CS378 - Autonomous Intelligent Robotics
Spring 2013
Programming Assignment 1 (10 % of total grade)
Due Date: Thursday 1/31 12:30 PM

Notes

- It is important that you work on this assignment independently as it will serve as a building block for the rest of the course.
- It is OK to discuss the assignment with your friends and classmates, but please do run the simulator and robot on your own!
- The standard late policy as detailed in the syllabus will be followed.
- If you encounter an error or bug, try to solve it on your own first. Solving problems on your own is a good skill for research. Google your problem or error message see if anyone has had a similar problem and found or posted a solution. If you can't find the answer easily, then please e-mail the instructors or come to office hours.
- Ask the teaching staff if you have any problems you can't solve easily.
- Occasionally things will go really wrong in the simulator (for example, the navigation planner may crash if you give it an impossible goal). Generally, you can always exit everything and restart to resolve these problems.

Goal

The goal for this assignment is to let you get a first look at the existing codebase for the BWI project and familiarize yourself with it. There is NO programming required for this assignment. The main tasks of this assignment will be:

1. Download and install the BWI code
2. Compile the code
3. Run the simulator
4. Drive the robot with the keyboard
5. Drive the robot using navigation
6. Look at the simulated sensor output of the robot.

Instructions

1. **Find a computer with ROS-Groovy Installed (either yours or a public machine).**
   You can either use one of the computers in the public computer labs in ENS (either ENS 1 or ENS 31NR) or you can install ROS on your own machine if it is running Ubuntu Precise Pangolin 12.04.1 LTS.

   a. **Using a public CS machine.**
      Use one of the machines in ENS 1 or ENS 31 NR.

To get a list of the machines with ROS installed, type the following in the terminal of a public CS machine:

cshosts nethack64
cshosts kol64

Machines from either of these two lists have ROS-Groovy installed.

To use ROS, you will have to set some paths properly in your ~/.bashrc file. Add the following line to the bottom of your ~/.bashrc file using your favorite editor (emacs, gedit, vim):

source /opt/ros/groovy/setup.bash

b. **Using your own computer (running Ubuntu Pangolin Penguin 12.04.1 LTS).** Install ROS-Groovy on your laptop using these instructions: http://www.ros.org/wiki/groovy/Installation/Ubuntu

Test that your install of ROS works properly. Open a new terminal and type: *roscore*
If this command fails, your ROS installation is not setup properly.

2. **Download and Install the BWI code.**
Piyush has setup a python script to download the BWI code from our github repository and setup appropriate folders and workspaces for you. This script will create a ros directory inside your home directory and then create rosbuild_ws, catkin_ws, and gazebo directories inside that ros folder. We'll explain these directories later in the class. There are different directions for using the script on the department machines or your laptop, as for the laptop you'll have to download the script and a few extra ROS packages that the department machines already have.

Using a public CS machine
Here are the directions to install the code using the script:

a. Run the following command (**all one line**) to setup the configuration for the script:

/u/piyushk/bwi-update --config https://raw.github.com/bwi-spring-2013/bwi/master/bwi-update-config/user_config.yaml

This command is all one line. Here's a tiny one line version for copy and pasting:
/u/piyushk/bwi-update --config https://raw.github.com/bwi-spring-2013/bwi/master/bwi-update-config/user_config.yaml

b. Run the following command to download the BWI code from the appropriate github repositories (it may take a while):

/u/piyushk/bwi-update --sync

c. At the end of the script, it will print two lines that you need to add to the bottom of your ~/.bashrc file. Add these two lines to the bottom of ~/.bashrc.

d. Run the following command to execute your .bashrc file so that the new lines are run:

source ~/.bashrc

Make sure that any later commands that you run are in terminal windows opened after editing your .bashrc file, or that you have run source ~/.bashrc in that terminal window.

