CIS 3415, Project-3 (**8-pt**)

1 Description

This project is all about sensors and using sensor data. You will start by using a camera and colored beacons to navigate the Create around the classroom, and then progress to using a laser to help a simulated robot avoid obstacles.

2 Starting with the camera

1. Login to your computer (user name is `student`, password is `student`) open terminal windows, get a copy of the project files (`project3.tgz`) from your instructor, put this on the Desktop and extract the file.
2. The first thing to do is to play with the camera. Plug the camera into the laptop. Run player on the camera configuration file:

   ```
   player camera.cfg
   ```

   which will give you the usual player messages. Then, in a separate window on the same folder run:

   ```
   playercam
   ```

   This runs one of the player utilities.
3. What you should get is a little window popping up on the screen showing you the view from the camera.
4. You may also see rectangles of blue, green and/or red. This is the result of running some image processing to detect blobs of color (just like we talked about in the lecture).

   This particular blobfinding code is part of the CMVision system from Carnegie Mellon University.

3 Using camera and robot together

1. This time your starting point is the file `blobs.cc`.
2. Start by running it. To do that, plug the robot into the laptop and start up player:

   ```
   player roomba+camera.cfg
   ```

   Then you can compile and run blobs in your second window:

   ```
   ./build blobs
   ./blobs
   ```

   And you may want to start a third window so that you can run `playercam` also.
3. `blobs` will either report it sees no blobs, or it will report all the blobs it sees, including information on color, area, and the x and y coordinates of the blob in the image.
4. The challenge is to take `blobs.cc` and edit it so that the robot will first search for a beacon with the red band. It then drives toward that beacon with a proportional control (see below for explanation), and stops when it gets within two feet of that beacon.
5. It should be possible to do this just using the data from the blobfinder.

   Hint1: If the camera points directly forward on the robot, then if the blob is in the middle of the camera image, the robot is facing the beacon. If it's off center, the robot should turn to make the blob shift towards the center.

   Hint2: The area of the blob grows (non-linearly) as the robot gets closer to beacon, which could be used as a surrogate for distance.

[**Note1**] *Proportional control*, which is often effective for this kind of task, has the robot move more quickly when it is far from the position/orientation it is aiming for, and more slowly as it gets closer. That way the robot doesn't overshoot its target much, also it doesn't take too long to get into the neighborhood of the target.

[**Note2**] The variable that matters to you is of the type `player_blobfinder_blob_t` (see `blobs.cc`), which is returned by `GetBlob(i)`. The `color` member variable represents color in the following format (32bits):

```
        00000000 | red(8-bit) | green(8-bit) | blue(8-bit)
```

As you can read from `blobs.cc`, there is a (`short`) cast, which retains only the lower 16 bits of the color information. The color calibration file specifies that only blobs of red, or green, or blue can be detected. This means if the actual color of the blob detected is red, the "short" cast will produce a color value of `0`. Blue color corresponds to `255`, and green is `-256`!

6. When you are done, save your program as **(your-names)-proj3-part1.cc** and make sure you put your name in the comments.

7. You'll need to submit this to me after you are done with the project.

4 Now the laser

1. The second part of this project is to use a laser with the simulated Create.

2. To run the world, use:

```
player world3.cfg
```

3. This pops up a smaller simulator window than before (which hopefully will fit on the screen of the laptop), and the robot.

   This time the robot has a small blue blob on it. This is the simulated laser.

4. Compile and run the controller:

```
./build laser-roomba
./laser-roomba
```

   and watch the robot run down the corridor and hit the wall.

5. The blue shape that is projected from the robot is the area scanned by the laser. Watch how it changes as the robot moves, and notice how the different values returned by the laser (printed in the terminal window) change also.

6. The challenge is to make the robot drive along the *middle* of the corridor from bottom left to top left, and to do this without hitting the wall, and without using odometry.

7. When you are done, save your program as **(your-names)-proj3-part2.cc** and make sure you put your name in the comments.

8. You'll need to submit this to me after you are done with the project.

5 Handling Proxies

All the relevant commands for the blobfinder and laser proxies are demonstrated in the code.