ME 499/599: Computer Programming for Mechanical Systems

Winter 2015

Lab 7: Manipulating Images

Purpose

The main goal of this lab is to give you some experience dealing with an external library for Python. We're not going to tell you exactly *how* to do things for this lab; you should look at the documentation, and figure out the right function calls for yourself. This is preparation for the future when you might not have someone looking over your shoulder who already has the answers.

This lab is harder than some of the previous labs, and will use all of the skills you've built up over the term. Accordingly, we're going to give you two weeks to do this lab, and it will count for twice the points of the previous labs. Make sure you bring your questions to class.

Preparation

Make sure you have the `grabber.py` working on the computer you're using. Some of the lab machines are misconfigured (we're working on that), for which we apologize. If you're using your own machine, then it's up to you to make sure the grabber code works on it.

For this lab, we're going to get you to use the Python Imaging Library (PIL). This is a relatively common library that lets you work with images, drawing on them, and performing a number of manipulations of the image data.

Assignment

1. Take a look at this code for an example of how to grab, display, change, and draw on images. We're going to ask you to grab images and do some manipulations on them for this lab. For all of this lab, you should use the

Image class.

2. Write code that turns all of the grass in the image blue. In the simple case, it should turn the grass bright blue, (0, 0, 255). In the harder case, it should turn the grass a shade of blue that looks "natural" (if grass was actually blue). Display the image before and after the manipulation. You should be able to easily select whether it turns the grass bright or natural blue easily in your code.

3. Calculate an image showing motion in the image. Take two images from the webcam, separated by a small amount of time. Subtract the pixel values in the one from the other, and threshold them based on absolute value. Display this image, and verify that it corresponds to moving things in the image.

4. Write code that draws a circle around every person moving in the image. Display the original image, and the image with the circles around the people.

5. **599 students (extra credit for 499 students):** Modify the code from the last question to draw an ellipse around people who are moving, such that it appears to be on the ground.

Grading

There are 14 points available in this lab for 499 students (plus 3 extra credit), and 17 for 599 students. Upload to TEACH as usual.

1. Turning the grass bright blue (2 points). Not turning much else blue (1 point). Not turning *anything* else bright blue (1 point). Turning the grass a natural shade of blue (2 points). [6 points total]

2. Correctly calculating the motion image and displaying it (3 points) [3 points total]

3. Correctly clustering the motion points (2 points). Drawing a circle around the clusters (1 point). Not drawing a circle around things that aren't people (2 points). [5 points total]

4. Drawing an ellipse, rather than a circle (1 point). Ellipse centered on bottom of cluster, corresponding to feet (2 points). [3 points total]

Thoughts

1. Look at the natural values of grass in (r, g, b). Your "natural" values for blue grass should look like these, with the same sort of variation in them. There's no established way to do this. Come up with something that looks pleasing.

2. You're probably going to want to come up with a function `grass()` that takes a color tuple, and returns `True` if it's a grass color, and `False` otherwise. Look at the ranges of grass in an image, and see if you can come up with a rule for this. One way to do it is to define an area that you know to be grass (a rectangle in the image), and look at a min and max values in each of (r, g, b) for this area. Then, any color that falls into this range could reasonably be called "grass". Remember that the apparent color of grass will change, depending on the lighting conditions and the time of day.

3. If you're coloring some not-grass blue, then you might want to make your `grass()` function more complex, or only look at areas in the image where you know there are grass (ie, not the sky).

4. If you have two images `p1` and `p2`, you should calculate their difference by taking the absolute value of the euclidean distance between the pixel values `abs(d(p1[x, y] - p2[x, y]))` and storing that in an image (either overwriting the second one you captured, or making a new one). This will make it easy to display.

5. Threshold the image, so that only pixels that have changed "a lot" appear as motion. This will get around measurement error in the pixel values.

6. Clustering points is tricky, and we'll talk about it in class on 2/24. If you know how to do it, go ahead and get started. If not, wait for the class.

The Rules

Everything you do for this lab should be your own work. Don't look up the answers on the web, or copy them from any other source. You can look up general information about Python on the web, but no copying code you find there. Read the code, close the browser, then write your own code.

Page written by Bill Smart.