Neato Plays Fetch

Computational Robotics: Computer Vision Project

Authors: Kaitlin Gallagher, Christina Fong, Victoria Preston
Date: 11/7/14
Time spent: Uncertain

TABLE OF CONTENTS:

I. Project overview

For this project, we decided to make the Neato robot play “fetch”. When a ball is rolled down a hallway past the Neato, the robot should use the onboard camera to identify the ball and its relative location and then move towards the ball until it has reached or lost sight of the ball. This allows us to explore optical tracking and visual servoing.

Our minimum viable product for this project was to have the Neato autonomously navigate to a stationary target object (ie: a ball). Our main goal was to have it autonomously navigate to a slowly moving target object in its field of view.

Our learning goals included:

- gain more knowledge about computer vision targets
- learn how to process an image under real-time, high-speed conditions
- learn how to extract coordinates from obstacle data
- practice best methods of coding in Python and ROS
- better understanding of accuracy vs. computational costs
- learning the difference between OpenCV’s descriptors (eg: SIFT, SURF, ORB, etc.)

On the whole, we accomplished most of these learning goals with the exception of the latter two. Due to the way that we approached object recognition, we did not use keypoints and therefore did not spend much time looking into OpenCV’s descriptors. We also did not run

into much of an issue with computational cost in our algorithm. It was observed however, that the more 'vision' we displayed on our computer screens, the slower the processing speed of the robot. There is a clear relationship between how fast a process can be, and how much can be displayed. Due to OpenCV's packages however, we found that computational cost was kept minimal.

II. Implementation of object recognition

In order to navigate towards the target object, we need to successfully and consistently recognize it -- consistent recognition is particularly important when the target object is moving. We implemented object recognition in our image_converter class, which subscribes to the /camera/image_raw topic and publishes to the /processed_image and /ball_coords.

callback function

The callback function executes every time the /camera/image_raw topic updates (every new frame). It does basic pre-processing of the frame, including converting it to greyscale and then running Canny edge detection, before calling find_circles.

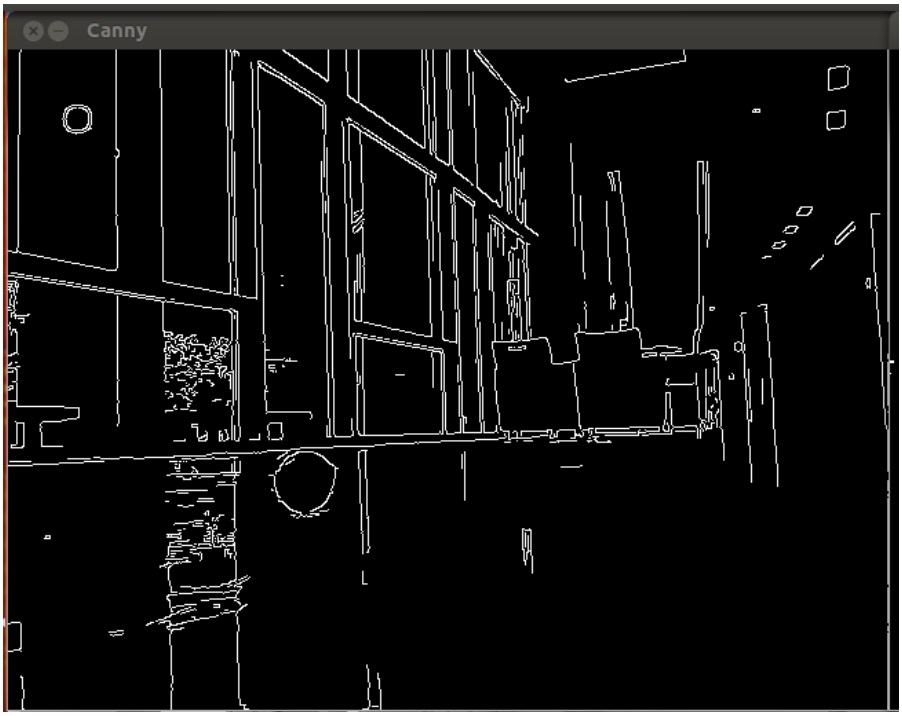

Figure 1: Camera image after being pre-processed, with Canny edge detection

After find_circles executes, it publishes the updated processed image and ball locations.

find_circles function

The find_circles function applies the Hough circle transform to the Canny edge-detected imagine in order to identify potential balls. Because this has a high rate of false positives, we

then cross-reference it with red areas in the image (using an HSV image mask) to narrow down on the correct target.

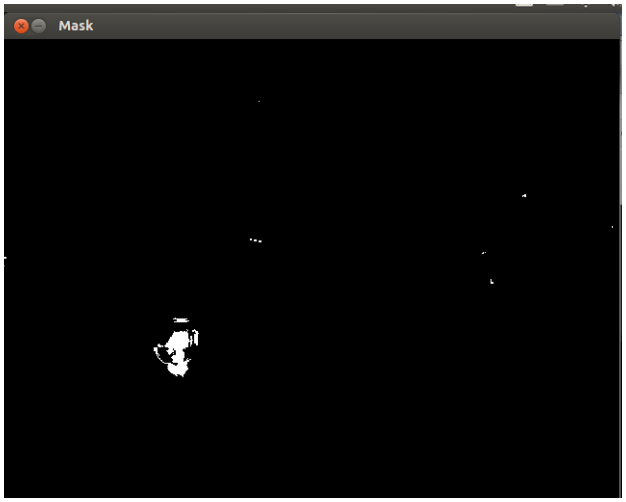

Figure 2: Results of HSV mask

Figure 3: Correctly identified target object

We then update ball_location with the center of mass of the circle and its radius as a Vector3 of (x, y, radius).

