Facial Recognition and Eye Tracking

Final Project APPM 4380

by

Zachary Vogel

December 7, 2015

Introduction

Motivation

- Cameras/LIDARs and a few other sensors surrounding a room to ID, image, and model people
- Use Virtual Retina Display to project an image straight into each users eye
- This creates a virtual reality tool that has a variety of uses, including medical applications, gaming, computer use, and general media consumption
- It would be even better with real time Haptic Feedback

Possible Projects

- Identification of Targets
- Facial Recognition and Eye Tracking
- Tracking and Modeling the Human Body
- Haptic Feedback
- Virtual Retina Display Functionality

Outline of Project

- Facial Recognition
  - Face Detection
  - Facial Tracking
- Eye Tracking
- Facial Recognition with Eye Tracking

Facial Recognition

Facial Recognition

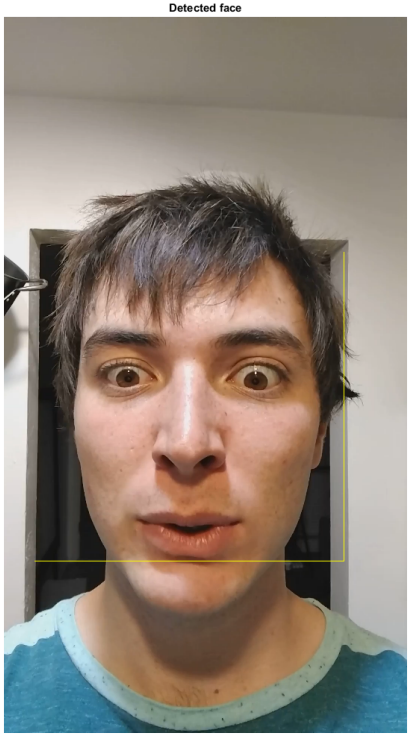
- Viola-Jones Object Detection to find a face
- Determine features of the persons face, known as eigenfeatures
- Keep track of the eigenfeatures as the person moves
- Keep track of the location of eyes in the image as well

Viola-Jones Object Detection

- Identify basic common features of whatever you want to detect
- These come in the form of one area of an object being brighter than another
- Than you sum all the pixels in two or more regions and take the difference
- Viola-Jones uses two rectangle areas

Viola-Jones Continued

- Lots of possible features, so you want to be smart about which features you pick
- Viola-Jones algorithm uses the AdaBoost algorithm to learn given features for an object type
- There is a process of training the algorithm too understand the features it needs to look for in a given object



