CS 237B: Principles of Robot Autonomy II
Section 3: Turtlebot gets an Upgrade

Finally, our Turtlebot can physically grab objects.

Our goals for this week's section is to:

1. interface with Raspberry Pi and setup ROS compatible gripper nodes
2. upgrade last week's digit detector
3. teleop the Turtlebot to grasp the flag with grippers and deliver it to a desired location!

By the end of the section, you should have a working demo of the Turtlebot physically delivering the flag. The TAs will check off your group as soon as you can show this demo.

1 Background

So far our food delivery has been virtual – the robots came back home empty-handed, without physical objects. Starting this section, our robots finally get the grippers! This section is designed to make sure you understand the basics of operating the gripper with ROS and also give you a chance to learn more about the Raspberry Pi.

2 Environment Setup

SCPD students: Because you will be using the gripper in simulation, there is no need to set up the gripper hardware. Simply pull the latest changes to the `asl_turtlebot` repo and skip to the next section.

On-campus students: We will download required files to Raspberry Pi. There's one problem: Raspberry Pi is not connected to the Internet! Look for the bright yellow Ethernet cable on the Turtlebot. That is the communication line between Jetson ('robot' you're ssh'ing into) and Raspberry pi. Other than the phone cable, we don't have any way to log into Raspberry Pi. So our plan of attack to update the software stack necessary for the grippers is by (1) ssh'ing into Jetson (2) ssh in to Raspberry Pi from Jetson.

IMPORTANT: Please make sure that you follow the steps sequentially to avoid any mistakes. Also, do not add/remove any other files to/from the Raspberry Pi. Otherwise, you may not be able to use Turtlebot's camera.

First, ssh into the robot as usual and clone the repo:

```
ssh aa274@<robot name>.local
cd ~
git clone https://github.com/PrinciplesofRobotAutonomy/CS237B_Section3.git
```

The repo contains the following files:

```
1. pigpio.tar
2. gripper_subscriber.py
3. gripper_publisher. py
4. turtlebot3_bringup_jetson_pi_cs237b.launch
5. camera_transform_relay.py
6. setup_pi.sh
7. setup_jetson.sh
8. pigpiod.service.d.tar
```

Next, we transfer the files to Raspberry Pi. To do this, we need to known the name of your Pi, which you can find here. Transfer the necessary files to your Raspberry and configure Jetson using the following commands:

```
cd CS237B_Section3
scp gripper_subscriber.py setup_pi.sh aa274@<Raspi Name>.local:
scp pigpiod.service.d.tar pigpio.tar aa274@<Raspi Name>.local:
```

We will move rest of the files accordingly. You may make edits to `./setup_jetson.sh` to update your repo name and use the script name, or make changes manually.

```
% Putting this in grey background since students skip the instructions otherwise!!
Copy gripper_publisher.py and camera_transform_relay.py to your scripts folder
Copy turtlebot3_bringup_jetson_pi_cs237b.launch to your launch folder
```

We don't need `CS237B_Section3` folder anymore. Clean up the repo using:

```
cd ..
rm -rf CS237B_Section3
```

Problem 1: You will notice a new script called `camera_transform_relay.py` in your `scripts` folder. What do you think is the purpose of this script?

Now, we can ssh into the Raspberry Pi:

```
ssh aa274@<Raspi Name>.local
```

IMPORTANT: Do not try to ssh into the Pi from your computer. This is a common mistake and very easy to forget.

Once we're inside the Rasberry Pi, there are a few things we need to do. First, we need to install `pigpio` library. `pigpio` is a library for Raspberry Pi that allows control of the General Purpose Input Outputs (GPIO) using Python.

```
./setup_pi.sh
```

This command (i) copies **gripper_subscriber** to our scripts folder, (ii) extracts the necessary files and (iii) installs `pigpio` library. This could take a few minutes.

Clean up our mess using:

```
rm -rf pigpio.tar
rm -rf PIGPIO
rm -rf setup_pi.sh
rm -rf pigpiod.service.d.tar
```

Now we're done! Let's get back to our robot.

```
logout
```

3 Upgrade to 3-digit detector

In the previous section, most of you have observed many false positives in the digit detection. For example, the detector frequently confused fences with 1s or 7s. This is of course undesirable.

One way to make detection more robust (although this means training is more difficult) is to introduce some more structure into the number plates. For example, detecting 3-digit numbers is easier, because the detector needs to see 3 numbers to accept it as a detection. We could also add a frame to the plate to make it easier to identify.

We did both! Luckily, it is not a lot of changes. You will just need to update the neural network model, its weights and one function in the detector script.

First, go to the scripts folder in the robot:

```
cd ~/catkin_ws/src/<your project folder name>/scripts/
```

Again, being careful with the last line, just like you did in the previous section, run the following 4 lines:

```
git clone http://github.com/PrinciplesofRobotAutonomy/CS237B_Section2.git
mv CS237B_Section2/* .
mv anpr_weights.npz ../tfmodels/anpr_weights.npz
rm -rf CS237B_Section2
```

Next, edit `detector_mobilenet.py`. Add

```
from anpr_common import LENGTH as PLATE_LENGTH
```

and replace whatever you have in the "load_object_labels" function for the "digits" case with

```
object_labels = {i: str(i).zfill(3) for i in np.arange(len(CHARS)**PLATE_LENGTH)}
```

This is to make sure each object ID is assigned to its corresponding 3-digit string.

SCPD students: Download the plate image files from here:
https://stanford.box.com/s/fanymvsq00d42iod1s4dr63pa4wylpbg

Use these to replace your digit textures from the previous section. A flag pole has been added to the `project_city.world` Gazebo environment. Due to issues with the Gazebo friction/contact simulation, the flag pole is attached to a base which has zero friction so that the turtlebot can push it easily along the ground.

On-campus students: Finally, just check if there are subscribed topics whose names include "raspicam_node". If there are, then replace "raspicam_node" with "camera_relay". That's it!

4 Playing with the gripper

We're about to play with our robot. Run the bringup script on the turtlebot:

```
ssh aa274@<robot name>.local
roslaunch asl_turtlebot turtlebot3_bringup_jetson_pi_cs237b.launch
```

Problem 2: You will notice that we are using a new launch file. Identify the difference between this launch file and the `turtlebot3_bringup_jetson_pi.launch`. Briefly comment on what changes were made.

Now, run `gripper_publisher.py` to test the gripper using the command:

```
python gripper_publisher.py
```

Press '0' to open the gripper and press '1' to close the gripper. But with this publisher, you have so much control over the grippers!

Problem 3: Take a look at `gripper_publisher.py`to understand how it sends control commands to the gripper. Based on the topic and the type of message, write an one-line instruction to give the gripper a custom request (between 1 and 100) using the `rostopic pub` command. Include this command in your writeup.

You're ready to go out to the field and grab objects.

5 Object Gripping

The final task for this section is to use the 3-digit detector to detect the numbers on the flag and navigate towards it manually. Once you are close to the flag, switch to teleop and align the flag within your gripper.

Once the gripper is in position, you can use the `gripper_publisher` to grab the object and deliver the flag. Try to return the flag to its original position and release it from the gripper. While this is the end of the project, remember that your robot will have to do all of this autonomously for your final project – what functionality will need to be added to the robot?