1 Dates

Lab date: September 21, 2006

2 Introduction

In Lab 1, you were introduced to the Player/Stage/Gazebo (PSG) robot interface and simulation system. Lab 1 demonstrated how to interact with a simulated robot in Gazebo (reading sensor data and sending control commands) via manual controls (playerv) and/or a client program (lab1.c). Using a laser rangefinder device for sensing the environment, you wrote a client controller that reactively avoided obstacles and exited a enclosure. Although Lab 1 involved a simulated robot, the PSG controller you wrote can (without modification) also control a real mobile robot to perform the same avoidance task.

In Lab2, you will be extending your work Lab 1 control client to perform an object seeking task. In this seeking task, your robot will look for and drive to objects (or “fiducials”) that are recognizable from visual sensing (i.e., camera). To accomplish this task, you will be working with a simulated video camera. While laser rangefinders provide more accurate information, cameras are often more practical in terms of their cost, weight, size, and sampling frequency. Cameras have the additional benefit of sensing color, which you will leverage in this lab.

Given the world, illustrated in Figure 1, your goal for this lab is to create a Player client that continually drives between the red and blue fiducials. The robot should maintain one bit of state, specifying the currently sought fiducial. The robot should move such that it seeks and identifies the current fiducial, drive as close as possible to the fiducial without hitting it, flip the state bit, and continue the process for the next fiducial.

For fiducial recognition, you will use the blobfinder proxy in Player. The
blobfinder proxy examines the data received from the camera and groups like-colored pixels of a particular color into “blobs.” The proxy returns information about detected blobs and their bounding boxes in the image. The specification for the blobfinder proxy can be found from the Player manual online.

Looking to the future, Lab 3 will be the same object seeking task using real-world Roombas instead of simulated Pioneers. The code you write for Lab 2 will likely be a cornerstone of Lab 3 and the first project (Subsumption Pac-Man). Thus, you should take care to write clean, nice, modular code. Don’t say we didn’t warn you.

3 Instructions


- As in Lab 1, templates for player and gazebo files useful for this lab have been placed in /course/cs148/asgn/lab2. These files are a modification of the Lab 1 files to include: a SonyVID30 camera on the simulated robot in the Gazebo world file (gazebo/lab2.world), camera and blobfinder drivers (camera and cmvision) in the Player configuration (player/lab2.cfg), declaring and subscribing to the blobfinder in the player client template (player/client/lab2.c), and the client makefile (player/client/Makefile). Take a look at these files to get a sense of how PSG can be adapted.

- Run lab2.world and then lab2.cfg. In case you have forgotten, the commands are (executed from gazebo/ and player/, respectively):

  `wxgazebo lab2.world`
  
  `player lab2.cfg`

- Open up playerv, using the command “playerv”. The position proxy and laser proxy from lab1 are still available, but we have also added a blobfinder proxy. Subscribe to the position and blobfinder proxies. The blobfinder proxy should open up a window within playerv displaying the currently detected blobs and their bounding boxes. You may also use the “playercam” application to see the camera images directly.

- After selecting the “command” option for the position proxy, manually navigate the robot (using only the blobfinder for sensing) to the red, green, and blue fiducials using playerv.

- Having trouble seeing the blobs in the blobfinder? Thought so. Cameras and computer vision algorithms often have a delicate relationship that is sensitive to many factors, such as lighting, shadowing, motion blur, etc. Even though your eyes can distinguish the various color fiducials, making such distinctions from a matrix of pixel intensities without a human’s

---

1. [http://playerstage.sourceforge.net/doc/Player-2.0.0/player/group_playerc_proxy_blobfinder.html](http://playerstage.sourceforge.net/doc/Player-2.0.0/player/group_playerc_proxy_blobfinder.html)
ability to learn can be difficult. For this example, you probably need to simply add more light to your world to successfully identify objects. Go to your lab2.world file and a second light source. You can copy the specification for 1st light source, with a new location.

- Keep trying manual navigation until you can successfully navigate between the red and blue fiducials using only the blobfinder.

2. Now, let’s write a Player client that performs a similar object seeking behavior autonomously. Your client should have a finite state machine that controls the robot to continually move between the red and blue fiducials. Each state should be associated with seeking and driving to a specific fiducial. Once you drive to a specific fiducial, you should transition the state variable and seek the other fiducial. Some things to remember/think about as you go:

- Included in the lab2.c template, we have marked the location where you should include your control loop for object seeking. In this control loop, you should use information from the blobfinder to identify each fiducial. The blobfinder provides various information about the image (such as its dimensions) and each detected blob (such as its location in the image, width/height, bounding box).

- Build your code up slowly. Start by trying to face the red fiducial. How do you know when you see it? Can you center the fiducial in the image?

- How do you determine you’ve reached the fiducial? There isn’t necessarily a correct answer to this question, although if you stop 4 meters away from the fiducial your control solution is less than desireable. Come up with a reasonable (defendable) solution.

3. Once you have a working client you are ready to be checked off. Find a TA and show them what your robot can do. Here are some interesting things to think about (and implement if you have time in lab or on your own - cause playing with robots is fun, and addicting):

- Does your robot drive forward and turn simultaneously? If not, how could you change your code so it would?

- What would happen if you started your robot in different positions on the map? You can test this very easily by editing your .world file.

- How close can you get a fiducial without hitting it? You can feel free to hit the fiducials for some fun.

- What happens if your robot doesn’t start facing a blob? Can your robot wonder intelligently (without running into walls and covering the space) if it doesn’t see a blob?