msgs/msg/PointCloud2 topic to subscribe to.
```

```
Type: Orbit (rviz_defa
Current View     Orbit (rviz)
  Near Clip ...  0.01
  Invert Z Axis  ☐
  Target Fra...  <Fixed Frame>
  Distance       2.44372
  Focal Sha...   0.05
  Focal Sha...   ☐
  Yaw            1.93539
  Pitch          0.540399
  Focal Point    0; 0; 0
```

```
Time
ROS Time: 1729333529.41  ROS Elapsed: 722.08  Wall Time: 1729333529.44  Wall Elapsed: 722.08   ☐ Experimental
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.   31 fps
```

Experiment with PCL

❖ Downsample the original point cloud using a voxel grid.

by using:

```
pcl_voxel_grid table.pcd table_downsampled.pcd -leaf 0.05,0.05,0.05
pcl_viewer table_downsampled.pcd
```

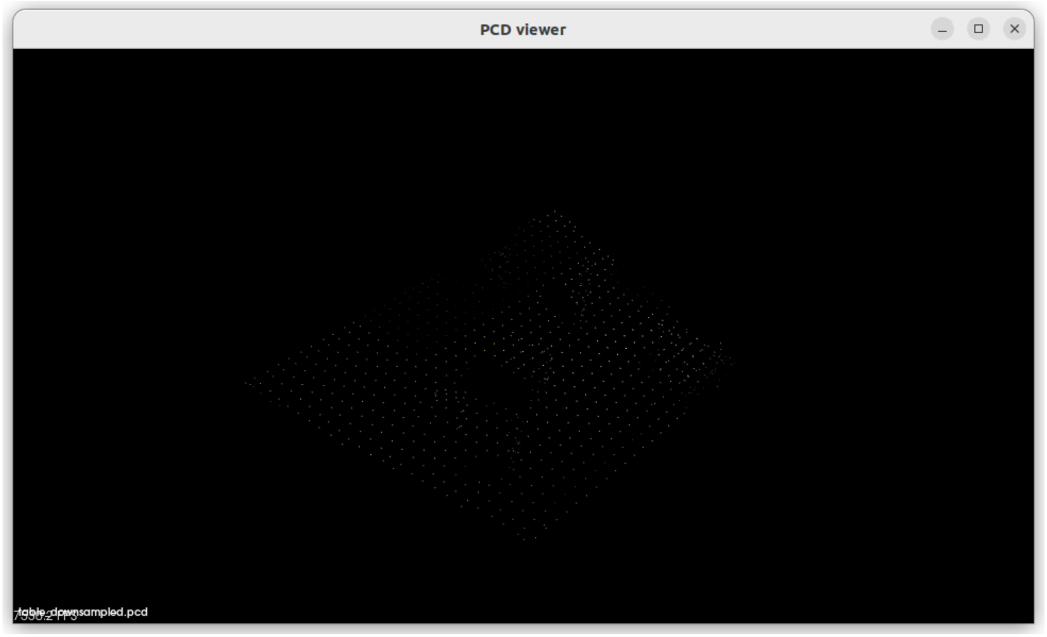
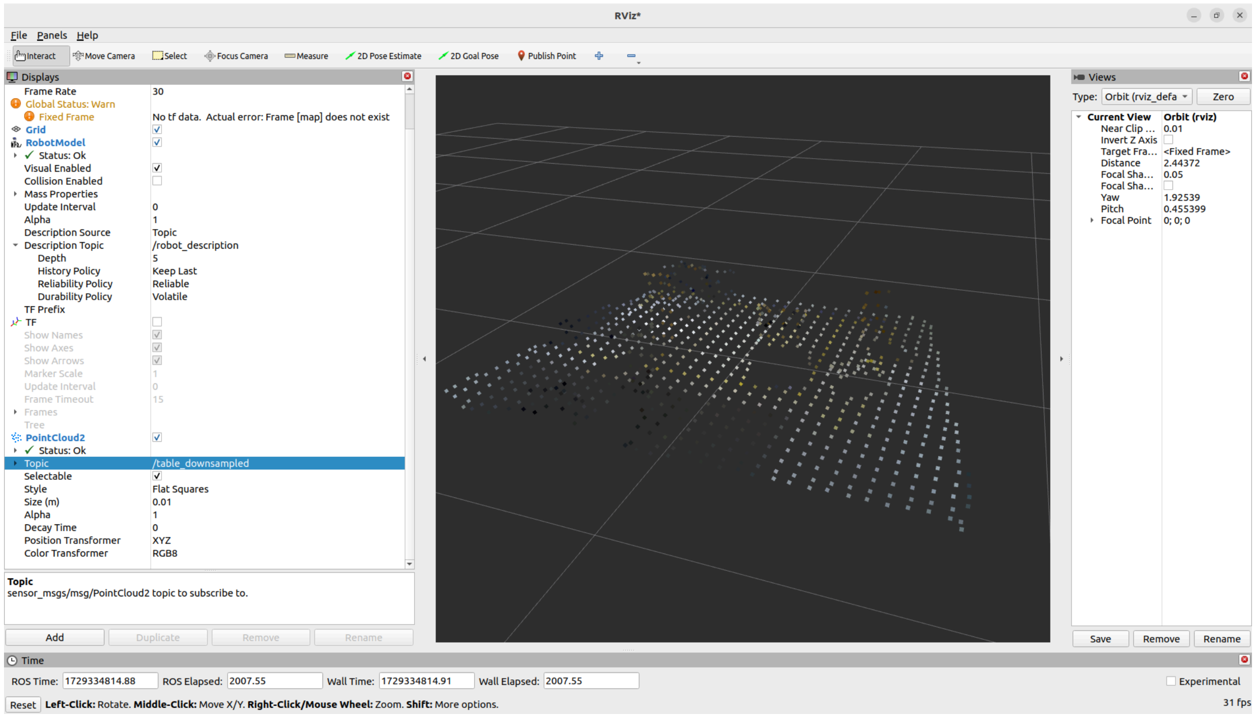


Figure 3. downsampled table viewed in pcl_viewer


RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Displays
Views
Add Duplicate Remove Rename
Save Remove Rename
Time
Reset

Frame Rate	30
Global Status: Warn	
Fixed Frame	No tf data. Actual error: Frame [map] does not exist
Grid	✓
RobotModel	✓
Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Mass Properties	
Update Interval	0
Alpha	1
Description Source	Topic
Description Topic	/robot_description
Depth	5
History Policy	Keep Last
Reliability Policy	Reliable
Durability Policy	Volatile
TF Prefix	
TF	☐
Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
Tree	
PointCloud2	✓
Status: Ok	
Topic	/table_downsampled
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	RGB8

Topic
sensor_msgs/msg/PointCloud2 topic to subscribe to.

Type: Orbit (rviz_defa

Current View	**Orbit (rviz)**
Near Clip …	0.01
Invert Z Axis	☐
Target Fra…	<Fixed Frame>
Distance	2.44372
Focal Sha…	0.05
Focal Sha…	☐
Yaw	1.92539
Pitch	0.455399
Focal Point	0; 0; 0

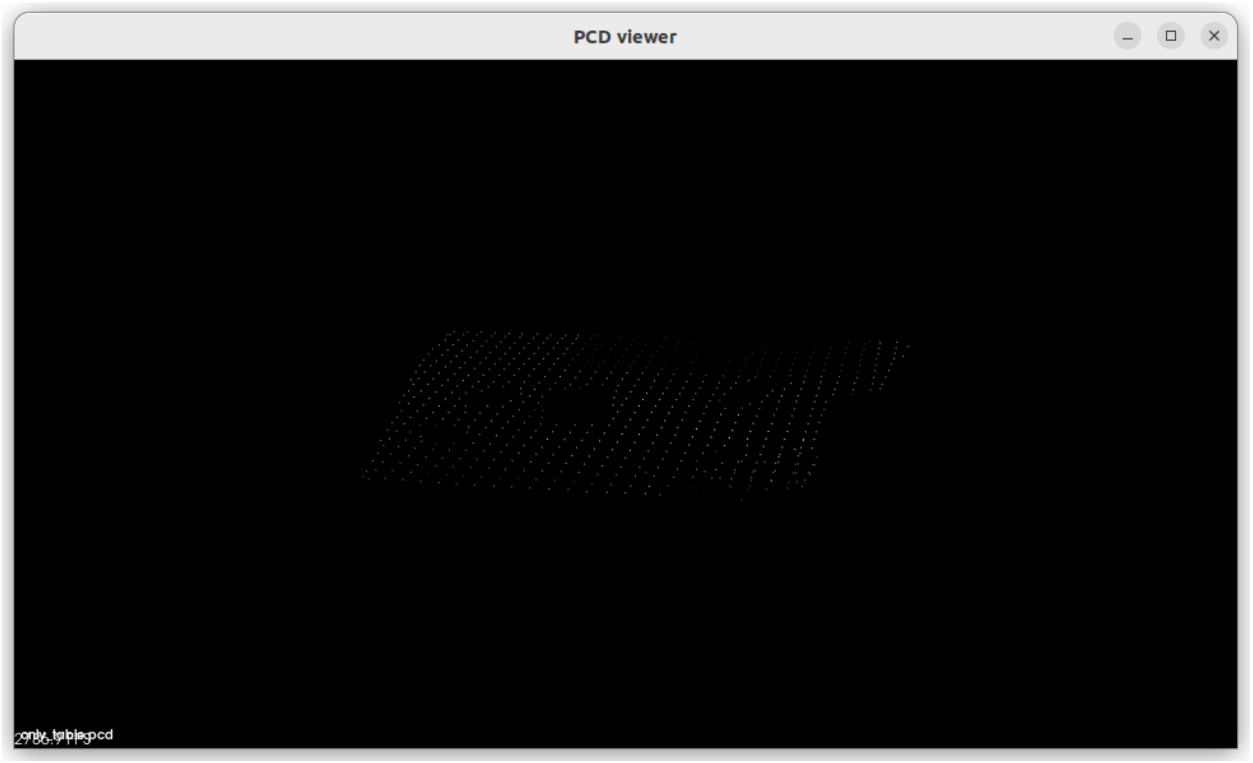
ROS Time: 1729334814.88 ROS Elapsed: 2007.55 Wall Time: 1729334814.91 Wall Elapsed: 2007.55 ☐ Experimental

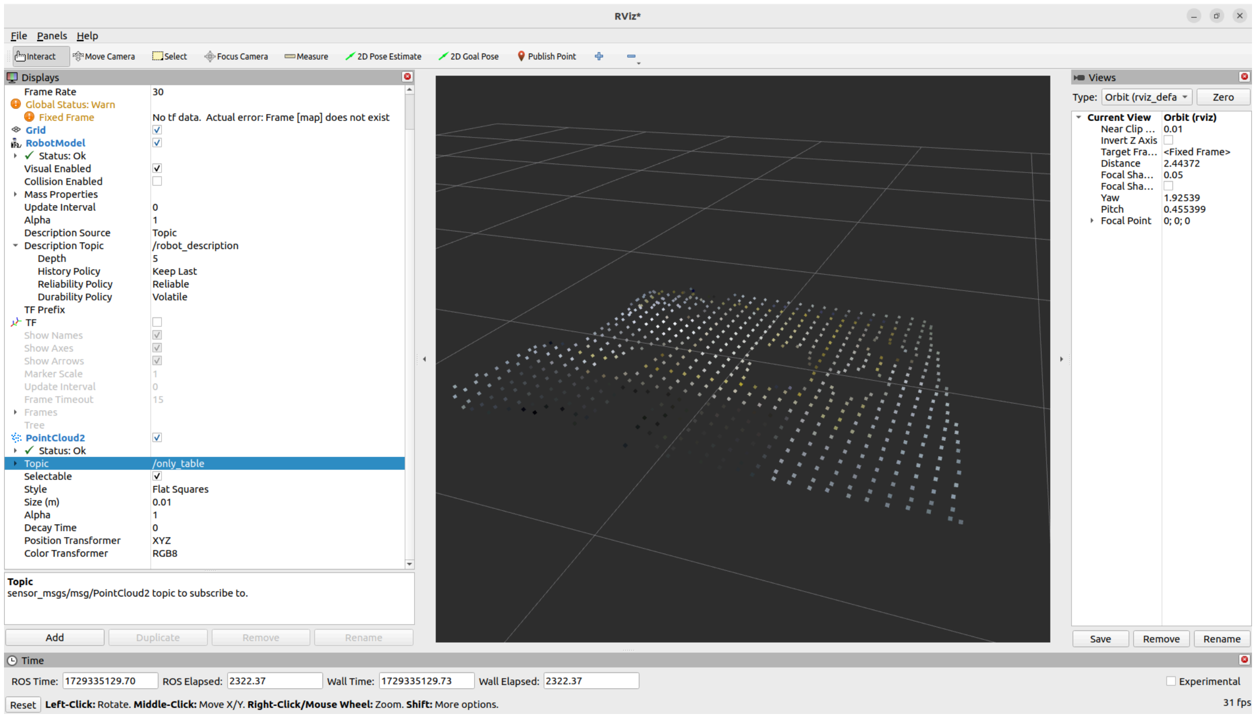
Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:** Zoom. **Shift:** More options. 31 fps

Figure 4. downsampled table viewed in rviz2

- ❖ Extracting the table surface from the point cloud using the pcl_sac_segmentation_plane. Find the largest plane and extract points that belong to that plane (within a given threshold).