Using your own laptop

Here are the directions to download the script and use it to install the code:

a. Run the following command to add Piyush's repository to your sources file so that you can then download packages from his repository:

```
sudo apt-add-repository ppa:piyushk/bwi
```

b. Download and install the update script from the repository by running the following command:

```
sudo apt-get update && sudo apt-get install bwi-update
```

c. Run the following command (**all one line**) to setup the system configuration for the script:

```
sudo bwi-update --system --config
https://raw.github.com/bwi-spring-2013/bwi/master/bwi-update-config/
system_config.yaml
```

This command is all one line. Here's a tiny one line version for copy and pasting:

```
sudo bwi-update --system --config https://raw.github.com/bwi-spring-2013/bwi/master/bwi-update-config/system_config.yaml
```

d. Run the following command to install more ROS packages required for this project:

```
sudo bwi-update --system --sync
```

e. Run the following command (**all one line**) to setup the user configuration for the script:

```
bwi-update --config https://raw.github.com/bwi-spring-2013/bwi/master/
bwi-update-config/user_config.yaml
```

This command is all one line. Here's a tiny one line version for copy and pasting:

```
bwi-update --config https://raw.github.com/bwi-spring-2013/bwi/master/bwi-update-config/user_config.yaml
```

e. Run this command to download and install the BWI code from github (it may take a while):

```
bwi-update --sync
```

f. At the end of the script, it will print two lines that you need to add to the bottom of your ~/.bashrc file. Add these two lines to the bottom of ~/.bashrc.

g. Run the following command to execute your .bashrc file so that the new lines are run:

```
source ~/.bashrc
```

Make sure that any later commands that you run are in terminal windows opened after editing your .bashrc file, or that you have run source ~/.bashrc in that terminal window.

3. Compile the BWI code

ROS provides a tool called rosmake to compile packages. It will automatically find their dependencies and compile them as well. For this project, we need to compile 2 different packages relating to the segbot and the gazebo simulator.

Note that for all of these ros commands, ros will find the paths for the packages automatically from the information you added to your ~/.bashrc file. So all of these commands can be run from any directory.

Run the command:

rosmake segbot_apps segbot_gazebo

This command will compile these two packages along with their dependencies (it may take a while). If this command completes without error, then everything should be ready to run.

4. **Run the simulator and simulated robot (with the Hokoyu laser scanner)**
   Now, we're going to start the simulator and the robot. The simulator is available from ROS and is called **Gazebo**. The simulated robot model is something we built ourselves. Starting the simulator and the robot will involve a few steps. Again, all of these commands can be run from any directory as ROS will figure out the right path for each.
   a. Run the command:

      roscore

      This command starts the ros master, which sets up communication between different ros nodes.
   b. Next, we'll start the simulator with the simulated world, but no robot yet. Open a new terminal. (Tip: hit Ctrl+Shift+T to open a terminal in a new tab in your current terminal window.) In the new terminal, start the simulator by running:

      roslaunch segbot_gazebo bwi_test_world.launch

      This command launches multiple ros nodes related to running the simulator, all of which are listed in the test_world.launch launch file.
   c. Now we'll start the simulated robot, this version of the robot has the Hokoyu laser scanner we talked about in class. In another terminal, start the simulated robot by running:

      roslaunch segbot_gazebo segbot_mobile_base.launch

      This command launches multiple ros nodes related to running the simulated robot, all of which are listed in the segbot_mobile_base.launch launch file. This command will launch a simulated robot with a Hokoyu laser scanner. Later, we'll look at a simulated robot that has a Kinect sensor instead of a laser scanner.

   You should now see a window with the simulated world and simulated robot. You can use your mouse to pan and zoom around inside the simulator. Hold down the left mouse button and move the mouse around to move your view of the world. Hold the center button and move your mouse to tilt and pan your camera. Finally, scrolling with your mouse should zoom in and out (slowly). Note: if moving and panning are incredibly slow, your machine may be too slow to run the simulator.

