Getting Familiar with Controlling a Mobile Manipulator

Zackory Erickson

Carnegie Mellon University
The Robotics Institute

16-762: Mobile Manipulation
Spring 2026

Today

- Getting familiar with controlling a mobile manipulator
    - (Features of Stretch 3, code, and hardware)
- Begin learning Stretch 3 code base

Python code

- We will be looking at a lot of Python code today.
- You should be familiar with the Python programming language.
- You do not need to be familiar with programming a robot (hence this lecture)

Stretch 3 at a glance

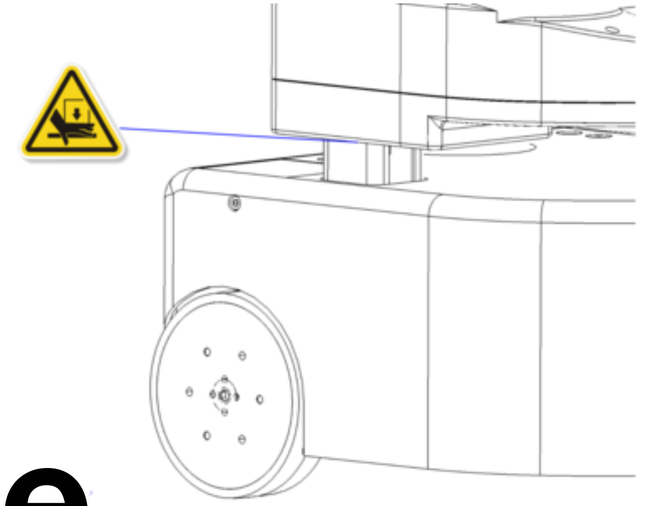
- Mobile manipulator
- Telescoping arm - 50cm
- Prismatic lift - 110cm
- Differential drive base with a small footprint
- 3-axis dexterous wrist
- Built-in Linux computer and battery
- Only 51 lbs
- Plenty of sensors and features

Stretch 3 features

- Compliant gripper
- Back-drivable motors with effort sensing
- RealSense depth camera on pan/tilt head
- Laser range finder
- IMUs
- Speaker/microphone array
- Open source software

Stretch 3 software

- Simple Python interface
- ROS integration
- Calibrated URDF
- Open source demo code for:
    - Autonomously grasping objects
    - Handing objects to people
    - 3D Mapping
    - Localization and navigation
    - https://github.com/hello-robot/stretch_ros2/tree/humble/stretch_demos/stretch_demos



