CSCE 496/896: Robotics
Lab 3: Visual Servoing, Localization, and Navigation

Instructor: Carrick Detweiler
carrick _at_ cse.unl.edu
University of Nebraska-Lincoln
Spring 2011

Started: October 13, 2011
Gripper Design Due: October 28, 2011
Lab 3 Due Date: November 4, 2011

1 Overview

In this lab you will create the design of a gripper, which you will build in the next lab. You will also start to work with cameras on a gumstix processor (`www.gumstix.com`). You implement visual servoing to follow easily identifiable self-similar visual landmarks (barcode-like tags). You will also implement a global localization system for the hovercraft by identifying the location of landmarks with known locations.

2 Gripper Design (25pts.)

NOTE: the writeup and design from this section is due October 28, 2011. We will discuss your design on this date in lab and I will order the parts you need. For your final report, you should also include this section and include any modifications to the design you have based on our discussions. I strongly suggest discussing design ideas with me before this due date.

For this section of the lab, you will design a gripper that can be mounted on your hovercraft and can pick up and drop off balls. In the next lab, you will build the device and work on the vision code to locate and identify the balls. Part of the final project will entail collecting and dropping off as many balls as possible. There are no hard constraints, however, you should consider the following items in the design of your arm:

- The complete parts for the arm should not cost more than $100.
- The balls are light-weight, brightly colored balls that are approximately $3\frac{3}{16}$ inches in diameter. Most balls will be velcored to the ground to prevent them from blowing around.
- Most likely, you should construct and mount your arm in such a way that the camera can be used to locate and identify balls.
- You can use up to 3 servos (such as `http://acroname.com/robotics/parts/R276-S03N-SERVO.html` or similar). You can also use other types of actuators, but you should check with me to see if they can be controlled by the hoverboard.
- The gripper should be compliant to misalignment of the ball since it is unlikely that you will be able to locate and track the ball with high precision, especially considering the dynamics of the hovercraft.

Some materials that you may want to consider using are:

- Legos, which are located in the lab (I can show you where). These are considered "free" and do not count towards your budget.
- Erector set components (for instance `http://www.amazon.com/Erector-Multi-Model-Construction-Set/dp/B000A10Z4U/ref=sr_1_1?ie=UTF8&qid=1318535676&sr=8-1`).

- Balsa wood or other easy to work with materials.
- Rubber bands or springs to enable compliance in the gripper.
- Cables or gears to transmit power.
- IR, break-beam, or contact sensors to detect when you have gripped a ball.

For the report you should include at least the following information:

- A list of parts you need, where they can be purchased, and an approximate cost.
- A detailed sketch or model of the gripper.
- An analysis that verifies the motors will be powerful enough to actuate your gripper and pick up a ball that is lightly velcroed to the ground.
- A plan as to where you will mount the gripper on your robot.

3 Visual Landmarks (20pts.)

On the course website there is link to download ROS modules needed to identify barcode-like visual landmarks. Download this code and place it in your ros directory. 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"[1], 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.

This vision code is already installed on the Gumstix, however, I suggest you setup the code on your netbooks for initial testing and to familiarize yourself with the system. 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. Note that when using the gumstix, displaying the debug images requires a lot of network bandwidth, which can significantly slow everything down (so only display a debug image when you really need it).

When answering the following questions, you should perform experimental tests both with your netbook computer camera (or your own computer camera, but please specify what camera you use) **and** the gumstix.

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 on the netbook, how about the gumstix?

Question: What is the maximum range you reliably can detect that a landmark is in the image (gumstix and netbook)?

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?

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 for both cameras.

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

4 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 running on the gumstix, 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). Utilize your arbitrator from the previous lab to enable controlling the hovercraft from your visual servoing code.

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).

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

5 Visual Localization (15pts.)

The landmark vision code only gives you the range to a target (after you calibrate the system to convert between target pixel height to range). This does not provide sufficient information to localize your hovercraft when only seeing one target. In this section, you should implement a visual localization system that determines the location of the hovercraft by looking for and identifying the location of multiple visual landmarks. You can do this by trilaterating based on the known positions of the landmarks. The positions of landmarks will be given in a file with the format `id, x, y`, for example:

```
12, 2.1, 3.4
34, 1.1, 0.9
22, 0.2, 2.2
```

You should create a ROS node that will output the location of the robot given the input file of positions and data from the camera[2]. This node should process information from currently seen landmarks and potentially try to rotate to see other landmarks that may be visible from the current location at different rotation angles. Note that you can simply use range information to the landmarks, or you could also angle information using the gyro to more accurately localize your robot. With the former, you can test your implementation using the netbook camera without having to use the gumstix.

Question: Describe the implementation of your localization algorithm.

Question: What is the minimum number of landmarks that you need to see to localize your robot? Why is this the minimum number?