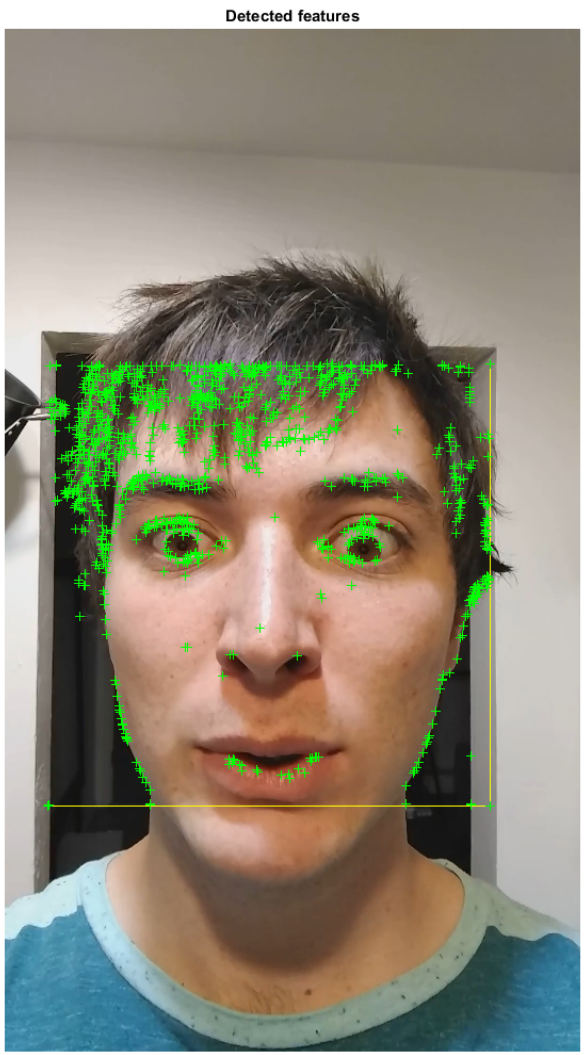
Tracking the Object

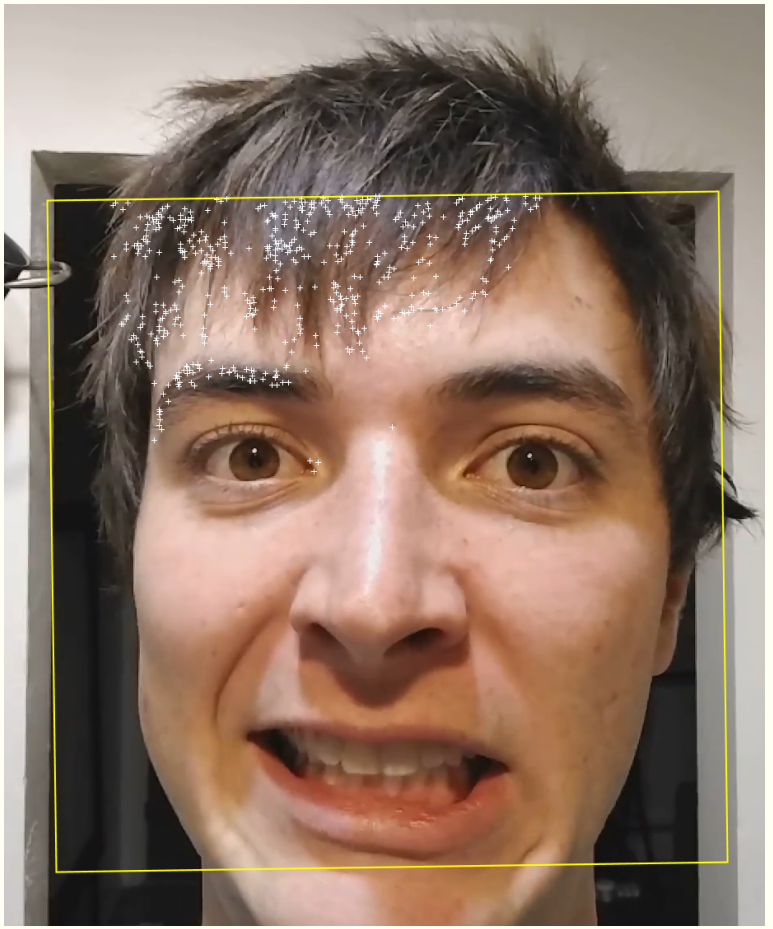
- It's expensive to do Viola-Jones in real-time
- Use location of face to find unique identifiers
- Then use the Kanade-Lucas-Tomasi feature tracker to track these features
- Trying to find an H that minimizes the difference (usually normed) of $F(x+h)$ and $G(x)$

$$w(x) = \frac{1}{|G'(x) - F'(x)|}$$

$$h_0 = 0 \quad h_{k+1} = h_k + \frac{\sum_x \frac{w(x)(G(x) - F(x+h_k))}{F'(x+h_k)}}{\sum_x w(x)}$$