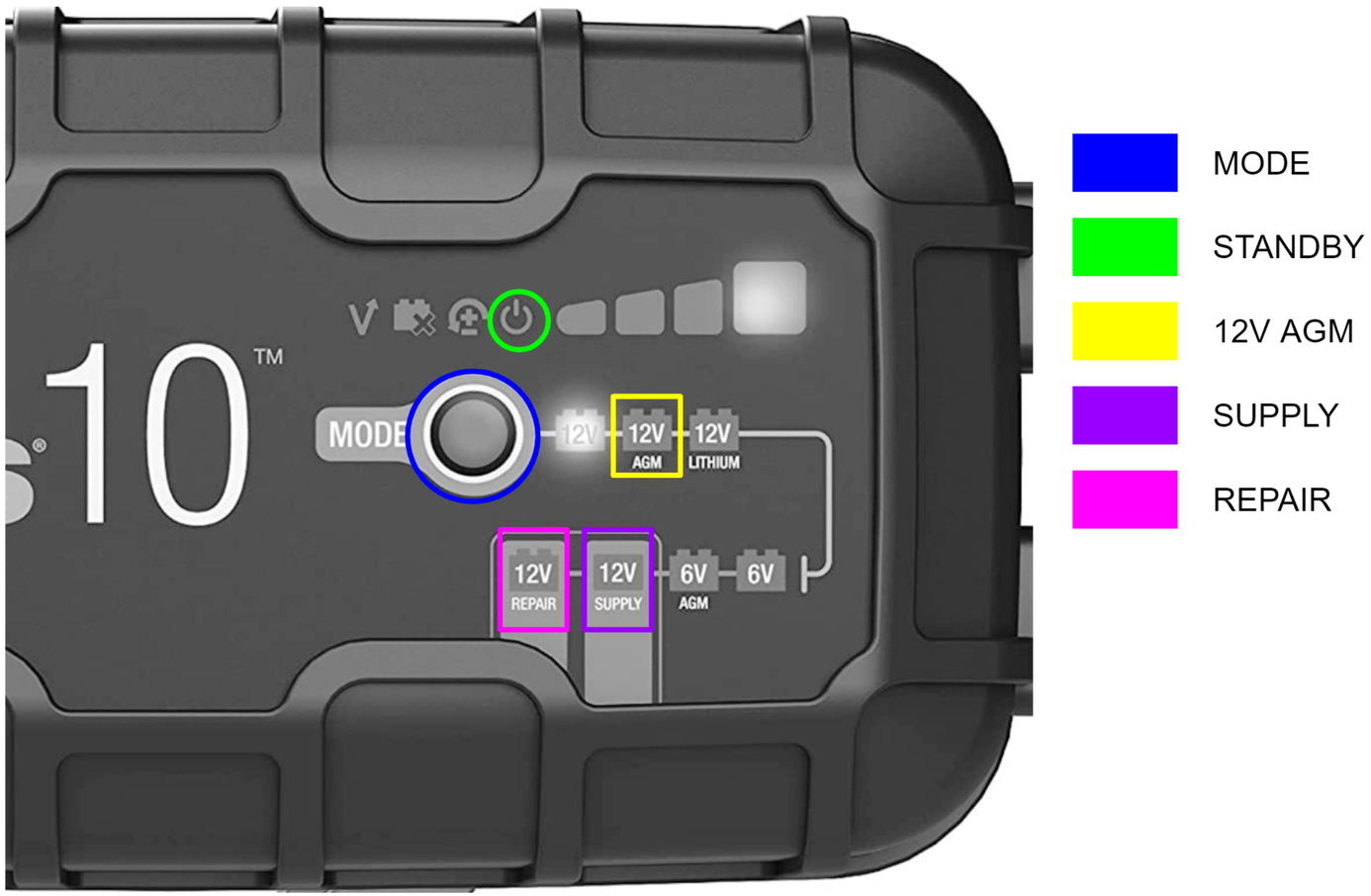
Have you read the safety manual yet?

https://docs.hello-robot.com/0.3/hardware/safety_guide/

Battery Maintenance

- Very important!
- Not Lithium Ion
- Max 2 hour use unplugged.
- When you are using the robot, put it in SUPPLY, otherwise 12V AGM.

Robot is not in use	12V AGM
Robot is in use	SUPPLY



GitHub Repo for Examples

- Today we will go through a bunch of simple code examples to demonstrate basic functionality of the robot.
- This lecture is like a “cheat sheet” for common things you may want to do on the robot.
- https://github.com/Zackory/mm2026/tree/main/stretch_python
- Terminal command:
  git clone https://github.com/Zackory/mm2026.git

How to connect into the robot

I will share SSH and anydesk (remote desktop) login details for both robots next week during the lab.

Stretch code interface

- Stretch 3 API tutorials: https://docs.hello-robot.com/latest/python/moving/
- Stretch 3 ROS tutorials: https://docs.hello-robot.com/latest/ros2/getting_started/

Setup a folder on the robot for your own code

```
cd 16762/
mkdir your_name
cd your_name

# Create a Python virtual env (that way your pip installs don’t break other people’s code)
python3 -m venv env
source env/bin/activate
pip3 install --upgrade pip
pip3 install hello-robot-stretch-body
pip3 install numpy==1.26.4 opencv-contrib-python==4.10.0.84 opencv-python==4.11.0.86

# Create your own GitHub repo. Below is mine.
git clone https://github.com/Zackory/mm2026.git
```

NOTE

Run the below lines of code every time you log into the robot or open a new terminal window:

```
cd 16762/your_name
source env/bin/activate
```

When you turn on the robot

Put the battery into SUPPLY mode.