Question: Do you use more than the minimum number of landmarks if there are more available? If so, how do you integrate this information? If you don't use all of the landmarks, how do you choose which ones to use?

Question: Characterize the accuracy of your localization given different landmark configurations (e.g. seeing three in one frame, one in front and one behind the robot, etc.). What are the sources of error?

6 Navigation

As you might guess, being able to move from one known location to another is extremely useful and will be needed for the final project. It is not required for this lab, however, you may want to get started or at least think about how you could implement global navigation now that you are able to localize your robot based on the landmark tags. One approach is to compute your current location based on the landmarks and then compute the angle you need to travel to get to your target position and use the Tangent Bug algorithm from Lab 2 to navigate towards the target position. As you see new landmarks, you can update your position.

[2]You may want to create a different ROS node that parses this file and stores them as parameters on the parameter server or this node could respond with the location of a landmark given a particular request for a landmark location.

7 Gumstix Camera

Each group has been given a gumstix processor with a camera. These are sensitive electronic devices. Please be extremely careful when handling them to prevent damage. These processors are similar to those that you would find in a smart-phone. They contain a 600MHz ARM processor and have 512MB of RAM. I have preinstalled Ubuntu and ROS on these gumstix.

You need to decide how to mount the gumstix on your hovercraft. I have mounted the gumstix and camera together, you will need to place it in a location that gives the camera a good field of view. You should make sure that the board is securely attached and that loose wires or cables will not come into contact with it. Do not cover the board itself as it gets quite warm when operating.

You can power the gumstix two different ways. First, you can use the supplied wall power supply. This is the preferred method if you are not actively testing the gumstix on your hovercraft. Second, you can use the supplied cable to power the gumstix from the hovercraft. **Do not connect both power supplies at the same time!** In addition, the gumstix is running an operating system and it will not be happy if you disconnect the power without shutting down the OS first. This means if your gumstix is connected to the hoverboard, you should not disconnect the hoverboard power without first shutting down the gumstix. To turn off the gumstix, type the command `halt` at the commandline of the gumstix. It takes a while to turn off, so be patient.

The gumstix automatically connects to the stand-alone wifi router in the lab. You need to plug your netbook into this router using the ethernet cables provided in the lab[3]. Note that this router is not connected to the internet, so you cannot connect to the internet from the gumstix. Since the netbook is connected to the unl network via wifi and the local network via the ethernet cable, you can connect to the internet from your netbook, however, you have to manually tell it to use the wifi link. To do this, run the command `sudo route add default wlan0` from the command line (assuming the wifi interface is wlan0, use the command `ifconfig` to see if this is correct).

You can access the gumstix (once you are connected to the lab network) by `ssh`ing to the gumstix using the supplied ip address, username, and password. If this doesn't work, you can manually connect to the gumstix serial console by connecting the usb cable and then using the command `screen /dev/ttyUSB1 115200`[4] from the netbook[5]. You will then be able to log in to the gumstix from the serial console (note that you won't be able to communicate with it via ROS over the serial port).

From the serial console, you can see which wifi network it is connected to by using the command `iwconfig`. It should list an interface `wlanX` where X is some number. You can try having it reconnect to the wifi network by using the command `ifdown wlanX` followed by `ifup wlanX`. You can also try rebooting the gumstix from the serial console to see if it will reconnect.

The gumstix is configured to connect to the ROS master node on the netbook. This is configured by setting the environment variable `ROS_MASTER_URI`. You can check what it is on the gumstix by running the command `echo $ROS_MASTER_URI`. You can set it by running `export ROS_MASTER_URI=http://HoverControlX:11311`, where `HoverControlX` is the hostname of your netbook. If you want to use a machine other than the netbook as the rosmaster, let me know and I can help you.

The camera on the gumstix has a number of parameters that can be set. The website `http://wiki.gumstix.org/index.php?title=Caspa_camera_boards` lists some of the details. One thing you may want to do is to disable the auto exposure (with auto exposure enabled, the framerate is sometimes extremely slow). You can do this by running the commands:

```
rmmod mt9v032;
insmod /lib/modules/2.6.34/kernel/drivers/media/video/mt9v032.ko auto_exp=0
```

To see all the parameters you can set use the command `modinfo mt9v032`.

[3] I will need to first help you edit the /etc/hosts file on the netbook to tell it what the IP address of the gumstix is. Also I will need to assign the correct IP address to your netbook.

[4] This assumes that the radio for the hovercraft was already connected, if it isn't then you should use /dev/ttyUSB0, but in that case you won't be able to run your ROS code to control the hovercraft.

[5] To exit screen you need to use the command ctrl-a k. If you just close the terminal window, screen will continue to run in the background. You can also just disconnect the usb cable and it should cause screen to exit.

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 email before the start of class on the due date. A pdf formatted document is preferred.

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 `rxplot` 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?

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