III. Implementation of visual servoing

We implemented visual servoing in the ball_follower class, which subscribes to the /ball_coords topic and publishes to the /cmd_vel topic. Every time the /ball_coords topic updates, it executes the coordinate_to_action method. In this method, we calculate the depth (distance from the ball to the robot) based on the radius in pixels, and the x and y locations of the ball (proportionally to the frame).

The robot's linear velocity is set to be proportional to the depth, so that the robot moves more slowly as it draws nearer to the target. Angular velocity is set based on the proportional (x,y) coordinates of the ball location -- the farther from the center the ball is, the faster the robot will turn.

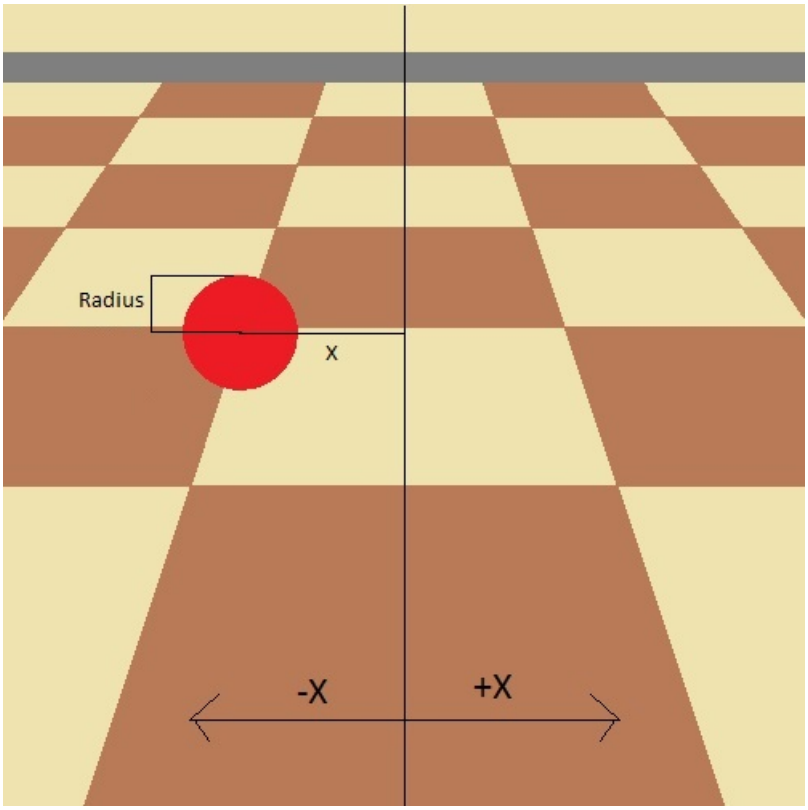

Figure 4: Visual servoing implementation

IV. Challenges faced

Hardware

We had some minor difficulty interfacing with the Neatos without a lot of lag, although this could be overcome using ad-hoc mode. Motion blur from the cameras were our major hardware challenge faced, as the blur made it difficult to identify the ball by both shape and color. This was resolved by a combination of slower robot motion and changing the Raspberry Pi cameras to sports mode, which vastly decreased motion blur.

Software

On a software side, it was a challenge to determine parameters for the Hough circle transform. As mentioned previously, we were unable to work out Hough circles parameters that let us accurately and consistently recognize the ball without any false positives, even at different distances from the robot (as its observed radius changed). Ultimately, we chose to

address this problem by allowing more false positives and then using HSV masking to narrow down on the desired object.

We also ran into difficulty determining the upper and lower bound ranges for “red” in the HSV color mask, especially under different lighting conditions. Lighting changes in general is a major failure in the robustness of the code, however, if we were able to auto-whitebalance the cameras at will, this problem could be avoided almost entirely. This is an interesting feature to add to the neato cameras in the future.

V. Future improvements

There are many ways that we would like to extend and/or improve this project with additional time. These include:

- Ability to extrapolate target object motion based on observed behavior, so that the robot can predict where it will be at some point in the future
- Ability to follow irregularly shaped targets (although this would require an overhaul of our current object recognition algorithm)
- Ability to follow an object moving quickly, following an irregular path, and going in and out of frame
- Ability to autonomously self-calibrate (color) under different lighting conditions

VI. Applications to future robotics projects

The things that we learned in this project are applicable to many robotics projects that involve computer vision. Learning about different ways to recognize target objects -- eg: keypoint matching, shape matching (a la Hough circle transforms), edge detection, color filtering, etc. -- was very valuable, because we gained a better sense of the strengths and weaknesses of certain approaches.

We also learned that lighting is extremely important in computer vision. Small variations in lighting had large effects on our ability to recognize colors, requiring repeated calibration.