Run:

```
cd 16762/your_name
source env/bin/activate
stretch_robot_home.py
```

Helpful Stretch Python Scripts

- https://docs.hello-robot.com/latest/developing/cli/
- A couple of examples:
    - `stretch_robot_system_check.py`
    - `stretch_robot_home.py`
    - `stretch_realsense_visualizer.py`
    - `stretch_respeaker_test.py`
    - `stretch_rp_lidar_jog.py`
    - Here is the underlying code: https://github.com/hello-robot/stretch_body/tree/master/tools/bin

Teleoperation

- XBox controller:
  stretch_xbox_controller_teleop.py

- Keyboard:
  stretch_robot_keyboard_teleop.py

You can find code scripts for the following slides on the course GitHub:

https://github.com/Zackory/mm2026/tree/main/stretch_python

Simple robot motion

```
import time
import stretch_body.robot
robot = stretch_body.robot.Robot()
robot.startup()

# Move to full extension
robot.arm.move_to(0.5)
robot.push_command()
robot.arm.wait_until_at_setpoint() # Wait for motion to complete

# Move the arm backwards by 0.3 meters
robot.arm.move_by(-0.3)
robot.push_command()
time.sleep(4.0)

# Print the current joint angle
print(robot.arm.status['pos'])

robot.stop()
```

01_simplemotion.py

Some more motors

```
robot.arm.move_to(0.5) # Move telescoping arm
robot.lift.move_to(0.4) # Move lift
robot.head.move_by('head_pan', np.radians(45)) # Move head pan
robot.head.move_by('head_tilt', np.radians(45)) # Move head tilt
robot.base.translate_by(0.2) # Move robot base 0.2 meters forward
robot.base.rotate_by(np.radians(30)) # Rotate base by 30 degrees
# robot.base.set_rotational_velocity(v_r=0.1) # You can also set
base rotational velocity

# Dexterous wrist and gripper
robot.end_of_arm.move_to('wrist_yaw', np.radians(30))
robot.end_of_arm.move_to('wrist_pitch', np.radians(30))
robot.end_of_arm.move_to('wrist_roll', np.radians(30))
robot.end_of_arm.move_to('stretch_gripper', 50)
```

02_motors.py

Motion runstop

```
import stretch_body.robot
robot = stretch_body.robot.Robot()
robot.startup()

robot.arm.move_to(0.0)
robot.push_command()
robot.arm.wait_until_at_setpoint()
robot.arm.move_to(0.5)
robot.push_command()

# Runstop motion midway through the motion
input('Hit enter to runstop motion')
robot.pimu.runstop_event_trigger() # pimu = Power+IMU
robot.push_command()

input('Hit enter to disable runstop')
robot.pimu.runstop_event_reset()
robot.push_command()

robot.stop()
```

03_runstop.py

Splined trajectories

```
# Setup imports/robot as before

# Define the waypoints
times = [0.0, 10.0, 20.0]
positions = [robot.arm.status['pos'], 0.45, 0.0]
velocities = [robot.arm.status['vel'], 0.0, 0.0]

# Create the spline trajectory
for waypoint in zip(times, positions, velocities):
    robot.arm.trajectory.add(waypoint[0], waypoint[1], waypoint[2])

# Begin execution
robot.arm.follow_trajectory()
robot.push_command()
time.sleep(0.1)
```

04_spline.py

```
# Wait until completion
while robot.arm.is_trajectory_active():
    print('Execution time: %f' % robot.arm.get_trajectory_time_remaining())
    time.sleep(0.1)
```

Safety: contact models

```
# Setup imports/robot as before

# Stops arm at 30% and lift at 50% max current detected (e.g. collides with a person)
robot.arm.move_to(0.5, contact_thresh_pos=30, contact_thresh_neg=-30)
robot.lift.move_to(0.9, contact_thresh_pos=50, contact_thresh_neg=-50)

robot.push_command()
robot.arm.wait_until_at_setpoint(timeout=5.0)
robot.lift.wait_until_at_setpoint(timeout=5.0)

robot.stop()
```

05_contact.py

Robot Operating System (ROS)

