CSCE 439/839: Robotics
Lab 3: Ball Detection and Collection

Instructor: Carrick Detweiler
carrick _at_ cse.unl.edu
University of Nebraska-Lincoln
Fall 2015

Started: Wednesday, October 21, 2015
Lab Checkpoint (Up through Sec. 4): Friday, October 30, 2015
Lab Due Date: Friday, November 20, 2015

Note: You have a lot of time for this lab, but plan accordingly with your group since there is a **Homework Due** during this time and you will need to do a **Final Project Proposal** as well.

1 Overview

In this lab you will locate a ball and then push the ball to a location defined by a visual landmark. This lab will require that you are able to visually localize a ball, drive to it, then bring it to a location marked by a visual landmark. I am giving you the ball and landmark detection code, but you will have to adapt it, characterize it, and integrate it into your system.

Make sure to read through the entire lab before starting to implement these. You are expected to complete up to and including Section 4 for the checkpoint (mainly demonstrating both visual systems working); however, I would encourage you to have completed more than this by the time of the checkpoint.

Remember and review the safety instructions from Lab 1.

2 Odroid Setup (10pts.)

You now have an ODROID-XU4 `http://odroid.com/dokuwiki/doku.php?id=en:odroid-xu4`, which is a small, yet powerful, processing board that can run GUN/Linux and ROS. We have already configured it with ROS and Ubuntu. You can use the monitors and keyboards in the lab to connect to it and do initial configuration (e.g. connecting to the lab wifi, figuring out the IP address, setting a new password, etc), but once you have it configured you will not need to use a monitor. Instead, you should use `ssh` to connect to it from your computer (and commands like `scp` to copy code) and launch ros nodes from the command line. Your first task is to move all of your code over to the Odroid and test your Lab 2 code.

Question: Describe the setup process and any challenges you encountered.

ROS nodes running on different computers can communicate with each other. In this section you should configure your Odroid and computer to communicate with each other. Read the tutorial on this at `http://wiki.ros.org/ROS/Tutorials/MultipleMachines` for details. You should also read the networking troubleshooting guide at `http://wiki.ros.org/ROS/NetworkSetup` as you will need to do some network setup to make this work right. To summarize the first step (2.3 in the NetworkSetup) is to add an entry in the `/etc/hosts` on your machine and the Odroid to allow them to resolve the IP addresses against the hostname of the machines. You can then start the roscore on the Odroid[1] and tell your computer where that core resides with the command `export ROS_MASTER_URI=http://odroid:11311`, where `odriod` is the hostname of your Odriod.

Once this is configured you can launch some rosnodes on the Odroid and others on your computer and they will be able to communicate with each other. This is advantageous as you can run the nodes that you are not changing on the Odroid and the ones that you are changing on your own laptop (to allow easier

[1] You could also configure it so that the roscore runs on your laptop, it is up to you.

editing and faster compiling). This is also good since you can run the vision processing code on the Odroid to prevent sending all of the images across the network, which can cause significant slowdowns.

Question: Describe the setup process for getting ROS to run on both your Odroid and your laptop. Write two nodes that communicate with each other to determine the network latency. To do this you can have node A send a message with a header time and then have node B just echo that message back to node A. Record the length of time that it takes for the message to do the round trip. Try changing the amount of data transmitted in the message (e.g. add an array or image that you transmit) to see how this impacts the latency. Compare this when running the nodes on a single machine versus across the network.

Some important reminders regarding your Odroid:

- When you are switching batteries, you can plug the Odroid into the wall power in addition to providing power from the battery. Do not leave both connected for any significant amount of time.
- Make sure to issue the `sudo shutdown` command before disconnecting power from the Odroid. If you do not, you may corrupt the file system or you may not be able to log in properly.
- You should set the password on your Odroid account to something your group knows and not just the default password as this could cause confusion among groups.
- You should primarily be connected to the lab wifi, which is not connected to the internet. You can connect to unl-air to update software, but you will not be able to easily set the ROS master node to another computer while on unl-air.
- If you have trouble getting your computer running ROS to talk with the Odroid be sure to double check the IP addresses and hostnames in `/etc/hosts` on both computers.