```
pcl_sac_segmentation_plane table_downsampled.pcd only_table.pcd
-thresh 0.01
```




RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Displays
Views
Add Duplicate Remove Rename
Save Remove Rename
Zero
Reset

Frame Rate	30
Global Status: Warn	
Fixed Frame	No tf data. Actual error: Frame [map] does not exist
Grid	✓
RobotModel	✓
Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Mass Properties	
Update Interval	0
Alpha	1
Description Source	Topic
Description Topic	/robot_description
Depth	5
History Policy	Keep Last
Reliability Policy	Reliable
Durability Policy	Volatile
TF Prefix	
TF	☐
Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
Tree	
PointCloud2	✓
Status: Ok	
Topic	/only_table
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	RGB8

Topic
sensor_msgs/msg/PointCloud2 topic to subscribe to.

Type: Orbit (rviz_defa

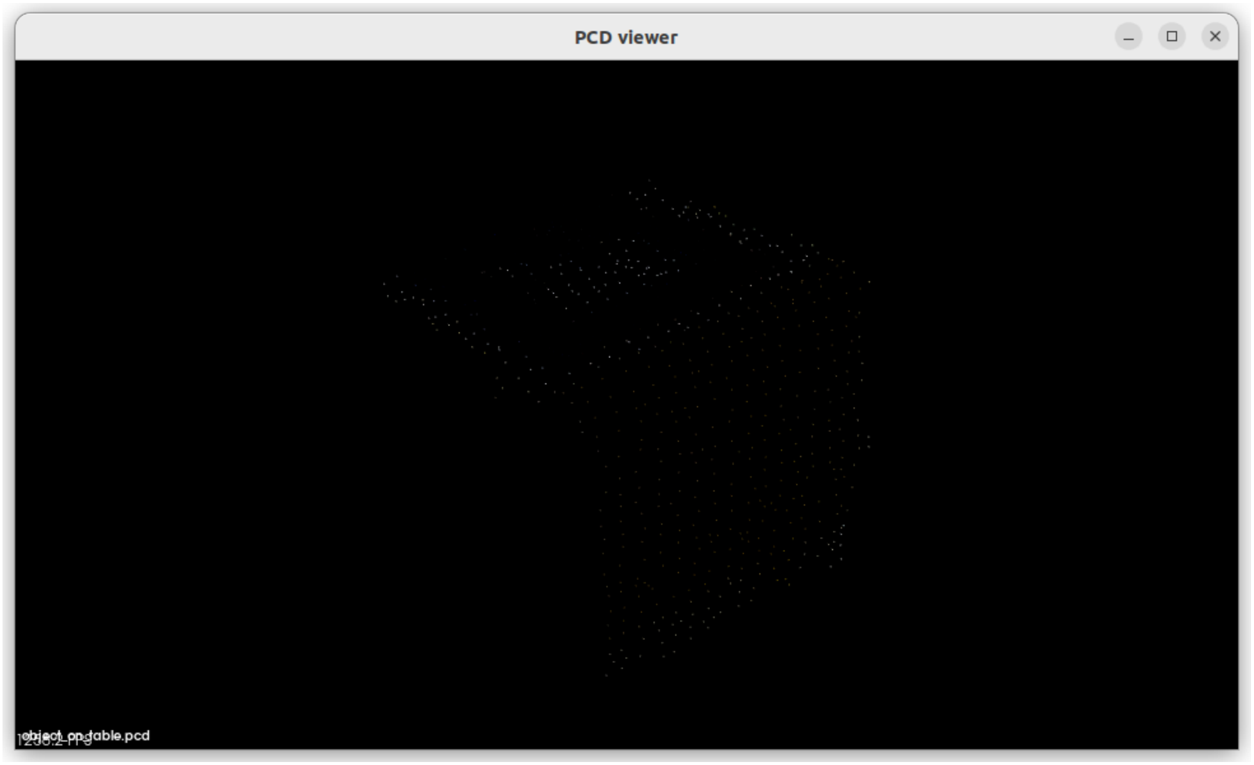
Current View	Orbit (rviz)
Near Clip …	0.01
Invert Z Axis	☐
Target Fra…	<Fixed Frame>
Distance	2.44372
Focal Sha…	0.05
Focal Sha…	☐
Yaw	1.92539
Pitch	0.455399
Focal Point	0; 0; 0

Time

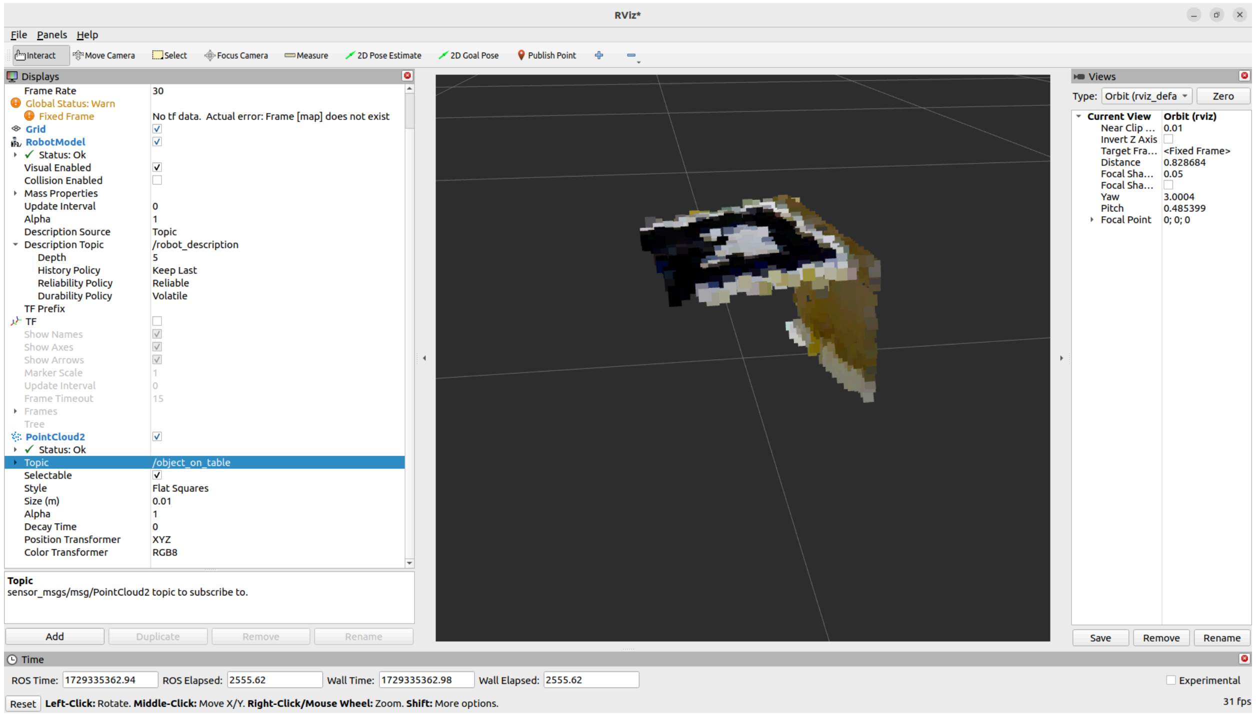
ROS Time: 1729335129.70 ROS Elapsed: 2322.37 Wall Time: 1729335129.73 Wall Elapsed: 2322.37 ☐ Experimental

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

❖ Extracting the largest cluster on the table from the point cloud using the pcl_sac_segmentation_plane.



largest object on table



Frame Rate	30
Global Status: Warn	
Fixed Frame	No tf data. Actual error: Frame [map] does not exist
Grid	✓
RobotModel	✓
Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Mass Properties	
Update Interval	0
Alpha	1
Description Source	Topic
Description Topic	/robot_description
Depth	5
History Policy	Keep Last
Reliability Policy	Reliable
Durability Policy	Volatile
TF Prefix	
TF	☐
Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
Tree	
PointCloud2	✓
Status: Ok	
Topic	/object_on_table
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	RGB8

Topic
sensor_msgs/msg/PointCloud2 topic to subscribe to.

Type: Orbit (rviz_defa

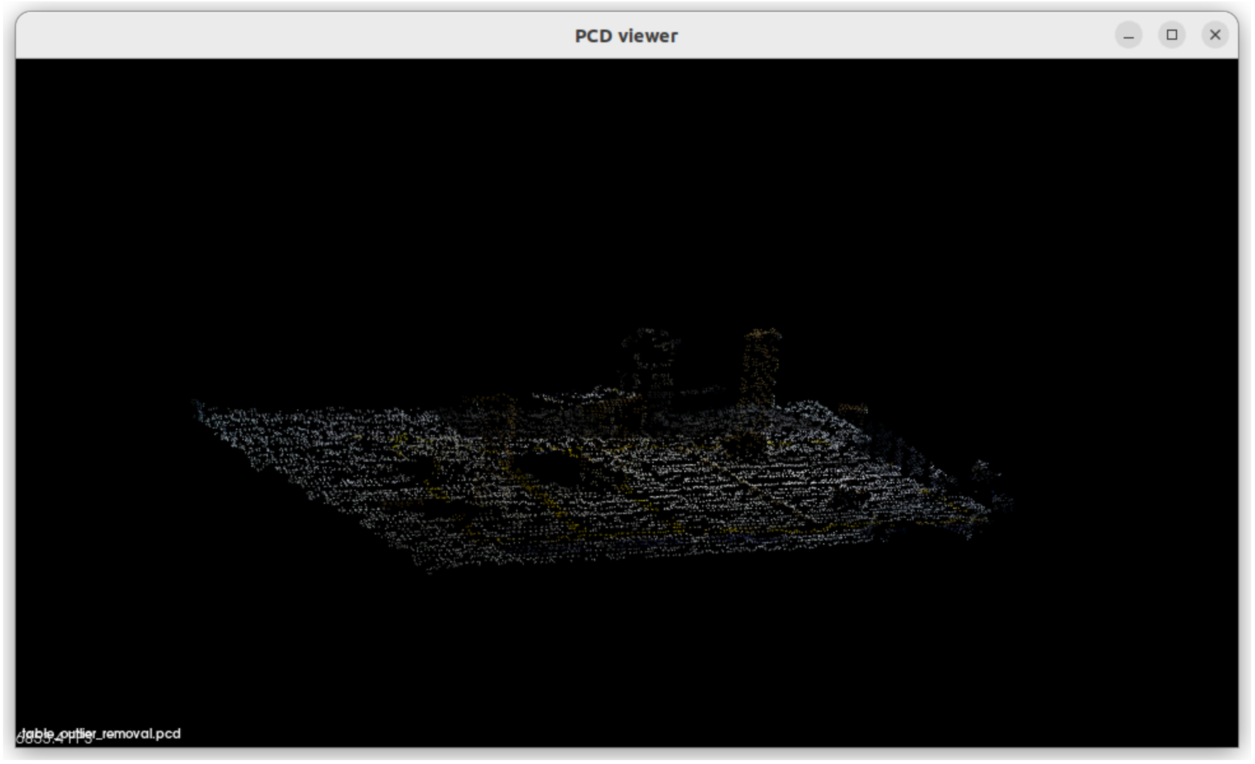
Current View	**Orbit (rviz)**
Near Clip …	0.01
Invert Z Axis	☐
Target Fra…	<Fixed Frame>
Distance	0.828684
Focal Sha…	0.05
Focal Sha…	☐
Yaw	3.0004
Pitch	0.485399
Focal Point	0; 0; 0

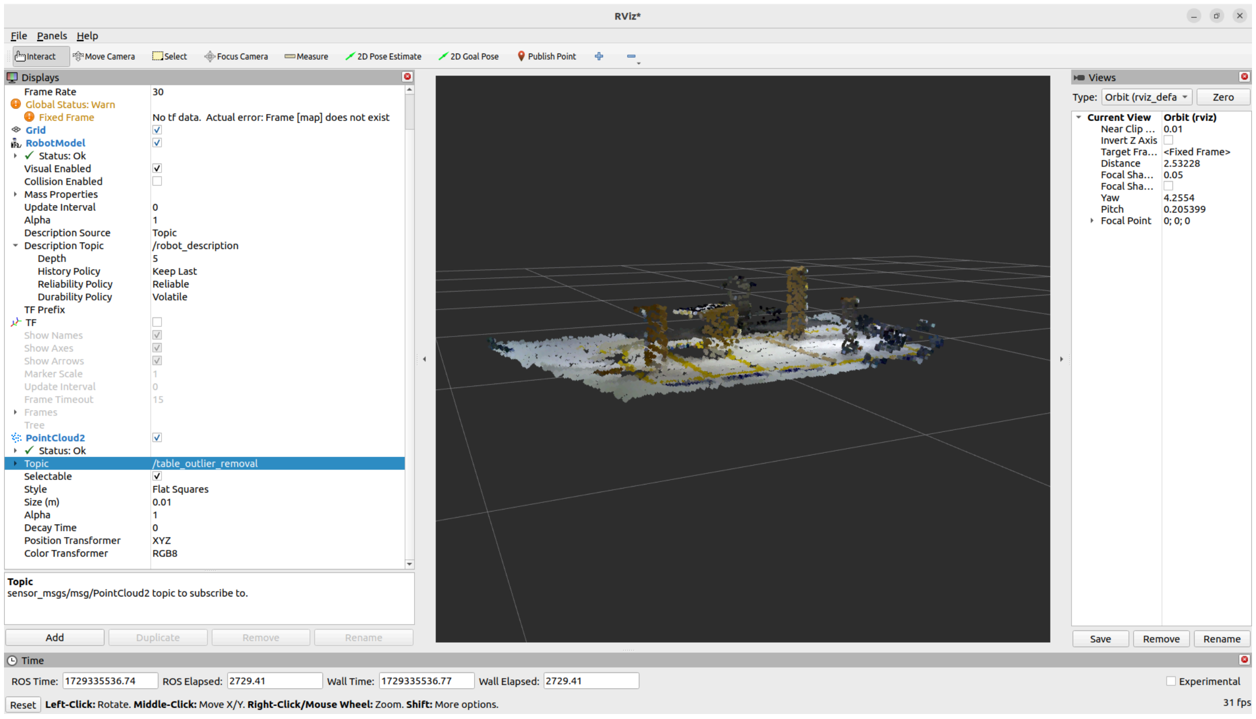
ROS Time: 1729335362.94 ROS Elapsed: 2555.62 Wall Time: 1729335362.98 Wall Elapsed: 2555.62 ☐ Experimental

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

in rviz

❖ Remove outliers from the cloud using the pcl_outlier_removal.




RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Add Duplicate Remove Rename
Save Remove Rename

Displays	
Frame Rate	30
Global Status: Warn	
Fixed Frame	No tf data. Actual error: Frame [map] does not exist
Grid	✓
RobotModel	✓
Status: Ok	
Visual Enabled	✓