5. **Control the robot with the keyboard**
   Next, we're going to control the robot with the keyboard. Open another terminal and run the following command to start the keyboard tele-operation script:

   rosrun teleop_twist_keyboard teleop_twist_keyboard.py

You can now press the designated keys within this terminal to drive the robot. **You must press the keys in the terminal window with the terminal window active.** You may want to resize the terminal window to make it small and then put it on top of the simulator window so you can see the robot moving in the simulated world as you command it. Try out driving the robot into the table and through the doorway in the simulated world.
Your robot may get stuck on walls, furniture, or other objects. If the robot gets stuck on the wall, use the move tool on the top of the Gazebo simulator window to move the robot.

Take a screenshot of you driving the robot somewhere by pressing the print screen button on your keyboard. You will turn in this screenshot later as proof that you completed the assignment.

6. **Control the robot using navigation goals**
   Now we're going to control the robot using the existing navigation controls. We will run **rviz**, which is a robot visualization tool available through ROS. This tool allows us to see a visualization of the robot, its map, where it thinks it is, and what sensor data it is receiving. This is the same tool we will use on the real robot to look at sensor data nad provide navigation goals. Starting the robot's navigation software will involve a few steps.
   a. Kill all the previously running code by hitting Ctrl+C in each terminal. Make sure to wait until all the processes have exited (some may take a while to exit).
   b. Restart the simulator and robot by running the following commands (both in their own terminal):
      roslaunch segbot_gazebo bwi_test_world.launch
      roslaunch segbot_gazebo segbot_mobile_base.launch
   c. Launch the robot's map server by typing the following command in a new terminal:
      roslaunch segbot_gazebo bwi_test_world_map_server.launch
   d. Launch the robot's particle filter localization by typing the following command in a new terminal:
      roslaunch segbot_navigation amcl.launch
   e. Launch the robot's navigation by typing the following command in a new terminal:
      roslaunch segbot_navigation move_base_eband.launch
   f. Launch the visualization and navigation controller (rviz) by typing the following command in a new terminal:
      roslaunch segbot_navigation rviz.launch config:=nav_eband
   g. Now you can give the robot goal locations to navigate to on its own. To autonomously navigate the robot, click the 2D Nav Goal button on the top of the rviz window (**not** the 2D pose estimate button). Then **click, hold down your mouse button, and drag** in the map to select a target location and orientation. You must click **and drag** in order to select both a desired location **and orientation**. If you just click without dragging, it will not work. You will now see

the robot driving towards your specified navigation goal. Play around with various goals, and examine how the robot performs when going through doorways and past other objects.

h. Your robot may get stuck on walls, furniture, or other objects. If the robot gets stuck on the wall, use the move tool on the top of the Gazebo simulator window to move the robot. If you move the robot too much, the robot may get lost.
i. In the original gazebo simulator window, you can also try adding objects into the robot's path by clicking the cylinder, sphere, or cube on the top and then double-clicking a location on the map to add them. Check out how the robot adjusts when navigating past or around unexpected obstacles.
j. You can also change where the robot thinks it is in the rviz window. If you click the 2D pose estimate button and then select a location and orientation by clicking and dragging, the robot will now think it is at that location. Notice that if you do this to the robot, it will have a very difficult time recovering and navigating anywhere. This sort of situation would be when the robot might want to ask a friendly human for help.
k. This version of the simulated robot is using the Hokoyu laser scanner we discussed in class. The laser scanner sends out laser beams at a variety of angles and finds the distance to the nearest object at each angle. Take a look at the simulated sensor output in the rviz window. You'll see red dots where it estimates it is getting returns from its laser scanner. Some of these red dots will be on walls, table legs, or other objects, and some will return the maximum distance the laser scanner can sense if there are no objects in that direction.
l. Take a screenshot of the robot navigating somewhere by pressing the print screen button on your keyboard. You will turn in this screenshot later as proof that you completed the assignment.

