CS378 - Autonomous Intelligent Robotics
Spring 2013
Programming Assignment 2 (10 % of total grade)
Due Date: Thursday 2/21 12:30 PM

Notes

- This project should be done in groups of 2-3 students.
- It is OK to discuss the assignment with other groups, but please write your own code
- 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.

Goal

The goal for this assignment is to let you get to play around with the robot and do some simple programming on the robot. The main goal of the assignment is make the robot follow a ball detected by the Kinect camera. The tasks of this assignment will be:

1. Download and install the BWI code
2. Write a node that receives detected image blobs and publishes command velocities for the robot to follow the ball.
3. Start up and run the robot.
4. Run your node on the lab machine to control the robot.

Overview

You are going to write code to drive the robot so that it follows an orange ball. We are running a vision node (called cm vision) that outputs all the detected regions of all orange found in an image. Based on the detected regions (called blobs), you will decide how to move the robot. For example, if the detected orange region is on the left of the image, you want to turn left, if its on the right, you should turn right. If the blob is really big, then you are really close to the ball and should back up, and if its really small, then you are far from it and you should go forward. In addition to the ball, there may be other regions of orange in the image. You will be given an array of all the orange blobs in an image and you may have to decide which one to follow.

General Robot and Assignment Tips:

- You need to run each command in a new terminal (not one where things are already running).

- Ctrl+Alt+T will open a new terminal or terminal tab.
- You can press the up key to go to previous commands run in that terminal.
- The rostopic echo /blobs line will print some errors and then not start printing blobs for about 10 seconds, be patient.
- You may want to drive the robot around. You can use the same keyboard control script you used in the first assignment in the simulator by running:
  rosrun teleop_twist_keyboard teleop_twist_keyboard.py
- If your robot does not move, you may need to restart the base (press the yellow button, then the green to turn it off, then green and then yellow to turn it back on).
- If things do not work properly, the first thing to try is closing all your nodes with ctrl+c and restarting them. Make sure you wait for them to fully exit.
- The color segment / blob detection is noisy. You may see multiple blobs in a ball, or you may get small blobs for other tiny bits of orange in the camera image.

Instructions

1. **Reserve time on a robot.**
   Since there a lot of students and only four (three for now) robots, we will require you to sign up for time on the robots ahead of time. Please do not reserve more than 2 hours at a time (once you've used your hours, you can reserve another 2 hours).
   Reserve a robot by filling in a block here:
   https://docs.google.com/spreadsheet/ccc?key=0Aq02p1zESnk2dFhsWFFqalFvLWtFQ1pSY252Z3lWX2c#gid=0

2. **Use one of the lab machines in the BWI Lab (GDC 3.414B).**
   We have each robot configured to communicate with and run with a specific machine in the BWI lab. On top of the laptop for each robot, there is a label telling you which desktop machine it connects to. You should use the machine for the robot you reserved. You can use your laptop for writing and testing code, but eventually you'll need to run your code on one of the lab machines.
   You will have a user account set up on these machines. They have shared folders and user info, similar to the CS lab machines, so you can login to any of the machines to access your files.

3. **Download and Install the BWI code.**
   Since we are using different computers, and we have added some code for this assignment, you will have to download and install the code again.
   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.

For all of these directions, we'll be using a terminal window. Either find it from the launch menu, or hit Ctrl+Alt+T to start a terminal window.

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:

   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

   Warning: Sometimes copying and pasting from the pdf (depending on what pdf viewer you use) creates weird symbols or missing hyphens - you may have to type in these lines instead.

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

   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. You can open a text editor for this at the command line, for example, either emacs or gedit:

   gedit ~/.bashrc
   emacs ~/.bashrc

d. You will also need to add two lines to the .bashrc file so that ROS knows the IP of your machine when setting communication with the robot. At the bottom of the ~/.bashrc file, add the following two lines:

   ASG1_LOCATION=`rospack find asg1`
   export ROS_IP=`${ASG1_LOCATION}/scripts/get_addr.py eth0`

   Note that the ` marks are backtics (top left of your keyboard). Also note that these two lines must be AFTER the other two lines you added.

e. 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.

4. 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 vision processing code.

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_bringup asg1

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. The second package, asg1, is the package you will be modifying. Right now it will simply printout some messages when orange blobs are seen, rather than drive the robot towards those blobs.

Next, we'll test things on the robot so you can see what its camera images look like, and what the blob output looks like.

5. **Unplug the robot from the wall.**

Find the robot for the computer you are using. Unplug the robot by following these steps:

1. Every time you start working with the robot, you should disconnect it from charging at the wall. Locate the correct robot.
2. Find the battery charger (sitting on the white shelf) that this robot is connected to. Unplug the battery charger from the wall or power strip.
3. There is a black and red cable coming from the big battery on the robot with a white connector, than then goes to a cable to the battery charger. Disconnect the white connector by pulling it apart.
4. The power inverter (black and blue box sitting on the bottom base of the robot) has another cable with a white connector. Connect this cable to cable going to the large battery on the robot (the one you just disconnected). The connector very occasionally produces a spark. This is fine.
5. Before leaving, ensure that you connect the robot back to the chargers. Follow the instructions at the bottom of the page.

6. **Power on the robot.**

Now we will turn on the robot and its laptop. To turn the power inverter (black and blue box) on, flip the switch on the power inverter An led will turn on indicating that the inverter is on and powering the Kinect and the Segway.

1. Now, turn the segway laptop on. Open up the laptop lid, and press the power button. Use the username/password fri/fri2013 to log onto the machine.
2. Now turn the Segway base on. Locate the power buttons underneath the bottom of the Segway base. First press the green to power the base, you'll feel it sink and stay sunken. Then press the yellow button to turn the segway processors on; you'll hear an audible beep. A picture of the segway buttons for reference is available here
3. The robot will already have all the code downloaded, installed, and compiled.

4. Note that **you should turn the laptop on first.** Sometimes the segway base does not boot properly if the laptop is not already on. If this happens (the robot will not move), you can always turn the segway base off and back on.

7. **Start the robot code to test things out.**

Now we'll start running the robot code so you can look at the camera images and blob output. First, we'll start a ROS master node on your desktop machine. On the lab desktop machine, type:

roscore

This command will start the ros master.

Note that all of these commands need to be run in new terminals or terminal windows (not where something is already running). You can start a new terminal using Ctrl+Alt+T.

Throughout all of these steps, if something does not work properly, he first thing to try is closing all your nodes with ctrl+c and restarting them. Make sure you wait for them to fully exit.

Start the robot code:

Next, we'll start up the appropriate nodes on the robot. We have a launch file full of the correct nodes for this assignment. On the robot's laptop, type:

roslaunch segbot_bringup segbot_asst1.launch

You should see a bunch of text go by as the robot launches various nodes.

Look at ROS topics:

Now, let's take a look some of the topics being published by the robot. You can type:

rostopic list

on either your desktop machine or the laptop to see a list of the available topics from the robot. You will see a lot of topics relating to the kinect sensor, publishing various information from the color and depth sensors.

Launch the blob detector:

Now we'll start the cmvision node that will take in images and publish blob messages. These blob messages provide info about where in the camera image there are regions of one color. These areas are called blobs. In particular, you are interested in orange blobs, which is all this node is setup to detect right now.

Essentially, this node looks at an image, and forms a rectangle around any regions of orange. It then fills in an array with info about all the regions detected in this image, and publishes this array on the topic /blobs. For each blob, it tells you the x,y coordinates of the center of the blob in the image, the area of the rectangle for the blob, and the x,y coordinates for the corners of the blob. Based on the location and size of the blob in the

image, you can decide how to move the robot.

To start this node, on the robot, type:

roslaunch asg1 cmvision_blob_detector.launch

This launches the cmvision node and passes some parameters and arguments. It tells the cmvision node what color table to use as well as what images it should process (which topic to listen to). You can take a look at the launch file if you like. On either the desktop or the robot, to go to the directory where the assignment code is, type:

roscd asg1

This command will cd (change directories) to the folder containing this package. Now cd into the launch directory by typing:

cd launch

Now you can open the launch file in your favorite editor (emacs, gedit, vim, nano). You'll see where the image topic is re-mapped to /kinect/rgb/image_color, and you'll also see where the color map file path is set.

You can also take a look at the color map file, it's one directory up, type:

cd ..

to go up a directory. And the color map is the colors.txt file. It simply gives a range of values for Red, Green, and Blue that should be classified as orange.

There are two orange balls in the lab: a soccer ball and a small plastic ball. You can use either of these for your assignment. Please hold the ball in the air in front of your robot, as on the ground the robot will have to be very far away to see the ball.

Look at the camera image:

Alright, back to looking at the output of the robot. You can view the robot's images by typing on the robot:

rosrun image_view image_view image:=/kinect/rgb/image_color

Now you can see what the robot's image looks like and what it is seeing.

Look at the blob info:

You can also take a look at the robot's blob messages being published by the cmvision node. On the desktop or the robot, type:

rostopic echo /blobs

This command will first produce a few error messages and then there will be about 10 seconds before it starts printing blob info. Be patient.

This command will print out each message on topic /blobs that is sent by the system. You will see lots of messages go by (particularly if there's an orange object in front of the camera). Again, these blob messages tell you about orange blobs (sections of all orange) found in the camera image. The /blobs message has a few important pieces of information. It gives you a blob_count of how many blobs it found in the image, and then for each gives an area, and coordinates of the center and corners of the blob found in the image. You will notice that it sometimes detects small bits of orange without there

being a ball (something else it is seeing looks orange). Also, the ball may sometimes be broken into multiple blobs.

Drive the robot:

You may want to drive the robot around. It's bad for the robot to move it while it is powered on. You can use the same keyboard control script you used in the first assignment in the simulator by running:

rosrun teleop_twist_keyboard teleop_twist_keyboard.py

You can run this node on the robot or on your desktop machine (but not both at once). I would recommend running it on the desktop, as it is much easier to control when your keyboard is not moving.