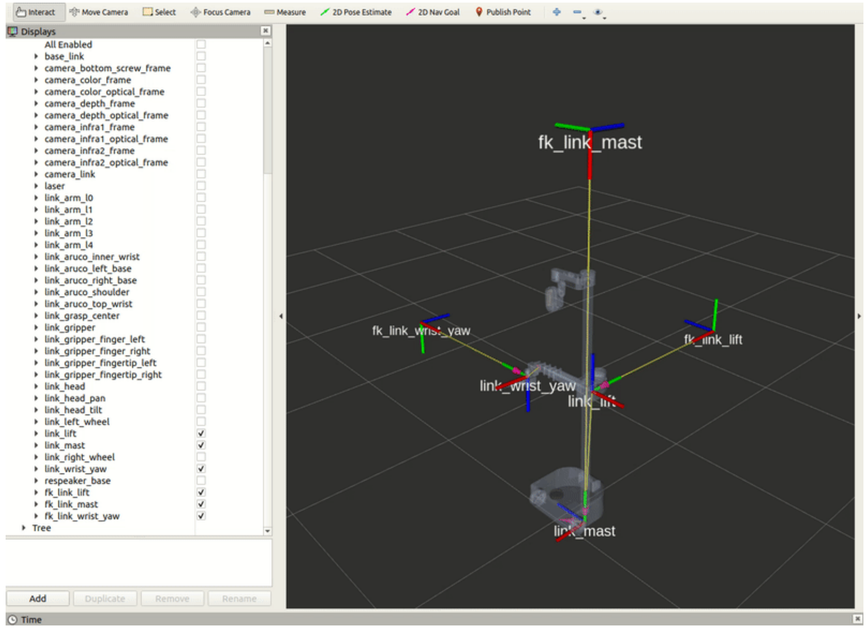
- Want to learn ROS? Here are tutorials to help: https://docs.ros.org/en/jazzy/Tutorials.html
- Instead, today I am just going to show you some helpful code examples to do advanced things on Stretch using ROS 2.
- First, launch the below command in a new terminal window/tab:
- `ros2 launch stretch_core stretch_driver.launch.py`

Transformations (forward kinematics)

```
import rclpy, tf2_ros
from rclpy.node import Node
import hello_helpers.hello_misc as hm

class FrameListener(Node):
    ... (next slide)

node = hm.HelloNode.quick_create('transformations')
fl = FrameListener()
node.move_to_pose({'joint_arm': 0.5}, blocking=False)
rclpy.spin(fl)
```



