Maggot.py

For instructions see page 2

Background Info

The goal of this project is to track the distance traveled by a *Drosophila* Larvae on a petri dish. The agar of the petri dish was made black using activated charcoal so that the contrast with the larvae would be higher. Modules used were 'os' (to interface with the operating system and set the directory) and 'math'(provides useful mathematical functions). Packages used were 'numpy' which was useful in making the video frames into large arrays, and OpenCV which has a lot of useful functions for video analysis. All the functions that I made are at the top of the script lines (0-42) to easily find them. It is important to note that when OpenCV creates a window the buttons do not work and the size of the window cannot be manipulated with the mouse. Because of this I've rescaled the videos using my function rescale_frame. Then, through a series of functions in OpenCV I gray-scaled the original video, used a "closing morphology" (which erodes the pixels of the images and then dilates the image to form what looks to be a more pixelated image), and then made it into a binary image and tinkered with the parameters so that the larvae would show up as a single, smooth, white blob on the screen. I then used a function in OpenCV called SimpleBlobDetector which, as the name implies, uses a series of parameters set by the user (in lines 50-80) to create a "detector" to identify and label the appropriate blobs with a red circle at their center. The functions and blob detecting are all set up before line 90, after which they are actually called in the order needed to run the program. After setting the parameters so that the larvae would be found I worked to find the distance it travels. The way I went about doing this was to extract the x,y coordinates for the blob from the first frame (using an **if statement**) to set the starting point at the **variables** x_0 and y_0. For the second frame (again, an **if statement**) I again extracted the coordinates but set x and y to the **variables** x_1 and y_1, respectively. I could then use the formula for the distance between two points to get the distance traveled from frame 1 to 2. Because all of the functions were being iterated in a **while loop** the variables would change so that x_1/y_1 would be the coordinates of the blob in the present frame and x_0/y_0 would be that of the immediately former frame. The distances would be summed together in the variable 'Totaldistance' which was to be overlayed on the display as well as printed on the Console. The coordinates of each iteration, however, were saved onto two lists, x_list and y_list, so that later a **for loop** would draw lines between all the points, in green, to make the trail that the larvae has traveled. This is the progress I've achieved in the last month of working on this program using only what I've learned this semester in class and online, with some direction from Ryan W. In the future I plan to also count the number of contractions that the larvae does using the smaller cropped window that appears when the program is run. The cropped window is known as a bounding box and typically is mouse-selected, but I wrote the function 'Bounds_from_center' so that it would automatically generate from the coordinates of the larvae in the frame. To do so I had to create a **tuple**, 'r', which would serve as the dimensions for the image to be cropped. From there I

intend to add contour lines for easier visualization and use corner detection to identify the head and tail of the larvae (useful in counting contractions and other behavior).

Instructions:

1. Save the video onto the computer (make a folder for it, still-frames will save here)
2. Change the directory (line 6) to the location of the video
3. Install openCV (I recommend using “pip-install” for ease)
4. Once all packages are installed, run the entire script
5. Three windows will pop up (check taskbar):
    a. Original video (resized)
    b. Binarized video with tracking information
    c. A cropped section of the binarized video without tracking information
6. While running you may end the program two ways:
    a. Letting the video time out
    b. Pressing the ‘q’ key on your keypad
7. You may also save a still-frame and save it into your folder in two ways:
    a. Letting the video timeout (automatically takes an image)
    b. Pressing the ‘p’ key on your keyboard

Note: If you end the program before the video times out it won’t save an image unless you press ‘p’ on your keyboardbut will print on the console the distance traveled so far.

Below is an example of how the three windows should look toward the end of the video: