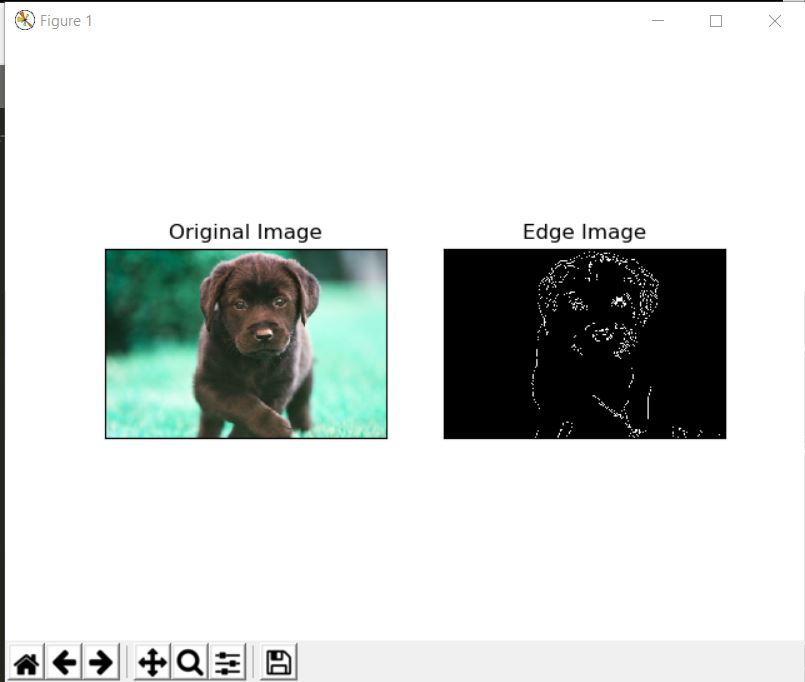
**-EE180DA Report: Lab 1**

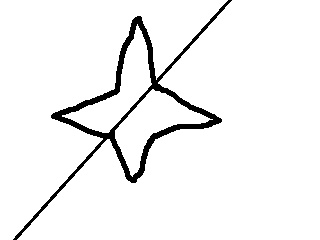
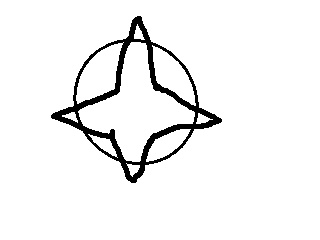
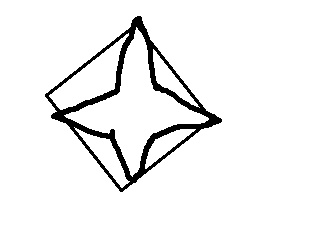
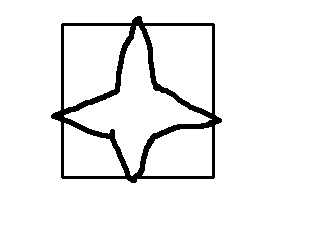
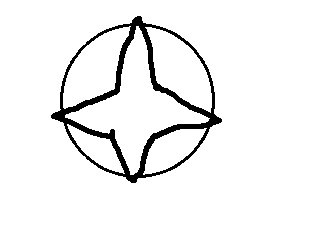
So I was able to do all three static image exercise, and was also able to do a video camera exercise. Here is what I implemented during Friday’s lab, with no difficulties *except* for the Mesi image of Template Matching. The code can be found at:

<https://github.com/AndrewBraunEE/EE180DA-Lab/tree/master/Lab1/images>

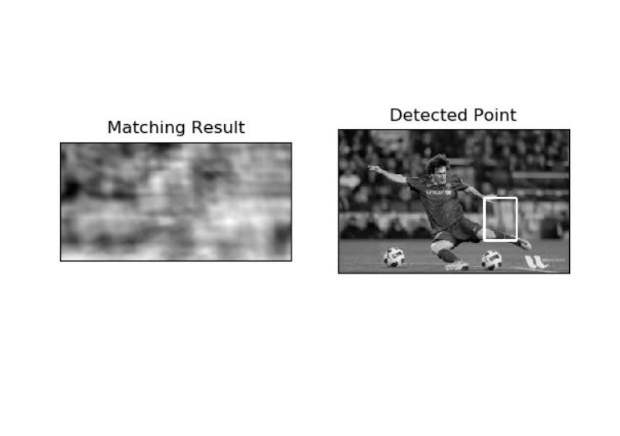
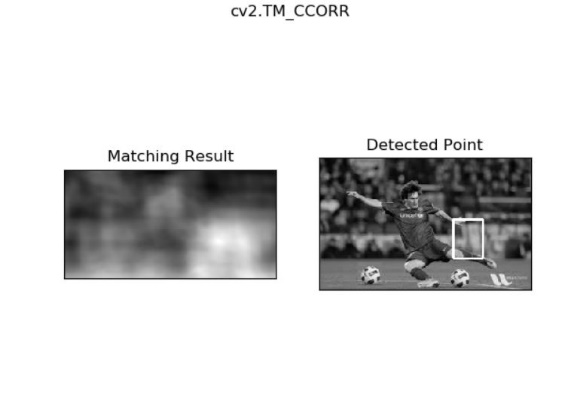
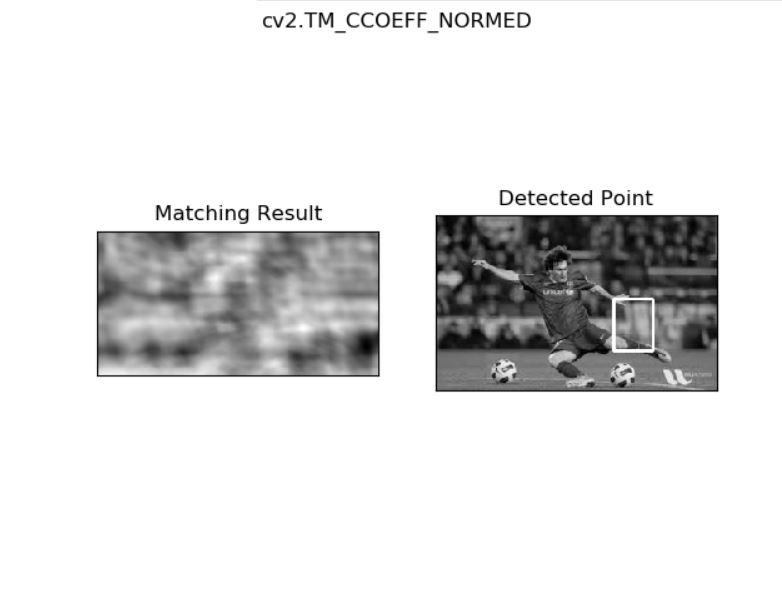
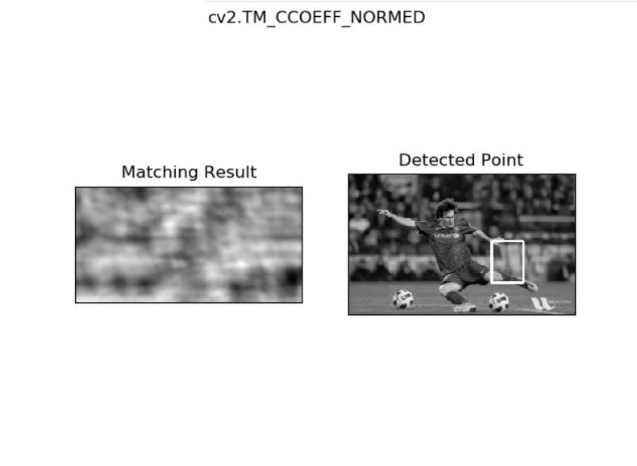
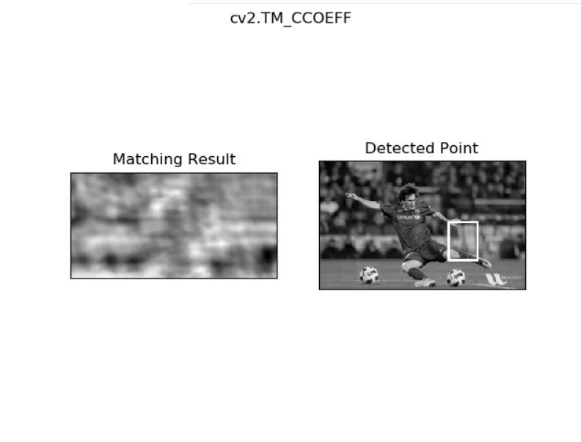
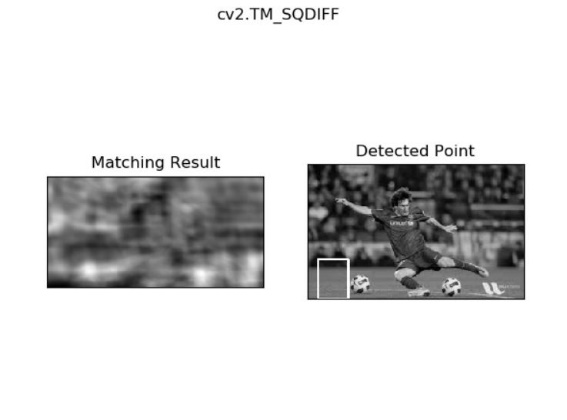
1. Static Images
   1. Edge Detection
      1. Importance: Edge Detection can be very important for our project! Since our project will likely use a webcam for our graduation hats, it might be useful to detect shapes (eg other grad hats). Since the eventual goal of this class is to display an image on top of our grad hats, it may be important to detect if the person in front is “wearing a grad-hat” and is “displaying specific pixels” on its hat.

