Computer Vision Project Writeup
Marena, Dara, Mike

What was the goal of your project?

The main goal for our project was to enable a Neato to identify objects and avoid them based off of the input from its camera. Our minimum viable product was enabling the Neato to do general obstacle avoidance around small objects using the camera. The Neato's LIDAR is good at picking up large objects, but not small ones, and that was our motivation. Our stretch goal would be to add navigation and path planning into the equation to create a robot that can intelligently identify objects and get from point to point by avoiding obstacles.

We also wanted to learn a lot more about some of the visual processing packages that are available in ROS and OpenCV and how they could be put to practical use. Collectively, we wanted to combine our previous projects using obstacle avoidance, path planning, and localization. We did not end up involving path planning or localization explicitly, but our final result was a combination of wall following, LIDAR obstacle avoidance of large objects, and camera obstacle avoidance of small objects for a neato that happily navigates around a controlled environment.

How did you solve the problem? (Note: this doesn't have to be super-detailed, you should try to explain what you did at a high-level so that others in the class could reasonably understand what you did).

In order to accomplish small obstacle detection with the camera, we first looked into selecting the object using a preset color range and then implementing contours to select the largest contiguous blob that would constitute an obstacle. We found this to work fairly well, but we did not want to hard-code the colors of our obstacles in order to allow for a variety of obstacle colors.

This goal was accomplished by using k-means clustering to create a color histogram of the environment. The histogram generated bins of colors from the image. We assumed that in our controlled environment the largest two color bins represented the floor and walls. The remaining bin, if it represented a small enough portion of the image (less than 10%), was then the target color of our object. The qualification that the obstacle color be less than 10% of the image helped avoid false positives created by the wall and

the floor when there was no obstacle in view. By running contours on a color range centered around our target color, we could extract the likely center of our object.

Incorporating this data into the LIDAR navigation required a finite state controller. Both the camera and LIDAR control the direction of the robot, based on a weighted priority variable. A delay between switching states was implemented to allow sufficient time for each controller to complete its desired task.

Describe a design decision you had to make when working on your project and what you ultimately did (and why)? These design decisions could be particular choices for how you implemented some part of an algorithm or perhaps a decision regarding which of two external packages to use in your project.

A major design decision was deciding to heavily control our environment. We found out that the background clutter was crippling the efficacy of our code dramatically and we needed to take drastic measures to ensure a simple, clean background. This manifested itself in the form of a giant cage we constructed out of design review boards we covered in paper.

How did you structure your code?

Our code was contained in a single file that had one main class. Upon initiation, our class ran in a while loop that contained the logic for our finite state controller. Our callback functions in the class updated the variables that controlled the Neato from the while function.

What if any challenges did you face along the way?

Our biggest challenge was controlling the background noise and clutter of our images. We found early on that it was very important to control your environment in order to get clean data from the camera. Our solution was building a “cage” for the robot out of white walls to box it in and remove the chance of background objects messing with our obstacle detection.

What would you do to improve your project if you had more time?

We would have liked to have a smarter histogram function. As it stands, our code always segments the image into three color bins. While this worked well for our environment, a more robust algorithm would allow for a flexible number of bins that was somehow optimized to best identify small objects in its view. This would allow the Neato to navigate more cluttered environments and improve the usefulness of our algorithm greatly.

Did you learn any interesting lessons for future robotic programming projects?

Computer vision is very difficult. It is very hard to get clean data from a robot, especially if your reference frame is constantly moving, as it is on a robot. In addition, a ground-level view from the the image frame of a robot in the real world is cluttered with objects of a variety of colors that are themselves subject to change under different lighting conditions and angles. Computer vision should probably be used sparingly on a robot like the Neato in the future.

Also, image processing algorithms can be slow, especially in Python. Given that Python is our language of choice and communications were taking place over the network, we stumbled a few times on how slow some functions run when processing live video. We found ways to optimize by limiting the amount of processing necessary, but this is definitely something to be aware of when designing elaborate algorithms.

Citations:

http://opencvpython.blogspot.com/2012/06/hi-this-article-is-tutorial-which-try.html