06_transformations.py

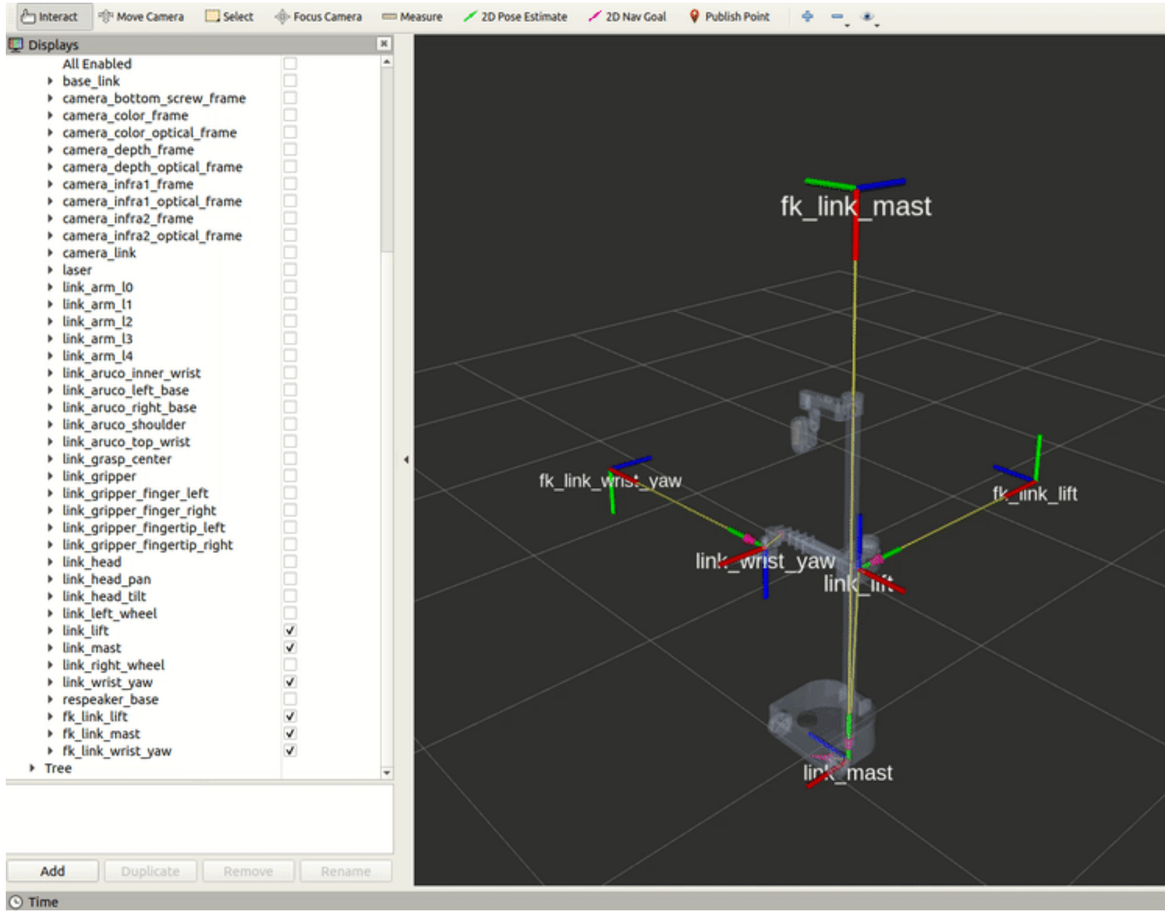
Transformations (forward kinematics)

```
class FrameListener(Node):
    def __init__(self):
        super().__init__('stretch_tf_listener')
        self.from_frame_rel = 'base_link'
        self.to_frame_rel = 'link_gripper_fingertip_left'
        self.tf_buffer = tf2_ros.Buffer()
        self.tf_listener = tf2_ros.TransformListener(self.tf_buffer, self)
        self.timer = self.create_timer(0.5, self.on_timer)

    def on_timer(self):
        try:
            t = self.tf_buffer.lookup_transform(self.to_frame_rel,
                                    self.from_frame_rel, rclpy.time.Time())
            print('XYZ:', t.transform.translation)
            print('Quaternion:', t.transform.rotation)
        except tf2_ros.LookupException as e:
            self.get_logger().info(f'Transform not ready: {e}')
```

What is the ‘string’ name for each robot link?

