CSCE 485
Bruno Lopes
LAB#6 - Hand Detection

Learning process:

Due to the simplicity and optimal result of the Sobel filter for edge detection I was under the belief that in order to achieve an accurate hand detection and eventually gesture recognition Sobel was the best path to do so, after a lot of research and failed attempts I concluded that the Sobel operator threshold analysis is not necessarily the best strategy to achieve hand gesture/recognition. The light computing power required to run is sure desirable but it becomes irrelevant and inefficient; Irrelevant because a binary threshold transformation alone will suffice the need for identifying a 2D pattern (hand) on a 3D image (video stream), and Inefficient because after the applying Sobel some of the borders were lost inside of the Convex Hull region surrounding the hand image due to too many convexity defects finds. When compared to a simple B/W binary transformation the Sobel filter was not needed, other edge detection algorithms could be of good use for more complex 3D gestures or analysis but after many failed attempts and for the purposes of this lab I chose to abandon the Sobel filter strategy and focus on the hand and fingers detection.

Strategy adopted:

The most efficient way I found to implement a simple hand-detection and fingers recognition using Opencv was using the contourArea, combined with the convexHull shape-descriptor and the convexityDefects to identify what could be a hand. This is a similar approach from Davies (cookies) but instead of seeking perfect convex hulls I was more interested on counting the imperfections from the convexity defects. To properly detect and identify a hand object on a video stream, background elimination was required, I had a few issues using the bottom-up strategy from the frame differencing algorithm we built on Lab#4, so I chose to use our book (O'Reilly Learning OpenCV chapter 9) background elimination codeBook implementation, which eliminates the background from a video feed by calibrating an empty static background for a few seconds to then differentiate the static frames with the moving hand, similar principle as the frame differentiation lab just on the Opencv more practical implementation. Once the background was removed I used the Opencv convexHull shape-descriptor to detect the overall boundary area of a hand and then the convexityDefects to find the space between each finger (cracks on the cookies). This simple but effective process allows us to identify the hand object and how many fingers are displaced.

Conclusion:

In the midst of finals exams and assignments I had the brilliant idea of clicking yes to an Ubuntu update which resulted on losing my OpenCV 2.4 instance, after some stress I end up reinstalling the OS from scratch and then the same version of OpenCV as I had before. This time wasted allowed me to dive a bit deeper into OpenCV different versions and added functionalities, I enjoyed to the opportunity and used to familiarize myself with the library which allowed me to eventually finish this project.

This simple strategy adopted seems to be common and popular on OpenCV.org forums, it is widely adopted using Python and it works well. The Opencv library makes it simple (although not always optimal) to implement object recognition with background elimination for simple shapes. There are several different ways to accomplish a hand detection, I believe that I spent too much time trying to make line-contours to work when a much simpler solution was available. I am glad with the final result and proud to apply everything learned, it was a great opportunity and I plan to explore more of OpenCV on my own specially on mobile platform implementation, which seems to be a new trending topic.