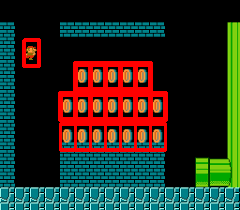


* 1. Contours / Convex-Hull
     1. Contours are super important for being able to visualize what is occurring in our image processing algorithm. Being able to bound an object in a rectangle or an ellipse allows the user to visualize what is occurring with his or her algorithm, which will be great for debugging!
     2. Additionally, checking for convexity defects may allow us to generate more information that can be used in parallel with edge detection to see how much / how many convex defects an edge may have.



* 1. Template Matching

1. An interesting thing about template matching is that the OpenCV example worked on the Mario example with template matching on coins. However, on Mesi’s face, it kept detecting random gray-scale as matching. Maybe I did something wrong, but it’s also possible that I have the wrong Mesi image for the OpenCV example listed
2. 
   * 1. Template Matching is super important for finding a specific template in an image / webcam stream in the future. So if we know what pixels to expect, we can find where those pixels occur in another image, which will *likely* be more useful when used in combination with other image-processing techniques for our application.



1. Webcam / Video:
   1. Object (Frontal-Face) Detection
      1. No issues while implementing, opencv was able to find my webcam just fine! I used a haarcascade classifier (provided and trained already by the OpenCV library). By doing that, I was able to draw and detect the number of faces in an image frame.
      2. Using classifiers may be super useful in detecting specific objects, which will likely come in handy later for the main project of this course in detecting specific objects (or even hues?).

The only difficulty that I had was that the Mesi template matching didn’t work with my images but worked on Mario. I’m guessing this is likely a bug somehow either in a) I didn’t have the exact same image(s) as the examples, or b) there is a 2-line difference if I recall correctly on the template matching implementations of the Mario example and the Mesi example. Overall, this was a good lab to build some intuition in OpenCV (which is great for our project and any future engineering applications!)