Collision Enabled	☐
Mass Properties	
Update Interval	0
Alpha	1
Description Source	Topic
Description Topic	/robot_description
Depth	5
History Policy	Keep Last
Reliability Policy	Reliable
Durability Policy	Volatile
TF Prefix	
TF	☐
Show Names	✓
Show Axes	✓
Show Arrows	✓
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
Tree	
PointCloud2	✓
Status: Ok	
Topic	/table_outlier_removal
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	RGB8

Topic
sensor_msgs/msg/PointCloud2 topic to subscribe to.

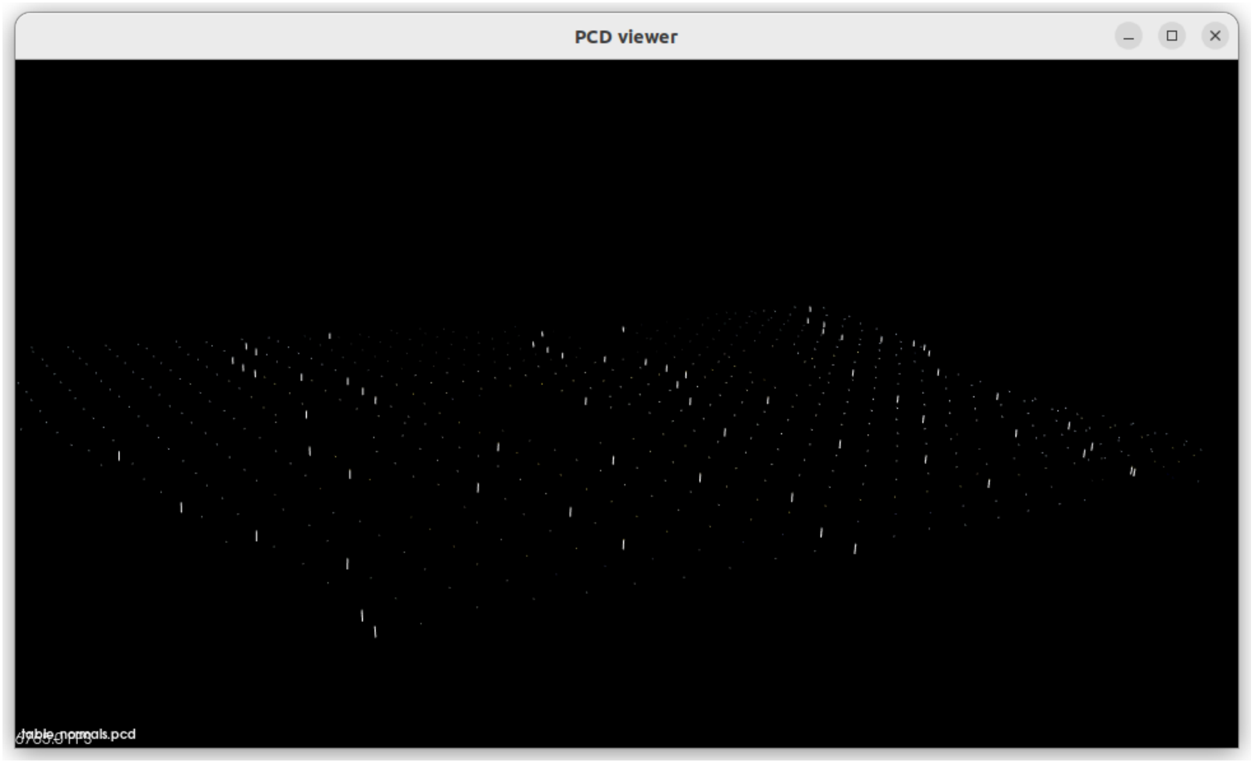
Views — Type: Orbit (rviz_defa

Current View	Orbit (rviz)
Near Clip ...	0.01
Invert Z Axis	☐
Target Fra...	<Fixed Frame>
Distance	2.53228
Focal Sha...	0.05
Focal Sha...	☐
Yaw	4.2554
Pitch	0.205399
Focal Point	0; 0; 0

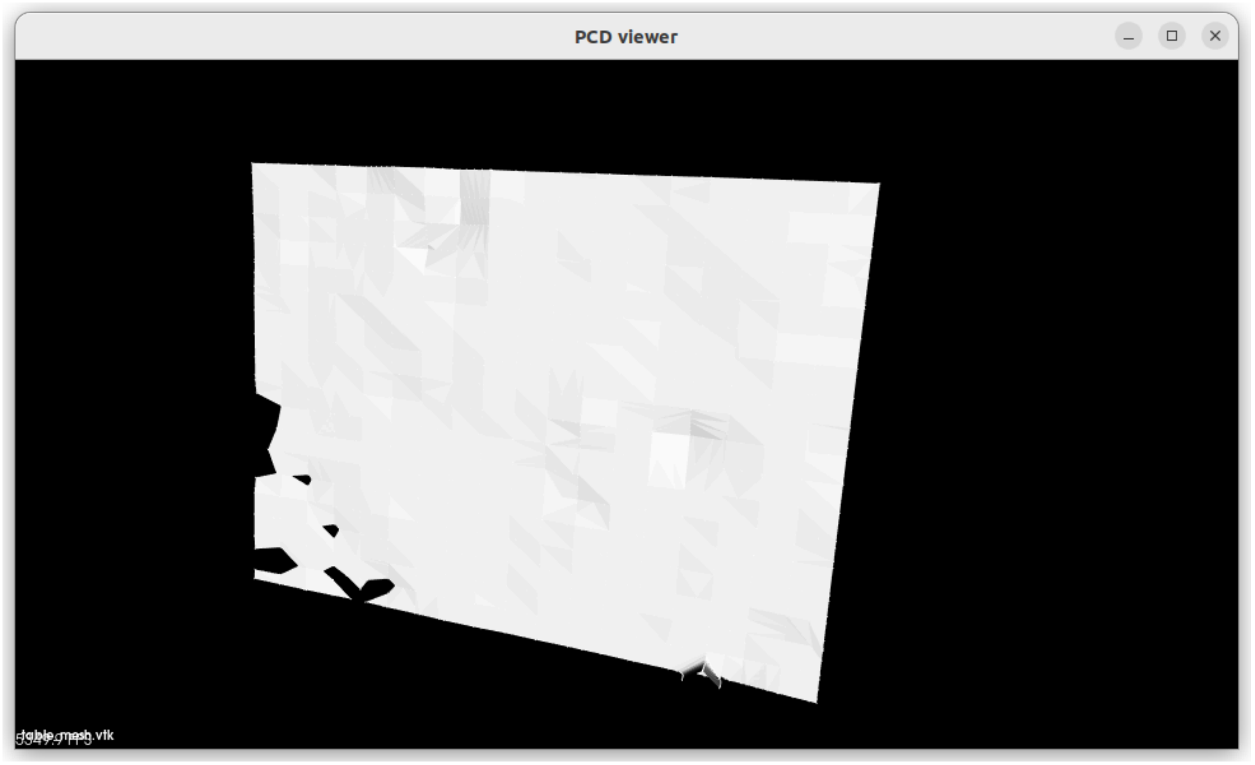
Time
ROS Time: 1729335536.74 ROS Elapsed: 2729.41 Wall Time: 1729335536.77 Wall Elapsed: 2729.41 Experimental

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

❖ Compute the normals for each point in the point cloud using the pcl_normal_estimation. This example estimates the local surface normal (perpendicular) vectors at each point. For each point, the algorithm uses nearby points (within the specified radius) to fit a plane and calculate the normal vector. Zoom in to view the normal vectors in more detail.



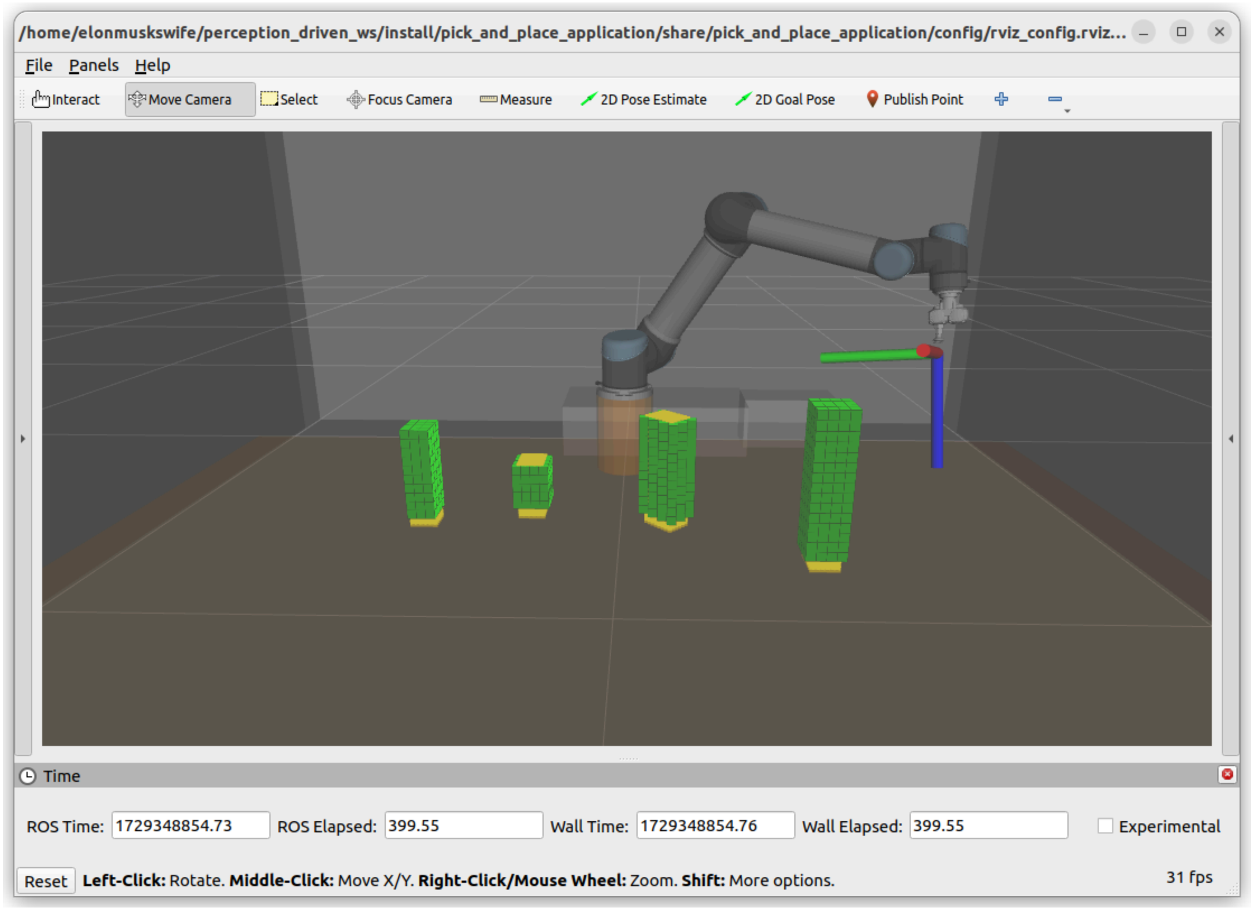
❖ Mesh a point cloud using the marching cubes reconstruction.



2. Complete Demo 1 – Perception-Driven Manipulation. To run the default simulation you need to replace files in /src/ur5_workcell_moveit2_config/config with the files from the provided archive.

Application Demo 1 - Perception-Driven Manipulation

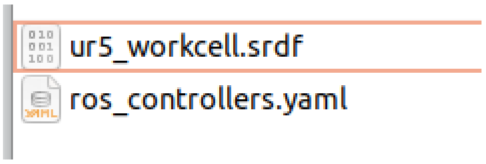
Demo 1.0-1.3 The package setup and start of the simulation


/home/elonmuskswife/perception_driven_ws/install/pick_and_place_application/share/pick_and_place_application/config/rviz_config.rviz...
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Time
Reset
31 fps

ROS Time: 1729348854.73 ROS Elapsed: 399.55 Wall Time: 1729348854.76 Wall Elapsed: 399.55 Experimental

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

These files were replaced by fixed files provided in moodle:



ur5_workcell.srdf
ros_controllers.yaml

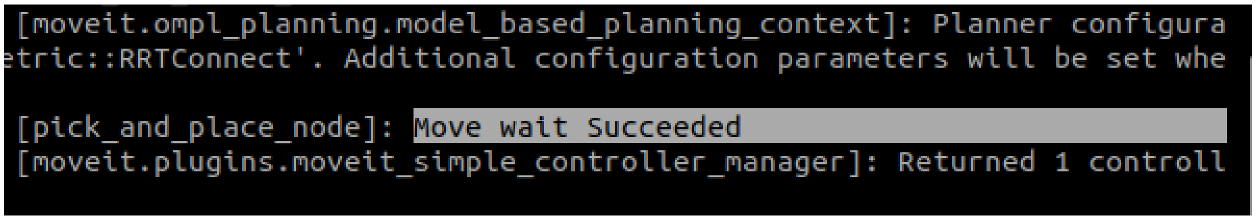
Demo 1.4-1.5: Initialization and Global Variables and move arm to wait position.

MoveIt!'s `moveit_cpp` API allows us to move the robot in various ways. With a `PlanningComponent` object, it is possible to create motion plans to move to a desired joint position, Cartesian goal or a predefined pose created with the Setup Assistant. Then we can use a `MoveItCpp` object to execute that plan. In this exercise, we will move the robot to a predefined joint pose.

Inside of the file "move_to_wait_position.cpp", we added the code for :

