

Due February 19th @ 7pm

Two part assignment.....using Python and OpenCV you need to do the following two parts (Should be two different applications).

Part One:

Tracking an object

Steps to do it properly.....

1. Capture live video from Webcam
2. Display live video
3. convert BGR to HSV
4. Display the HSV video
5. Click on HSV video and capture the HSV values at the location clicked, and a few other local values of the item you are going to track.
 1. use a MouseCallback to print the HSV values.
6. Using Sliders create scalars for the min and max values you want to track
 1. a Scalar will be a numpy array (np.array) that takes 3 values for minH, minS, and minV.....then a second scalar to catch the other three Max values
 2. create 6 trackbars, createTrackbar with callback methods to set your six variables
7. Use the OpenCV inRange method to find the values between the scalars from HSV image and the result will go to a grayscale image (make it a binary image, white/black).
8. Dilate, erode the grayscale image to get a better representation of the object you are tracking.
9. Display the original image and the binary image where everything is black except for the object you are tracking. The tracked object will be white.

Done Part One....

Part Two

Detect Motion

The Steps I took in the example I showed in class.

[Video Explanation](#) (this shows it working in Visual Studio, but we're using Python, I'll show that in class, which will be a video now since I'm in Billings).

1. Capture an image
2. Create blank images:
 1. grayscale image with proper dimensions
 2. 32f, 3 channel image
 3. a capture clone we'll call image1
 4. One to hold the result from the absDiff function. (absolute difference)
3. while loop to capture images.....
4. grab new frame
 1. I brightened image a bit first which helped
5. blur the image
6. take running average of frame: **accumulateWeighted**
7. swap running average result from step 6 to same bits as frame: **convertScaleAbs**

8. Take difference, built in OpenCV function to do a diff between two images.
9. convert to grayscale
10. Threshold grayscale (using a low number)
11. Blur grayscale image
12. Threshold grayscale again (using a high number)
13. find contours: **findContours**
14. Use contours to find significant blobs,
15. draw polygons of blobs
16. use dimensions of blobs to draw bounding boxes and center on original image

Done