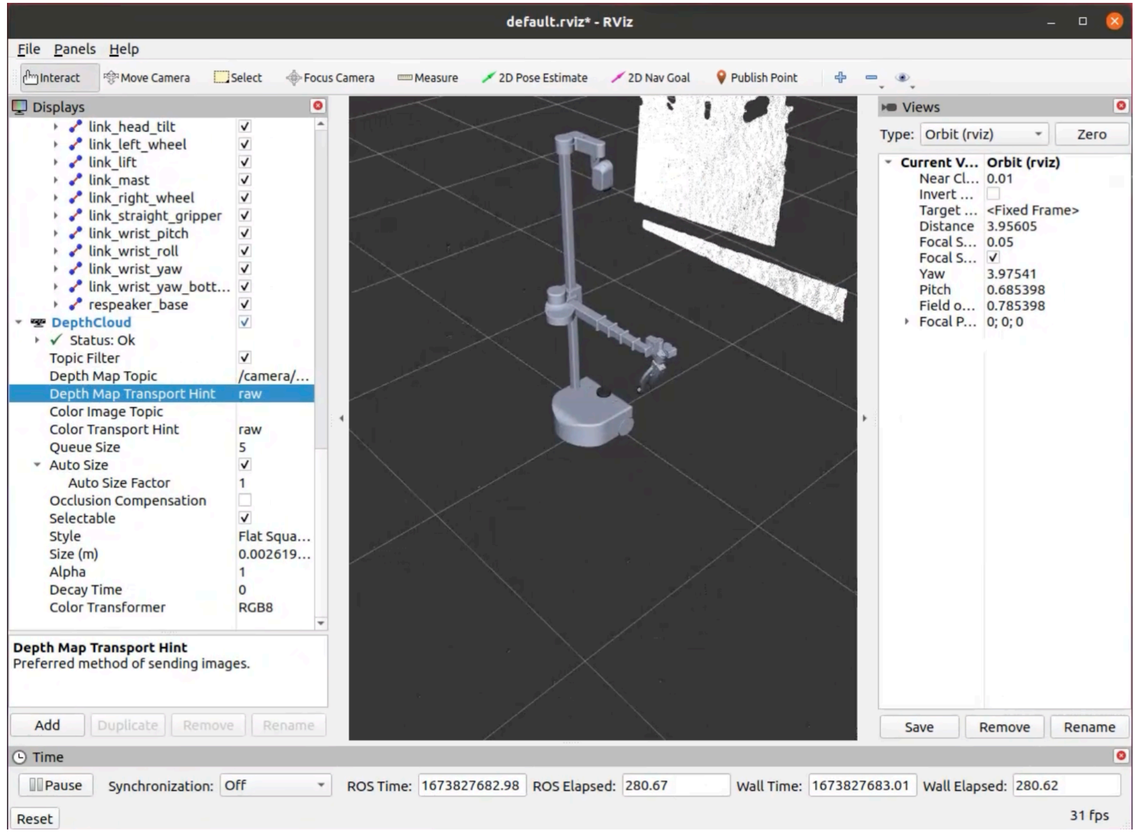
- `'base_link'`
- `'link_gripper_fingertip_left'`
- Rviz!



Rviz is your friend

- `ros2 run rviz2 rviz2 -d \`ros2 pkg prefix --share stretch_calibration\`/rviz/stretch_simple_test.rviz`

- Let's look at the list of all TF2 frames (that you can use for forward/inverse kinematics)



Capture an RGB-D image

- First, launch the below command in a new terminal window:
- `ros2 launch stretch_core d435i_low_resolution.launch.py`

Capture an RGB image

```
import rclpy, cv2
from rclpy.node import Node
from sensor_msgs.msg import Image
from cv_bridge import CvBridge

class ImageSubscriber(Node):
    def __init__(self):
        super().__init__('image_subscriber')
        self.sub = self.create_subscription(Image, '/camera/color/image_raw', self.callback, 10)
        self.bridge = CvBridge()

    def callback(self, msg):
        image = self.bridge.imgmsg_to_cv2(msg, 'bgr8')
        # Do something with your image
        cv2.imshow('image', image)
        cv2.waitKey(3)

rclpy.init()
image_sub = ImageSubscriber()
rclpy.spin(image_sub)
```