3 Visual Landmarks (20pts.)

On the course website there is link to download ROS modules needed to identify barcode-like visual landmarks (called `landmarkSelfSim`). Download this code and place it in your ros directory. You may need to install a couple of packages if they are not already installed[2]:

```
sudo apt-get install ros-indigo-usb-cam ros-indigo-cv-bridge ros-indigo-image-view python-wxgtk2.8
```

It should compile with some warnings, but no errors. I have also provided a number of these landmarks for use in class. These are “self-similar landmarks”[3], which means they are easily and quickly identified by only looking along one scan line in an image, no matter how far away they are. In addition, they have a binary bar code on the side that uniquely identifies each landmark.

There are two different launch files for the vision code `landmarkSelfSim.launch` and `displayDebugImages.launch`. The first launches the landmark detection system (along with the camera drivers) and the second is used to display the output images, which is useful for debugging.

Question: What message does the landmark detection code publish about the location of landmarks? What are the various fields and what do they mean?

Question: At what framerate do you detect landmarks? Does it matter how many landmarks are in the image? Test this on both your Odroid and another computer report the results. Make sure to describe the systems you tested this on (e.g. virtual machine, native, etc).

Question: What is the maximum range you reliably can detect that a landmark is in the image?

Question: What is the maximum distance that you can accurately identify the id of the landmarks?

Question: For the previous question, what impact does the angle of the landmark have on the identification?

[2]You can then test that the camera works by running roslaunch usb_cam usb_cam-test.launch

[3]D. Scharstein and A. Briggs. Real-time recognition of self-similar landmarks. Image and Vision Computing, 19(11):763-772, September 2001.

Question: How well does the landmark detection code work if the landmark is partially covered or out of the frame?

Question: Do experiments to characterize and calibrate the height of the landmark to the distance away it is in the image. Derive an equation that fits your data so that you can get a distance estimate to the landmark when you detect it. Characterize the performance of the range estimation.

***Question:* 839 majority groups only:** Adapt the landmark detection code so that it can detect landmarks that are upside down. Describe how you did this (hint: you should only need to modify `landmarkDetector.cpp`. You may also want to publish an additional debug image showing the results). Make sure that you publish the locations of any upside down landmarks and that you can use the same visual servoing code on upside down landmarks as regular ones.

4 Ball Detection (15 pts.)

In the code download for this lab there is code to detect balls (called `ballDetector`). In the `launch` directory, there are two launch files (similar to the landmark detection code). The first, `ballDetector.launch`, launches the camera and ball detector code. The second, `displayDebugImages.launch`, will display three different images you can use for debugging. If you find that the ball detector slows down your system, you can disable the debug images and information by commenting out the `#define BALLDETECTOR_DEBUG` line in the `ballDetector.cpp` file.

The ball detector works by first converting the image to HSV color space. It then performs thresholding to filter out all pixels that are between a low and high HSV threshold. These values can be configured in the launch file (see `ballDetector.launch` for an example) and can be dynamically changed by running the command `rosrun ball_detector configGUI.py`. It then searches for the largest group of connected pixels that have similar height and width (as a crude approximation to find circular groups of pixels). The largest group that meets this criteria is selected as the "ball" in the image and a message is sent out with this ball location.

The debug images display the HSV image, thresholded image (white pixels indicate those that are between the low and high thresholds), and a marked up image with white pixels indicating the boundaries of connected groups. In addition, in this image the final ball location is marked with a circle. Use these images and the configGUI to pick good HSV low and high thresholds.

Question: How did you go about picking the HSV thresholds for various balls? Do this for a couple of different color balls and report on the results. How does lighting affect the values?

Question: How many locations per second does the ball detector detect? Does it change based on different HSV thresholds? Make sure to describe the systems you tested this on (e.g. virtual machine, native, etc).

Question: Do experiments to characterize and calibrate range to the ball based on the reported radius. Derive an equation that fits your data so that you can get a distance estimate to the ball when you detect it. Characterize the performance of the range estimation.

***Question:* 839 majority groups only:** Write a new node that subscribes to the HSV debug image and automatically tunes the HSV values when a ball is placed on a white background (it is ok to assume that the ball is at least a certain size in the image). Describe the details of this node and how well it works compared to manual tuning.

5 Visual Servo (15pts.)

In the previous lab, you implemented a reactive controller that maintained a fixed distance from an object based on the IR sensor readings. In this lab, you will implement a similar behavior by visually servoing to center and maintain a fixed distance from a landmark. Using the landmark detection code implement a visual servo behavior. The visual servoing code should be able to follow a landmark with a particular ID at a particular distance (you should be able to easily change these parameters).

Question: Describe your visual servo implementation.

Question: How well can you perform visual servoing? If you start from far away, how quickly can you get to the target distance? Include useful plots to help characterize the performance (e.g. estimated distance over time). The characterization of the servoing behavior should be detailed.

Question: What do you do if you lose sight of the target?

6 Vision Arbitrator (5pts.)

Create a new launch file that will launch both the ball detector and landmark detector code.

Question: With both running, what detection rates do you achieve? Do they differ from when you only had one running?

Most of the time you will not need to detect landmarks at the same time as you will detect balls. In this section you should create a system that will allow you to switch between using either landmark, ball detection, or both. You should do this by creating a vision arbitrator that will only pass images on to these nodes when needed.

Question: Describe the structure and how you implemented your vision arbitrator.

7 Ball Collection (20pts.)

N.B. As is frequently the case, this section is the shortest length, but will likely take the most time. So plan accordingly!

Your goal in this section is to locate a ball, "collect it," and then bring it to a specified landmark number. The way you do this is up to you. I would recommend keeping it simple by perhaps adding a simple fork to the front of your boat to easily collect the ball and push it along. You could also make use of the range finders to tell if the ball is in the bin so that you don't need to aim the camera down too low. Note that there may be multiple landmarks and I will tell you which one you need to bring it to. You get to choose the color of the ball you want to use, although I will place it.

Question: Describe your approach and provide a detailed characterization of the performance. This is the only question for this section, but I expect a detailed explanation in this section.

8 To Hand In

You should designate one person from your group as the point person for this lab (each person needs to do this at least once over the semester). This person is responsible for organizing and handing in the report, but everyone must contribute to writing the text. You should list all group members and indicate who was the point person on this lab. Your lab should be submitted by handin, `http://cse.unl.edu/~cse439/handin/`, before the start of class on the due date. Include a pdf of your report and your source code in handin. Also make sure to bring in a **printed copy** to class on the due date.

Your lab report should have an introduction and conclusion and address the various questions (highlighted as ***Question:***) throughout the lab in detail. It should be well written and have a logical flow. Including pictures, charts, and graphs may be useful in explaining the results. There is no set page limit, but you should make sure to answer questions in detail and explain how you arrived at your decisions. You are also welcome to add additional insights and material to the lab beyond answering the required questions. The clarity, organization, grammar, and completeness of the report is worth **10 points** of your lab report grade.

In addition to your lab report, you will demonstrate your system and what you accomplished up to this point to the instructor at the beginning of lab on the due date. This is worth **15 points** of your overall lab grade. You do not need to prepare a formal presentation, however, you should plan to discuss and demonstrate what you learned and accomplished in all sections of the lab. This presentation should take around 10 minutes.

Question: Please include your code with the lab report. Note that you will receive deductions if your code is not reasonably well commented. You should comment the code as you write it, do not leave writing comments until the end.

Question: Include an `rqt_plot` of your final system and comment on your overall system architecture.

Question: For everyone in your group how many hours did each person spend on this part and the lab in total? Did you divide the work, if so how? Work on everything together? Give details on the hours each group member worked (e.g. keep a list of how many hours per day each person worked and on what).

Question: Please discuss and highlight any areas of this lab that you found unclear or difficult.