```
1) Set robot wait target
planning_component.setGoal(cfg.WAIT_POSE_NAME);
2) Execute the trajectory on the robot
succeeded = moveit_cpp->execute(cfg.ARM_GROUP_NAME,
plan_solution.trajectory, true);
```

The arm was moved to wait position:



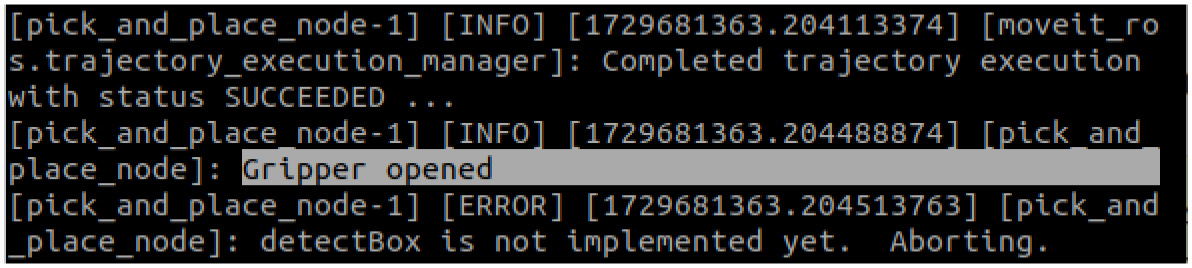
```
 [moveit.ompl_planning.model_based_planning_context]: Planner configura
etric::RRTConnect'. Additional configuration parameters will be set whe

