Fall 2019
CS 480/580: Intelligent Mobile Robotics
Assignment 3 (9% of total grade)
Due time: Tuesday 10/25 11:59 PM

GOAL

Students will learn how to use ROS packages for SLAM, localization, and navigation, and how to create programs to drive a physical robot to visit specific positions. The first goal of this assignment is to give students first-hand experience of working on real robots.

Students will implement a simple interface for spoken language-based human-robot interaction. The goal is to help students learn to implement a minimal set of components required by a robot spoken dialog system, and how this dialog-based interface connects to robot executors.

For technical questions, please first post on Piazza and leave it there for at least 24 hours. You are indirectly helping other people by doing so. If no one knows the answer, please email the instructor.

INSTRUCTIONS (2 parts)

PART 1.

Part 1 can be done using either C++ or Python (no preference). Students will first use ROS package, GMapping, to build a map of the lab, and then do autonomous navigation: let the robot visit the four corner areas of the lab one after another, and then move out of the lab to the corridor.

1. Turn on the robot (both computer and robot base).
2. Open a terminal and run the following command to start the robot base driver. The "screen" part is optional and enables more printouts that can be useful for debugging.

   ```
   roslaunch turtlebot_bringup minimal.launch --screen
   ```

3. Open another terminal and run the following command (for gmapping, which is a ROS implementation of FastSLAM).

   ```
   roslaunch turtlebot_navigation gmapping_demo.launch --screen
   ```

4. Open another terminal and run the following to start 'rviz' for visualization:

   ```
   roslaunch turtlebot_rviz_launchers view_navigation.launch --screen
   ```

5. Finally, in a new terminal, run the following command for teleoperation.

   ```
   roslaunch turtlebot_teleop keyboard_teleop.launch --screen
   ```

6. Keep the teleop terminal being focused, use the keys of "i, j, m, l" to control the robot moving forward, turning left, moving backward, and turning right.

   Steps 2-6 are demonstrated in this video (ssh is unnecessary):

https://youtu.be/9efBzYAi1QI

7. After driving the robot around, a map of the lab is expected to be generated and shown in rviz. Run the following command to save the map:

```
rosrun map_server map_saver -f /path/to/your_map
```

8. Now you have a map (that includes a yaml file and a pgm file) ready to be used for autonomous navigation.

   To do that, simply close all your ROS applications, then in a new terminal run the command in Step 2 to start the robot base driver, and run the following command for navigation:

```
roslaunch turtlebot_navigation amcl_demo.launch map_file:=/path/to/your_map.yaml
```

   To visualize the map and robot's pose, the command in Step 4 needs to be run as well. This video shows the steps of autonomous navigation (ssh is unnecessary):

   https://youtu.be/iGZPWjddans

9. The instructions above can be used for SLAM and autonomous navigation WITHOUT creating your own code.

   In order to let the robot visit the four corners in the lab in order, students will need to create code to send navigation goals to the robot (one goal at a time). More specifically, goal messages (in the ROS type of geometry_msgs/PoseStamped) need to be sent to the ROS topic of /move_base_simple/goal.

PART 2.

1. Speech recognition (from spoken language to text)

   Students will need a speech recognition tool to convert human voice to text. It is up to the student to select the tool, such as the ones that were discussed in students' proposal presentation. CMU Sphinx can be one of the good (but very old) options:

   https://cmusphinx.github.io/

   CMU Sphinx has a simple wrapper for ROS users, called pocketsphinx. The following page provides links to both the pre-built version and the source code hosted on github. Pay attention to the version in installation, which should be compatible with our computer and ROS versions.

   http://wiki.ros.org/pocketsphinx (outdated, not good for Kinetic anymore)
   https://github.com/shiqizhang6/ros_voice_control

   The above ros_voice_control provides the default vocabulary that includes only nine words:

   python ros_voice_control.py

   Please note that the above tool does simple "keyword" search, and cannot recognize complete sentences. In order to recognize natural speech with a language model, one needs to turn on the language model using "-lm /path/to/your/file/named/name.lm" From the link below (lmtool-new), you can generate your

own lm files.

It is likely that one needs to customize the dictionary file and/or language model, because the default ones include an unnecessarily large vocabulary. The following online tool allows users to upload example sentences and generate all necessary files for pocketsphinx.

http://www.speech.cs.cmu.edu/tools/lmtool-new.html

The generated language model include .dic and .vocab files.

Example commands include:

- "Move forward"
- "Turn left"
- "Stop" – It should be noted that the robot produces a lot of noise while moving, so you may want to use a wired microphone (the lab does not have it now) to allow the "stop" command.
- "Go to position one", where you can predefine a few positions in the lab to avoid repeatedly commanding the robot to "move forward"

Alternatively, students can use neural network-based speech recognition packages, such as:

https://github.com/mozilla/DeepSpeech

Another tool is the SpeechRecognition library, which provides more functionalities and can be used here.

https://pypi.org/project/SpeechRecognition/

2. Language understanding (parsing text)

Keyword search (shallow semantic parsing) can be the easiest way of understanding the text. However, there are other, more principled ways of parsing sentences. One example is the Cornell SPF on which the instructor has limited experience:

https://github.com/clic-lab/spf

Stanford CoreNLP, which provides a powerful neural parsing system.

https://corenlp.run

3. Dialog system

It is a good idea to implement a dialog agent that enables the robot to at least confirm your "expensive" command before it spends a considerable amount of time working on something. For instance, one can use the following "sound_play" package to speak out the current understanding of the user's command, where the user can confirm by simply saying "yes" or "no".

http://wiki.ros.org/sound_play

The instructor do not suggest students to develop complicated dialog managers (that suggests question-asking actions) in this assignment. However, that can be an interesting direction in final projects. The instructor is willing to discuss on related topics in person.

Finally, the robot will physically move its platform by generating local control signals (such as linear and angular velocities) and/or global control signals (such as 2D navigation goals), to accomplish navigation tasks.

WHAT TO TURN IN

Students will submit a link to the YouTube video (in the body of your submission email) that shows the robot autonomously navigates in the lab to visit the four corner areas, and then move out from the lab to the corridor, assuming the lab door is left open.

BONUS: if a student implements a detector for sensing whether the door is open or not, up to 10% bonus points would be given. Correspondingly, the video should demonstrate this by letting the robot visiting the corners, ***waiting in front of the lab door until it's opened by a person***, and then moving out from the lab through the door to the corridor.

Students also need to turn in a single file (in tar/zip/rar format), and name it as your last name (initial in uppercase) followed by the initial of your first name (uppercase). For example, the file name should be "ZhangS.tar" for the instructor.

Students will need to create a package (that includes two launch files) in such a way that

1. running the following command will make the robot ready for taking an initial position through rviz (students will decide what to be included in the launch files),

   roslaunch ZhangS assign3_bringup.launch

2. initial position of the robot can be assigned through "click and drag" on rviz
3. running the following command will start the robot's autonomous navigation behaviors.

   roslaunch ZhangS assign3_navigation.launch

4. running the following command will start the robot's voice-command behaviors.

   roslaunch ZhangS assign3_voice.launch

Upload your submission to Blackboard.

LIVE DEMONSTRATION

It is generally a challenge to evaluate dialog systems. Student will provide demonstrations of the "voice control" (part 2) to the instructor and other students in the lab. We will meet in the lab for demonstrations on the day of the submission deadline of this assignment.

The instructor will arbitrarily select a reasonably large area in the lab or in the corridor (say 2 meter by 2 meter) before the beginning of the demos. Each student will use voice command to drive the robot to that area, while avoiding possible obstacles, such as a rolling chair.