7. **Look at the output of a robot running with the Kinect sensor**
Now, instead of a robot with a Hokoyu laser scanner, we'll simulate a robot with an xbox Kinect sensor, which provides RGB-D information. The Kinect provides a 640x480 image with both color and depth for each pixel. There are a few steps to start the simulated robot with the Kinect sensor:
   a. Close all the windows you have running. To do so, hit Ctrl+C in each terminal window. Make sure all the processes have stopped running (some may take a few seconds to exit).
   b. Restart the simulator with the following command:
      roslaunch segbot_gazebo bwi_test_world.launch
   c. Run the simulated robot, this time with an extra option to use the Kinect sensor. To do so, run the following command (**all one line**) in a terminal:
      roslaunch segbot_gazebo segbot_mobile_base.launch configuration_file:=`rospack find segbot_bringup`/launch/includes/ segbot_kinect.auxillary.launch
      Note that this command is **all one line** and that the ` mark is a **backtick (top left of your keyboard)**. Here's a tiny one line version to copy and paste:

roslaunch segbot_gazebo segbot_mobile_base.launch configuration_file:=`rospack find segbot_bringup`/launch/includes/segbot_kinect.auxillary.launch

d. Start the navigation stack again. To do so, run the following commands each in their own terminal windows:

```
roslaunch segbot_gazebo bwi_test_world_map_server.launch
roslaunch segbot_navigation amcl.launch
roslaunch segbot_navigation move_base_eband.launch
```

e. Start rviz to look at the robot output. To do so, run the following command in a terminal:

```
roslaunch segbot_navigation rviz.launch config:=nav_eband
```

f. Next, we will turn on visualization of the Kinect point cloud. This will show the pixels captured by the Kinect camera, with their correct color at the sensed depth. In the rviz window, click the “Add” button on the bottom left. Select PointCloud2 and click OK. Scroll to the bottom of the left window and expand PointCloud2. Click the white box to the right of Topic under PointCloud2. Select “/nav_kinect/depth/points” as the topic, which should be your only option. Click to the right of Style, where it says Squares, and select Points instead. Now you should see a display of the Kinect data in the rviz window. If the robot is still in its starting location, you’ll see it sensing the wall in front of it, which is not very exciting.

g. Provide some navigation goals and look at the Kinect output as the robot drives around. Try panning and zooming in the rviz window to get a better look at the simulated Kinect data. See if there are any issues with navigating with the Kinect sensor only.

m. Take a screenshot of the robot’s Kinect visualization by pressing the print screen button on your keyboard. You will turn in this screenshot later as proof that you completed the assignment.

8. Submit the assignment

Each time you pressed the print screen button, the computer should have created an image file called “Screenshot from DATE TIME.png” in your ~/Pictures directory. If it is not in that directory, you may have to search for it. You will turn in your three screenshots as proof that you completed the assignment. You will turn in these screenshots using the turnin tool on the CS machines.

If you are using your own computer,

you will first have to copy the screenshots to the CS machines as the turnin tool only exists on the CS machines. To copy a file from your computer to a CS machine, you would type the following:

```
scp filename username@machinename.cs.utexas.edu:~
```

This command will copy the file “filename” to your home folder. You must also replace **username** with your CS username and **machinename** with your favorite CS machine name (such as ki-rin). Use this command to copy all three images to the CS machine.

To run the turnin commands on the CS machine, you can now ssh into a CS machine with the following command:

```
ssh username@machinename.cs.utexas.edu
```

Again, you must also replace **username** with your CS username and **machinename** with your favorite CS machine name (such as ki-rin).

On a CS machine (either directly or through ssh),

You will turnin the three captured screen images to the TA by using the turnin command with grader *shweta* and assignment name *hw1*. For full directions for using the turnin tool, type:

```
man turnin
```

For this assignment, you probably want something like the following:

```
turnin --submit shweta hw1 screen1.png screen2.png screen3.png
```

This command would turn in the three image files for assignment *hw1* to grader *shweta*.