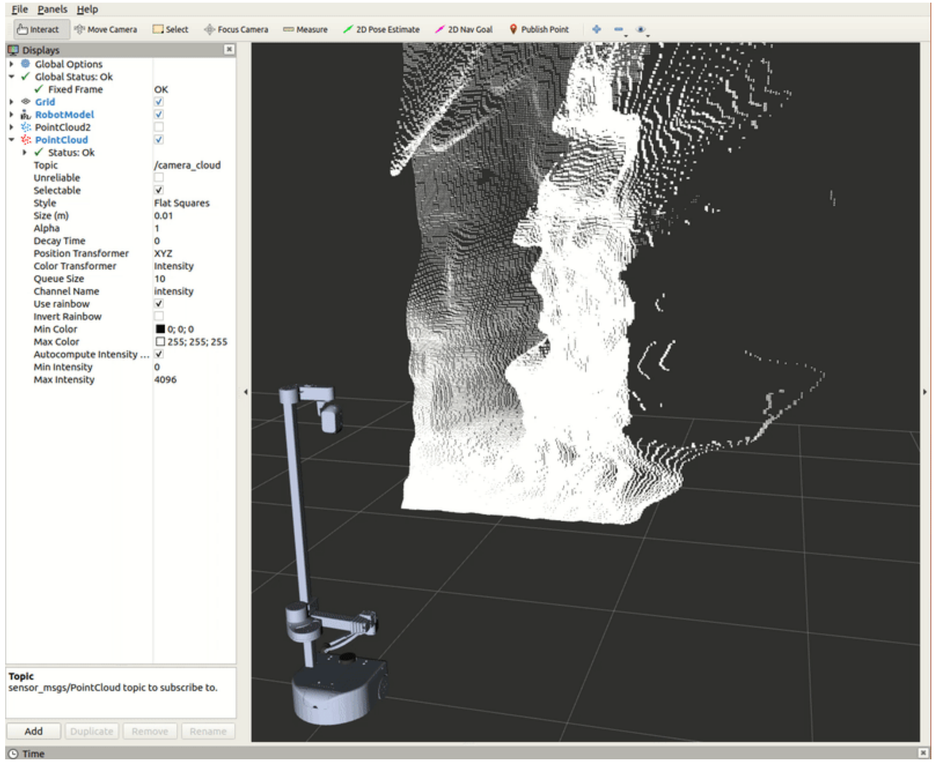
07_rgb.py

Capture a point cloud

```
import rclpy, tf2_ros, tf2_sensor_msgs
import numpy as np
from rclpy.node import Node
from sensor_msgs.msg import PointCloud2
import sensor_msgs_py.point_cloud2 as pc2

class PCSubscriber(Node):
    ... (next slide)

rclpy.init()
node = PCSubscriber()
rclpy.spin(node)
```



08_pointcloud.py

Capture a point cloud

```
class PCSubscriber(Node):
    def __init__(self):
        super().__init__('pointcloud_subscriber')
        self.tf_buffer = tf2_ros.Buffer()
        self.tf_listener = tf2_ros.TransformListener(self.tf_buffer, self)
        self.sub = self.create_subscription(PointCloud2, '/camera/depth/color/points',
                                            self.callback, 10)

    def callback(self, msg):
        try:
            transform = self.tf_buffer.lookup_transform('base_link',
                                        'camera_color_optical_frame', rclpy.time.Time())
            pc_transformed = tf2_sensor_msgs.do_transform_cloud(msg, transform)
            pc_numpy = pc2.read_points_numpy(pc_transformed, field_names=('x', 'y', 'z'),
                                             skip_nans=True)
            print('First 3 points:', pc_numpy[:3])
        except tf2_ros.LookupException as e:
            self.get_logger().info(f'Transform not ready: {e}')
```

Computer vision examples

- https://github.com/hello-robot/stretch_ros2/tree/humble/stretch_deep_perception/stretch_deep_perception
- Detect faces
- Detect 3D objects
- Track AR fiducial markers
- Estimate human body pose

Speech to text

- First, launch the below command in a new terminal window/tab:
- `ros2 launch respeaker_ros2 respeaker.launch.py`

Speech to text

```
import rclpy, sys
from rclpy.node import Node
from speech_recognition_msgs.msg import SpeechRecognitionCandidates

class SpeechSub(Node):
    def __init__(self):
        super().__init__('speech_sub')
        self.sub = self.create_subscription(SpeechRecognitionCandidates,
'speech_to_text', self.callback, 10)

    def callback(self, msg):
        transcript = ' '.join(map(str, msg.transcript))
        print(transcript)
        # Do something with the text
        sys.exit(0)

rclpy.init()
speech_sub = SpeechSub()
rclpy.spin(speech_sub)
```

09_speech.py

All done?

- Press Ctrl+C on the terminal windows to close down ROS.

- Reminder: Stretch 3 ROS tutorials https://docs.hello-robot.com/latest/ros2/getting_started/

Voice teleoperation of mobile base

- A neat example to try on your own.
- https://docs.hello-robot.com/latest/ros2/voice_teleop/

First Lab Next Week (Wednesday)

- Bring your laptop to class!
- No teams for the first lab.
- Getting connected to the robot
- Setting up version control (GitHub), shared robot so you should expect your code will be deleted after you log off the robot.
- Write a script to perform a series of simple motions and sensor capture.