 [pick_and_place_node]: Move wait Succeeded
 [moveit.plugins.moveit_simple_controller_manager]: Returned 1 controll
```

Note: In the pick and place parameters.yaml planner_id was changed to **RRTConnect**. Otherwise, there was an error in the run.

Demo 1.6: In this exercise, the objective is to use a "grasp action client" to send a grasp goal that will open the gripper.

```
/* Fill Code:
  * Goal:
  * - Send the grasp goal to the action server.
  * Hints:
  * - Use the "async_send_goal" method of the "grasp_action_client"
  * to send the goal to the server.
  */
 // UNCOMMENT AND COMPLETE:
 std::shared_future<GraspGoalHandle::SharedPtr> goal_fut =
grasp_action_client->async_send_goal(grasp_goal);
```

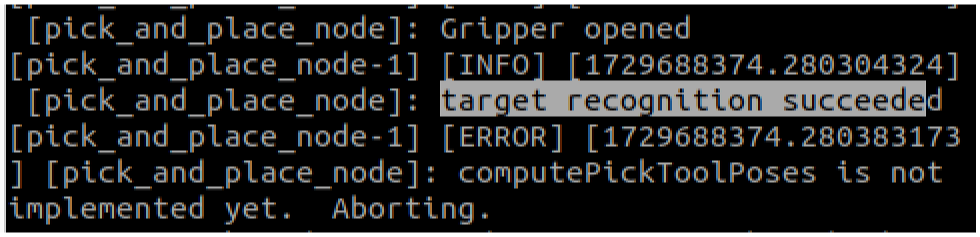
```
 /* Fill Code:
  * Goal:
  * - Confirm that a valid result was sent by the action server.
  * Hints:
  * - Use the "wait_for" method of the goal_fut future object to wait for
completion.
  * - 4 seconds is a reasonable timeout to wait for the gripper action
result
  * - "wait_for" uses std::chrono::duration to specify the timeout
  * - look up std::chrono::seconds or std::chrono_literals to specify the
time
  */
// UNCOMMENT AND COMPLETE:
 status = goal_fut.wait_for(std::chrono::seconds(4));
```



```
[pick_and_place_node-1] [INFO] [1729681363.204113374] [moveit_ro
s.trajectory_execution_manager]: Completed trajectory execution
with status SUCCEEDED ...
[pick_and_place_node-1] [INFO] [1729681363.204488874] [pick_and
place_node]: Gripper opened
[pick_and_place_node-1] [ERROR] [1729681363.204513763] [pick_and
_place_node]: detectBox is not implemented yet.  Aborting.
```

Demo 1.7: Detect Box Pick Point. The coordinate frame of the box's pick can be requested from a ROS service that detects it by processing the sensor data. In this exercise, we will learn how to use a service client to call that ROS service for the box pick pose.

