Lab 3: Color Tracker (100 Points)

Due electronically Tuesday 10/31/2017 at 11:30AM

You are required to submit all code samples and schedule a demo with the TAs to show that your code functions as advertised. **We will NOT be running your code**. If you do not participate in a demo you will not receive a grade for this assignment. There are no exceptions. Demos should last no longer than 10 minutes so please come prepared.

Assignment

In this assignment, we will explore vision processing with a mobile robot. We have some tasks below, ordered by difficulty.

Description

In this task, we will explore tracking a distinguished color marker. Create a target with a prominent unique color that the robot can image. Hold the target on a stick or other method and have the robot image it and track/follow the target as you move it in the robot's workspace. Your robot should maintain a constant distance from the target, moving toward it if the target starts to go farther away, and backing up if the target is moving towards the robot. The robot will also need to rotate and translate if the target moves at an angle – the idea is to keep the target centered in the image.

Part 1

You will initiate the color tracker manually. Take an image and have the user click on the image and draw a rectangle around the region you want to track. By computing statistics on the pixel colors in this region, you can create color thresholds to segment the target. Which color space you use may improve your system's performance. The standard Python OpenCV BGR color space (Blue, Green, Red) is more susceptible to illumination changes. You can try another color space called HSV (Hue, Saturation, Value). There are conversion functions in OpenCV to convert from one colorspace to another (e.g. the python OpenCV command:

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

will convert an image "frame" from BGR color space to an image hsv that uses HSV color space.

Part 2

Using the thresholds above, create a binary image: every pixel is either white (target regions to track, color 255) or black (non-target regions, color 0. Then find the largest target blob in the image, and calculate its centroid and area. To help you get rid of noise and small artifacts, you can use a series of image filters to a) Gaussian Blur the image –smooth it, b) Erode the image (morphological operation) to get rid of small blobs and c) Dilate the image to amplify and connect smaller regions into a larger blobs.

Part 3

Now, take a new image and again threshold and binarize the image. Calculate the centroid and area of the largest blob in the new image, and compare it to the previous centroid and area of this blob. If they don't change, the target hasn't moved. If they do, you need to move your robot to either increase or decrease the blob area (move forward and back) and rotate to keep the blob centered (centroid location).

You will have to play a bit with the gains on your robot's movement, i.e. how fast and far to move to re-adjust the image. Keep in mind you are doing this continuously as each image frame is read in real-time. Given the web link for the images, you probably will only get 2 or 3 frames per second which will help determine how fast to move the robot. You can also reduce the camera resolution to allow faster processing and a possibly higher frame rate.

Part 4

Move the marker to make the robot follow you. Show that it will stop when you stop, and turn when you turn, etc. Record a youtube video of the color tracking working that is no longer than 20 seconds.

Tips

- A good strategy is to do the image processing on your laptop first to test it. When it is working you can then port the code to the Raspberry Pi and integrate the motion control.

Instructions for submission

All assignments must be submitted with working code and a README that states that a section was completed and/or answers the questions listed in the section for the section of this assignment. Your README should also include All code must be written in Python, specifically python 2.7.

Your submitted code should be in a tarball (see here for how to tarball) and should be stored in a directory of the format 'hw3_<uni1>_<uni2>_<uni3>.tar.gz', where <uniX> is replaced with the uni of each teammate (i.e. hw3_djw2146_djw2156_djw2166.tar.gz). The directory of this folder should match the following:

hw3_djw2146_djw256_djw2166

-- README.md

-- colortracking.py

Your README should follow the following format:

“””

This is a Lab 2 submission for group **XXX**
Team Members: *member_1*(*UNI_1*), *member_2*(*UNI_2*), *member_3*(*UNI_3*)

Youtube link for color tracking demo: <link>

<Description of adjustments to color tracking Algorithm>
<Descriptions of assumptions based on robot>
“””
You should also include descriptions of any novel methodologies you used to implement the solutions for this assignment.

Any deviations from this will not be considered for grading.

Extra Credit (10 points)

Have your robot track a flesh colored object (i.e your hand!) and determine from user specified hand gestures if the robot should move forward, backward, turn left or turn right.