Tracking Results





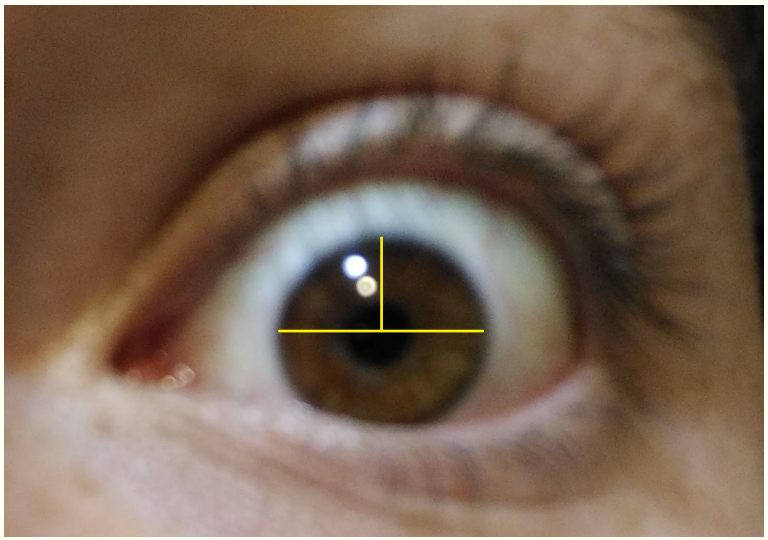
Zach’s Eye Tracking Ideas

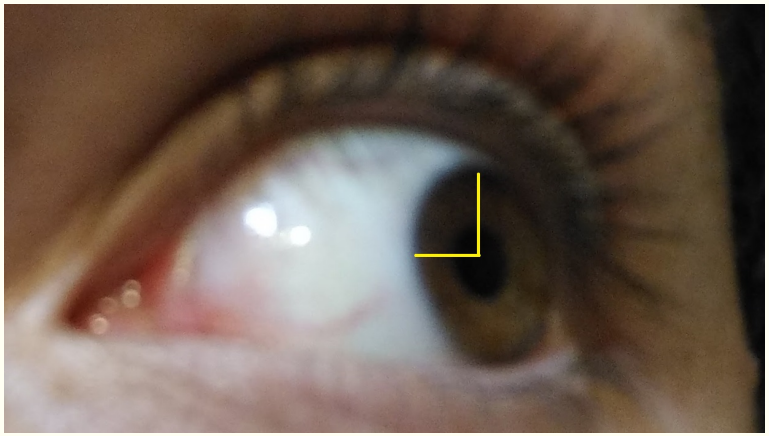
Model

- Two original ideas, measure outer pupil points, or measure whitespace
- Measure the distortions of the pupil relative to the eye looking straight at you
- Measure area of whitespace on either side of the eye and use known ratios
- Also split the eye between top and bottom to do the same thing

Radius of Iris

Here we use the inverse cosine to get an angle for both axis. In the first image, 379 pixels across, 174 pixels to center from top. Second image, 133 from top to center, 104 from left to middle. Looking 56.7 degrees to the left, and 40.1 degrees up.





Whitespace of eye

- The idea here was to measure the whitespace(sclera) on either side of the eye
- Find the area of both sides and use a LUT of previously taken data to get an approximation for the eye angle
- Will have to correct for the height of the eye
- Do area approximations with triangles, should be close enough assuming you have enough resolution
- If there is no area on one side of the pupil, find total area of eye and the percent that is whitespace

Ideas on Finding Areas

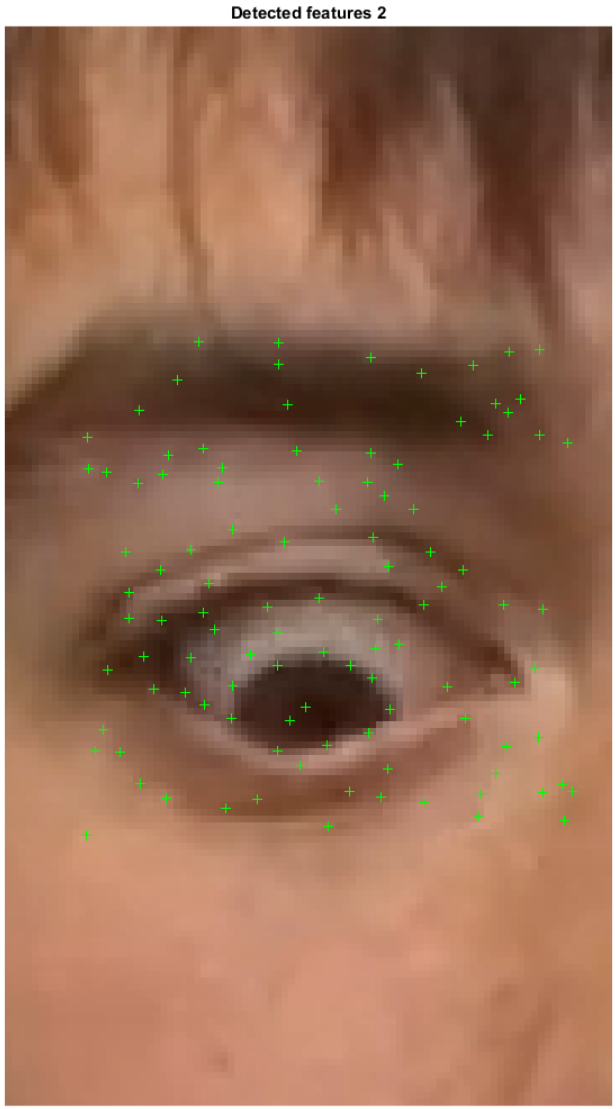
- My idea was to start at the corner of the eye
- Should be able to group points based on if they move up or down from that location
- Then find the shortest distance and go to that point to begin to define your shape
- Then use an approximation to these areas based on the triangles that are formed
- $A = \left|\frac{A_x(B_y - C_y) + B_x(C_y - A_y) + C_x(A_y - B_y)}{2}\right|$