```
/* DETECTING BOX PICK POSE
 Goal:
   - Get the box location in the world frame by calling the target
recognition service.
   - Return the box pose.
*/
// UNCOMMENT AND COMPLETE:
 response_fut = target_recognition_client->async_send_request(req);
```

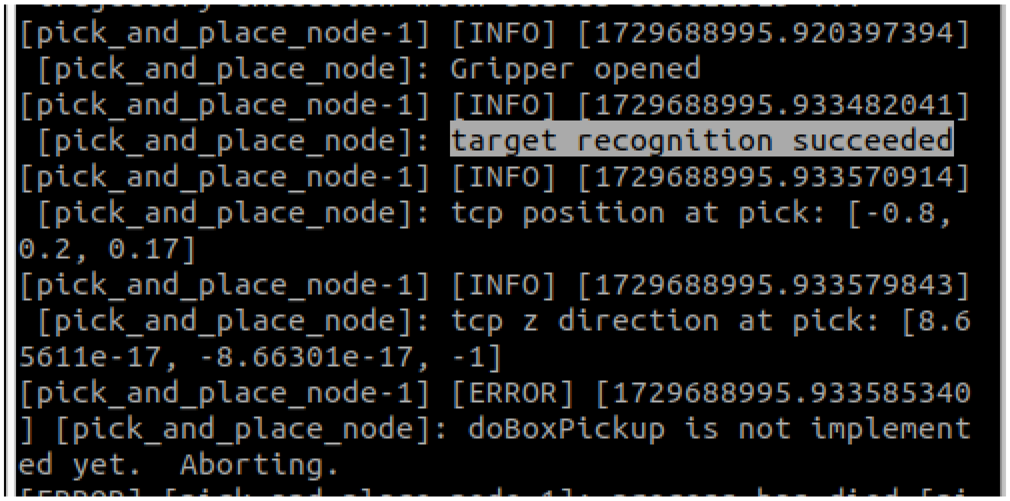


```
 [pick_and_place_node]: Gripper opened
[pick_and_place_node-1] [INFO] [1729688374.280304324]
 [pick_and_place_node]: target recognition succeeded
[pick_and_place_node-1] [ERROR] [1729688374.280383173
] [pick_and_place_node]: computePickToolPoses is not
implemented yet.  Aborting.
```

Demo 1.8: Create Pick Moves

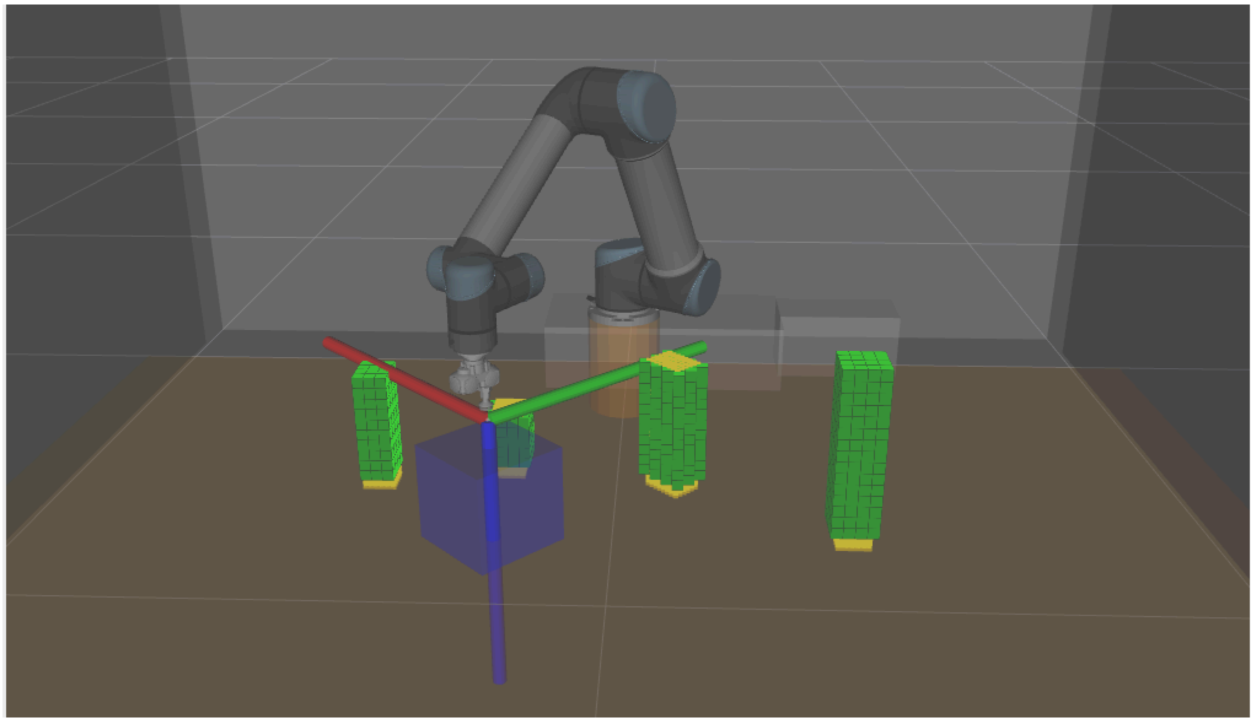
The gripper moves through three poses in order to do a pick: approach, target and retreat. In this exercise, we will use the box pick transform to create the pick poses for the TCP coordinate frame.

```
 /* Fill Code:
  * Goal:
  * - Compute the TCP pose at the box pick location.
  * Hints:
  * - Use the "setOrigin" to set the position of "tcp_at_box_tf".
  */
 box_tf = tf2::poseMsgToTransform(box_pose);
 tf2::Vector3 box_position(box_pose.position.x, box_pose.position.y,
box_pose.position.z);
 // UNCOMMENT AND COMPLETE:
 tcp_at_box_tf.setOrigin(box_position);

 /* Setting tcp orientation
  * Inverting the approach direction so that the tcp points towards the
box instead of
  * away from it.*/
 tcp_at_box_tf.setRotation(box_tf.getRotation() *
tf2::Quaternion(tf2::Vector3(1, 0, 0), M_PI));
```



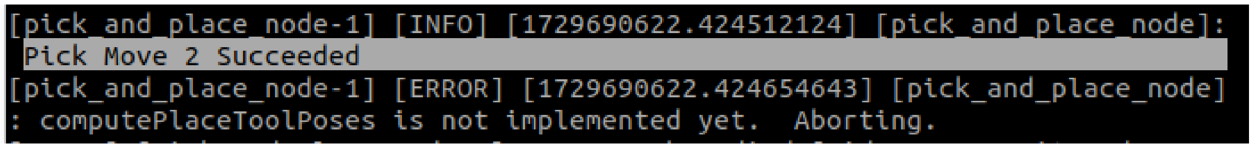
```
[pick_and_place_node-1] [INFO] [1729688995.920397394]
 [pick_and_place_node]: Gripper opened
[pick_and_place_node-1] [INFO] [1729688995.933482041]
 [pick_and_place_node]: target recognition succeeded
[pick_and_place_node-1] [INFO] [1729688995.933570914]
 [pick_and_place_node]: tcp position at pick: [-0.8,
0.2, 0.17]
[pick_and_place_node-1] [INFO] [1729688995.933579843]
 [pick_and_place_node]: tcp z direction at pick: [8.6
5611e-17, -8.66301e-17, -1]
[pick_and_place_node-1] [ERROR] [1729688995.933585340
] [pick_and_place_node]: doBoxPickup is not implement
ed yet.  Aborting.
```

Demo 1.9: Pick Up Box.
In this exercise, we will move the robot through the pick motion while avoiding obstacles in the environment. This will be accomplished by planning for each pose and closing or opening the vacuum gripper when appropriate. Also, we will demonstrate how to create a motion plan that MoveIt! can understand and solve.



```
  // UNCOMMENT AND COMPLETE:
  robot_state = moveit_cpp->getCurrentState(2);
  // UNCOMMENT AND COMPLETE:
```

```
  success = moveit_cpp->execute(cfg.ARM_GROUP_NAME,
    plan_solution.trajectory, true);
```