If your robot does not move, you may need to restart the base (press the yellow button, then the green to turn it off, then green and then yellow to turn it back on).

Start the dummy control node that you will be filling in:

Finally, you can start the dummy node that you will be filling in. We're going to run this control node on the desktop machine, and it will communicate through ROS to the robot. This gives us some advantages - you don't have to copy and compile your code on the robot, and you can hit ctrl+c easily on your desktop machine if things go wrong. On the desktop machine, in another terminal window, type the following:

rosrun asg1 asg1

For now, this node will always publish command velocities of 0, and will print out some information about the blob messages being sent to it.

Now that you have some idea how the code works, what the robot sees, and what the blob output looks like, you should ctrl+c all these processes and get to work on your code.

8. **Write your own code to give commands based on processed images.**

We have already setup the framework of a ROS node for the code you need to write. You most likely want to do the code writing and testing on the desktop machine, as this is also where you'll be running this node. We're going to run this control node on the desktop machine, and it will communicate through ROS to the robot. This gives us some advantages - you don't have to copy and compile your code on the robot, and you can hit ctrl+c easily on your desktop machine if things go wrong.

It's in the asg1 package in the class-code repository. You can access this node by typing:

roscd asg1

This will cd (change directories) to the folder containing this package. You may want to take a look at some of the files in this directory. The code is in the src directory. Launch files are in the launch directory. The color table is in the file called colors.txt.

CMakeLists.txt lists the files to be compiled when using rosmake. manifest.xml lists which other ros packages this code depends on.

Now cd into the directory with the source code:

cd src

In here you will see one file, called asg1.cpp
Open and edit this file in your favorite text editor, for example:

emacs asg1.cpp

gedit asg1.cpp

First, you may want to check out the file as is. You will notice at the bottom of the file, the node is subscribing to topic blobs, and passing in a callback method to be called whenever a blob message is received, called blobCallback. You'll also see that the node is publishing to topic cmd_vel.

Above this code, you'll see the blobCallback method. It takes in a pointer to the blob message published by cmvision. This message contains a count of all the blobs found in the latest image, as well as an array with info about each blob found in the latest image. Inside this blobCallback method, you'll see we've given examples of some of the variables you may want to access in deciding what velocities to send. Currently the code checks if there were any detected blobs, and loops through them, printing the area of each. Then if any blobs were detected, we publish a desired velocity of 0. This is the code you need to change!

A few things to consider:

a. To start with, you may want to just work on turning, try to turn the robot so that the x and y of the blob are in the center of the camera image. Turning should be safer than moving forward and backward.
b. Then, you may want to worry about the ball's area and move the robot forward and backward to get to a desired distance from the ball.
c. Finally, you may want to consider robustness to noise, other orange objects and the like. One option for this is to only consider detected orange blobs of some minimum size (smaller ones may simply be noise or random orange pixels).
d. You will also probably want to decide what to do when multiple orange blobs are detected (ignore them all? pick one to track?)
e. Another thing to think about is how to make the robot move smoothly rather than moving jerkily or randomly.
f. When you don't detect a ball, you probably want to send the robot a velocity of 0 to make it stop moving. Then it should only move if you place a ball in front of its camera.

After editing the code, you can then re-compile the asg1 package by typing:

rosmake asg1

Make sure it compiles with no errors. Now you can go back through step 6 to start

roscore on your desktop machine, launch segbot_bringup and the cmvision node on the robot, and then start your asg1 node on your machine. If the robot starts moving crazily or you want to stop it immediately, hitting Ctrl+C on your node should stop it from sending commands to the robot, so that the robot should stop.

Have fun!

9. Finally, shut down the robot and plug it back into the charger.

It's important that the robot is off and charging before you leave. To accomplish this:

1. You need to do this before you leave the lab.
2. Power off the Segway base. Press the yellow button and then the green button.
3. Turn off the laptop (standard laptop shutdown).
4. Turn off the power inverter (the black and blue box on the robot base).
5. Disconnect the power inverter from the battery (the white connector connecting these cables together).
6. Connect the battery to the battery charger while the battery charger is not plugged into the wall, by connecting the two white connectors together.
7. Plug in the battery power charger into the wall. Select battery size as large (6A)

10. Submit the assignment

Now you will turn in your edited asg1.cpp file. Please make sure that you list all your group members names and UT EID's at the top of the file, as only one of you should turnin the file. You will turn in the file using the turnin tool on the CS machines.

Since we're using the BWI lab machines, you will first have to copy the file 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 the file 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).

You will turnin the asg1.cpp file to the TA by using the turnin command with grader *shweta* and assignment name *hw2*. For full directions for using the turnin tool, type:

man turnin

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

turnin --submit shweta hw2 asg1.cpp

This command would turn in the asg1.cpp file for assignment *hw2* to grader *shweta*.