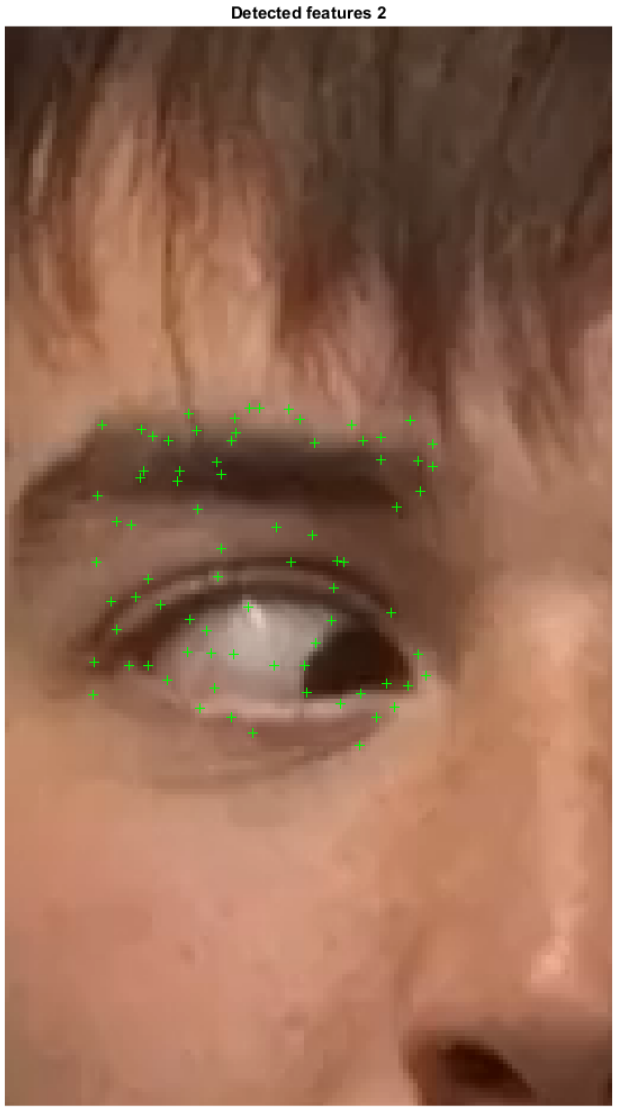
Faces and Eyes

Tracking Faces and Eyes

- Now I wanted to try doing this in a script
- Unfortunately, this is where the project ended
- I couldn’t figure out a way to identify which features where on the eye
- The resolution also started to become a problem, and this was with a 1080p camera
- This is why I believe electronically tilted LIDARs would be great because you could get higher resolution on just a few things

Illustration of the Problem





Concluding

Conclusions

- Facial Recognition is common enough now to where this could work really well within this project
- My basic example of eye tracking seemed to work well, but high resolution would be important for virtual reality imaging
- I need to figure out how to identify which features are part of the iris
- Future work includes training the KLT for my application and a more complex algorithm for eye tracking

Questions?

Thanks for Watching!!!

References

- Largely the wiki pages on Kanade-Lucas-Tomasi Feature Tracker, and Viola-Jones
- Also the mathworks face tracking:
- http://www.mathworks.com/help/vision/examples
- /face-detection-and-tracking-using-the-klt-algorithm.html