```
[pick_and_place_node-1] [INFO] [1729690622.424512124] [pick_and_place_node]:
 Pick Move 2 Succeeded
[pick_and_place_node-1] [ERROR] [1729690622.424654643] [pick_and_place_node]
: computePlaceToolPoses is not implemented yet.  Aborting.
```

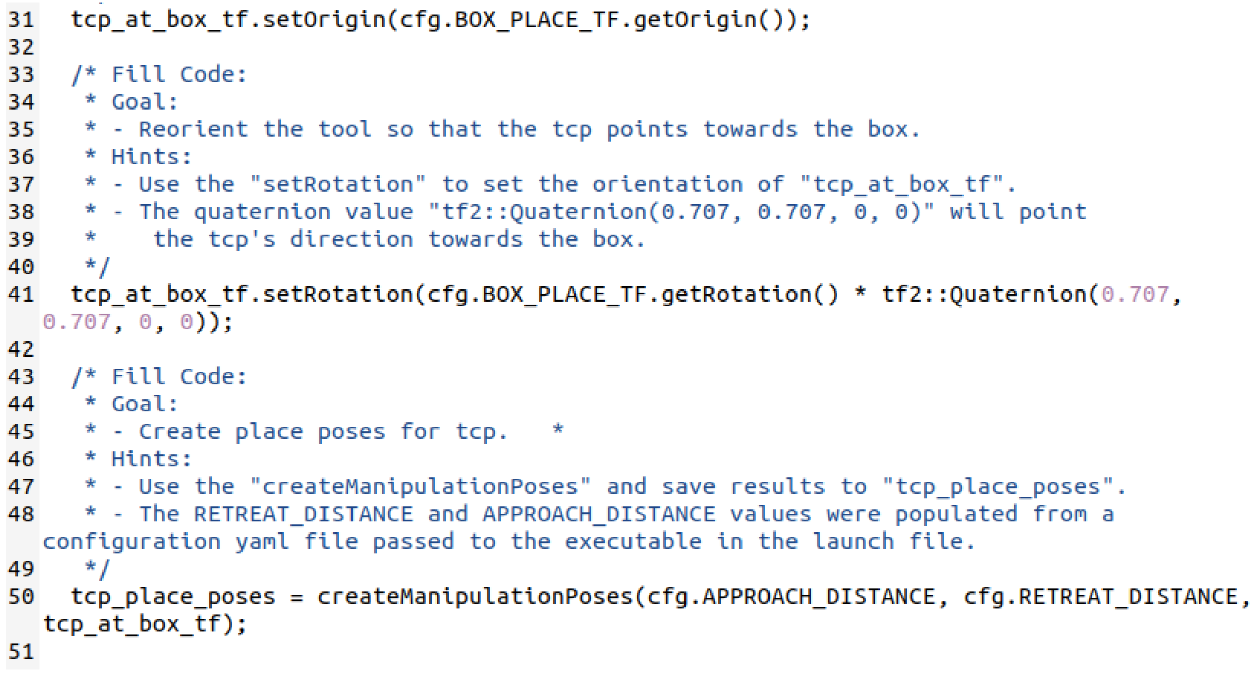
Demo 1.10: Create Place Moves.
In this part, we need to complete the code for create_place_moves.cpp.
First, as previously, we uncommented `"RCLCPP_ERROR_STREAM …"`
and then performed necessary changes.

// UNCOMMENT AND COMPLETE:
tcp_at_box_tf.setOrigin(cfg.BOX_PLACE_TF.getOrigin());

// UNCOMMENT AND COMPLETE:
tcp_at_box_tf.setRotation(cfg.BOX_PLACE_TF.getRotation() *
tf2::Quaternion(0.707, 0.707, 0, 0));

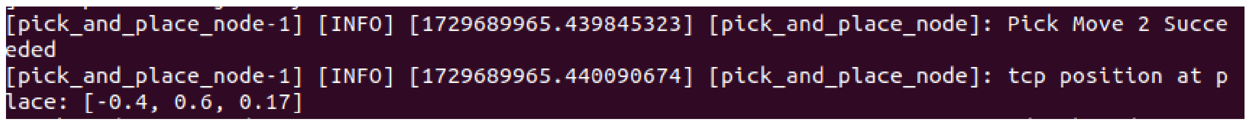
// UNCOMMENT AND COMPLETE: tcp_place_poses =
createManipulationPoses(cfg.APPROACH_DISTANCE,
cfg.RETREAT_DISTANCE, tcp_at_box_tf);

Below is the code with performed changes:



```
  tcp_at_box_tf.setOrigin(cfg.BOX_PLACE_TF.getOrigin());

  /* Fill Code:
   * Goal:
   * - Reorient the tool so that the tcp points towards the box.
   * Hints:
   * - Use the "setRotation" to set the orientation of "tcp_at_box_tf".
   * - The quaternion value "tf2::Quaternion(0.707, 0.707, 0, 0)" will point
   *     the tcp's direction towards the box.
   */
  tcp_at_box_tf.setRotation(cfg.BOX_PLACE_TF.getRotation() * tf2::Quaternion(0.707, 0.707, 0, 0));

  /* Fill Code:
   * Goal:
   * - Create place poses for tcp.   *
   * Hints:
   * - Use the "createManipulationPoses" and save results to "tcp_place_poses".
   * - The RETREAT_DISTANCE and APPROACH_DISTANCE values were populated from a configuration yaml file passed to the executable in the launch file.
   */
  tcp_place_poses = createManipulationPoses(cfg.APPROACH_DISTANCE, cfg.RETREAT_DISTANCE, tcp_at_box_tf);

```

At the end, when building and running, we've got the following result in the terminal:



```
[pick_and_place_node-1] [INFO] [1729689965.439845323] [pick_and_place_node]: Pick Move 2 Succe
eded
[pick_and_place_node-1] [INFO] [1729689965.440090674] [pick_and_place_node]: tcp position at p
lace: [-0.4, 0.6, 0.17]
```

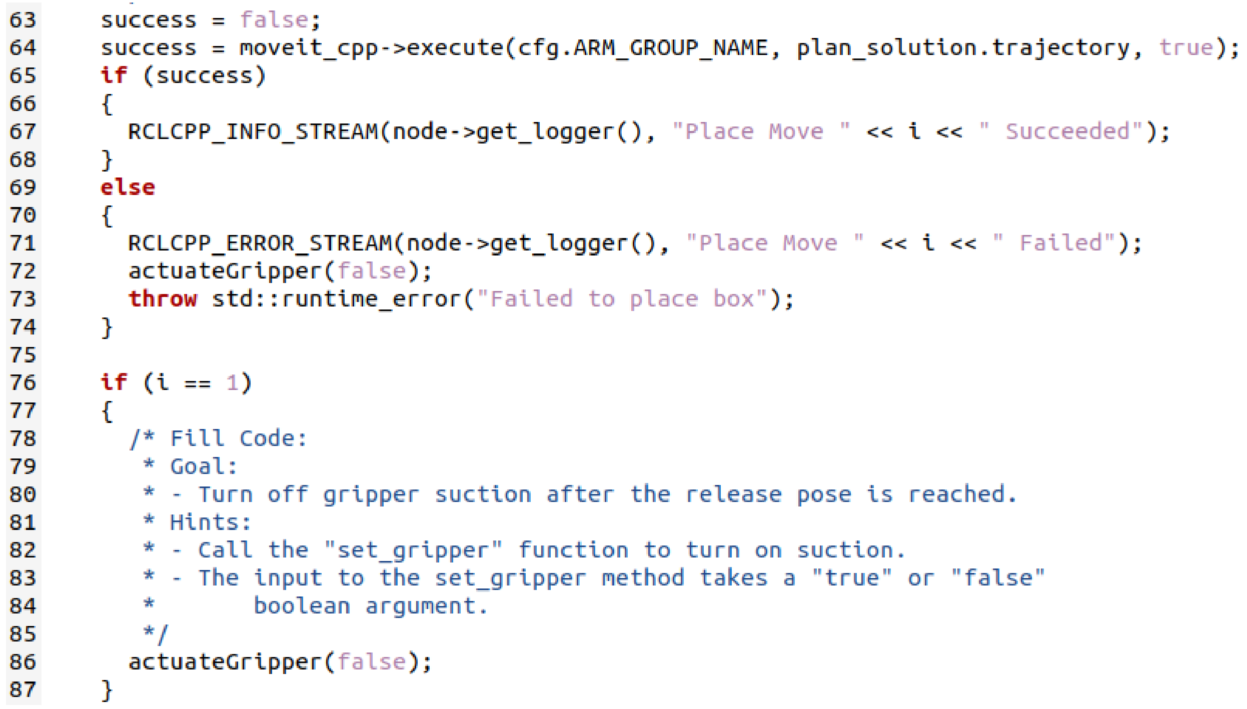
Demo 1.11: Place Box.

In this task, we performed the place box task by changing the place_box.cpp file. we uncommented `"RCLCPP_ERROR_STREAM …"` and then made the following changes:

// UNCOMMENT AND COMPLETE:
success = moveit_cpp->execute(cfg.ARM_GROUP_NAME, plan_solution.trajectory, true);

// UNCOMMENT AND COMPLETE:
actuateGripper(false);

Below is the code with changes:



```
    success = false;
    success = moveit_cpp->execute(cfg.ARM_GROUP_NAME, plan_solution.trajectory, true);
    if (success)
    {
      RCLCPP_INFO_STREAM(node->get_logger(), "Place Move " << i << " Succeeded");
    }
    else
    {
      RCLCPP_ERROR_STREAM(node->get_logger(), "Place Move " << i << " Failed");
      actuateGripper(false);
      throw std::runtime_error("Failed to place box");
    }

    if (i == 1)
    {
      /* Fill Code:
       * Goal:
       * - Turn off gripper suction after the release pose is reached.
       * Hints:
       * - Call the "set_gripper" function to turn on suction.
       * - The input to the set_gripper method takes a "true" or "false"
       *       boolean argument.
       */
      actuateGripper(false);
    }
```

Then, we build and run. After this, we see the robot completely perform the pick and place task. The video of the demo is attached below.

The video demo:

Kanagat's PC: project3_robt611.mp4 or on YouTube project3 robt611

Madina's PC: pick and place

3. Based on the previous exercises and Demo 1 prepare a 1-2 page final project proposal with **detailed task scenario** for **autonomous pick-place** of an object. Alternative project proposals can be discussed.

Proposal

Title: Transferring Snake Robot labs from ROS to ROS2

Aim: The aim of our final project is to transfer existing labs for the course "Robotics II: Control, Modeling and Learning" from ROS to ROS2. We plan to check if the current labs are feasible for transfer to ROS2 and if yes, prepare needed packages for two or more labs.

Background: Mastering Robot Operating System is essential for students and professionals working in the Robotics field, especially in this rapidly evolving robotic landscape. ROS provides a flexible and modular framework that allows the integration of software for robotics systems, making it the foundation for many real-world applications. With the release of ROS2, the system has undergone significant improvements in performance, scalability and security, which aligns more closely with today's industrial standards. Learning ROS2 will offer students hands-on experience with industry-relevant tools and will also equip them with the skills needed to tackle advanced robotic challenges. The transition from ROS to ROS2 in lab tasks is critical as it ensures that students are trained on the latest technologies, reflecting the growing demand for ROS2 expertise in sectors like automation, manufacturing or even healthcare.

Laboratory Sessions consist of Introduction to basic Linux commands and ROS/ROS2 (architecture and main concepts, including nodes, services, topics, packages, messages), continuing with modeling under ROS environment for simulation in Gazebo and visualization in RVIZ using URDF. By covering these necessary tools, students learn all the needed information for joint control of the robot end-effector. When ready to work with a real Snake Robot, students perform trajectory planning for a 3-DOF Planar Manipulator and develop the Inverse Kinematics package in MoveIt. By engaging in practical tasks such as robot control, simulation in Gazebo and motion planning with MoveIt, students gain a comprehensive understanding of robotic systems while staying ahead of technological trends that are widely adopted in the industry.

Delimitations that might be faced: Incompatibility of ROS packages; issues with message-passing systems in ROS2; control of hardware interfaces with dynamixel packages; differences in Gazebo integration and URDF models; custom messages and services; hardware communication and latency.

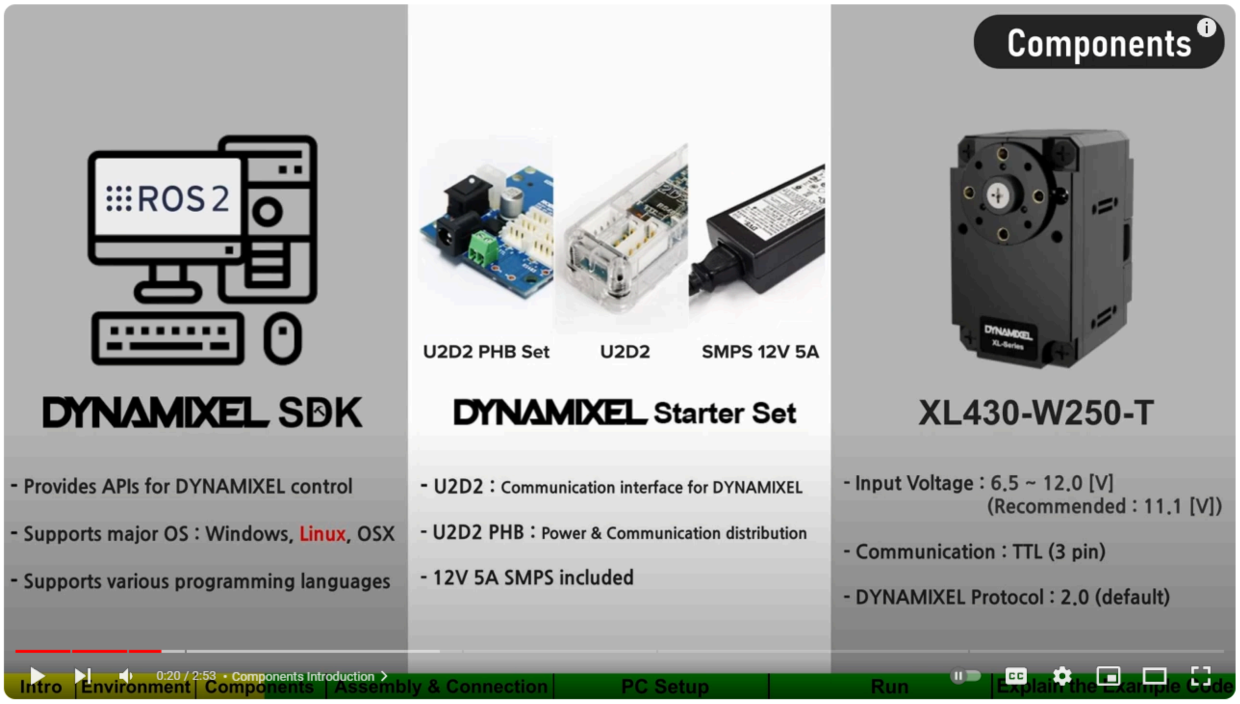
Methods: In this project, we will rely on previously developed ROS packages from the existing Git repository for Robt403 course, which contain the necessary control files for both the simulated and real 5-DOF planar robot. These packages will serve as a reference. The transition process will involve refactoring the original codebase, adapting it to ROS2 architecture and ensuring compatibility with the ROS2 middleware and communication protocols.

Additionally, we will utilize ROS2 documentation throughout the process of modifying existing custom message definitions, ensuring accurate control of the dynamixel hardware and adapting URDF models and simulation setups for the Gazebo environment in ROS2.

Below are dynamixel hardware manuals that will be consulted.

Plan:
0. Install dynamixel packages
1. move one dynamixel with ROS2
2. move all in series motors
3. pick and place application

The hardware:



DYNAMIXEL SDK

- Provides APIs for DYNAMIXEL control
- Supports major OS : Windows, Linux, OSX
- Supports various programming languages

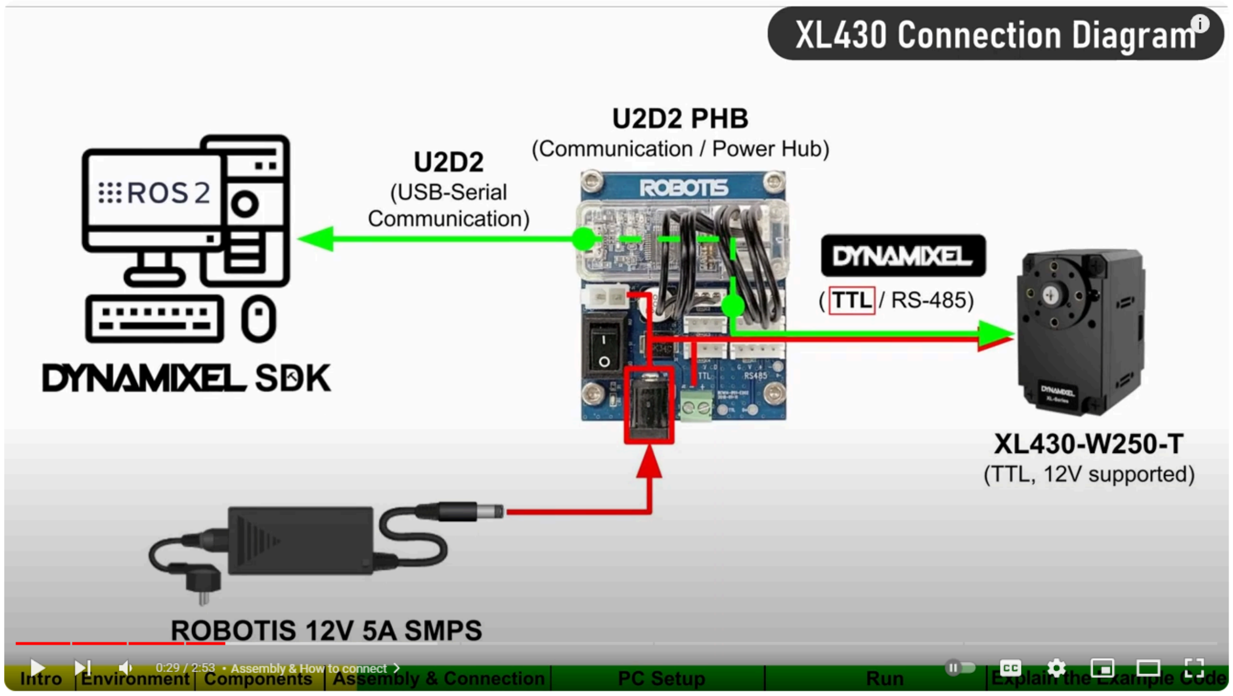
DYNAMIXEL Starter Set

- U2D2 : Communication interface for DYNAMIXEL
- U2D2 PHB : Power & Communication distribution
- 12V 5A SMPS included

XL430-W250-T

- Input Voltage : 6.5 ~ 12.0 [V] (Recommended : 11.1 [V])
- Communication : TTL (3 pin)
- DYNAMIXEL Protocol : 2.0 (default)

Connect the hardware:



The software:

- ROS2 on ubuntu 22.04

- Dynamixel ROS2 package: https://github.com/dynamixel-community/dynamixel_hardware
- Set topics, subscribers, functions to deal with motor data, get position service, port handling (source DYNAMIXEL Quick Start Guide for ROS 2)

Conclusion: To maintain clarity and completeness, we will ensure that the documentation of the new ROS2 packages follows established standards, allowing future users to easily understand and modify the codebase as needed. This systematic approach will help ensure the successful transition from ROS to ROS2 while adhering to the project's technical and educational objectives.