Samuel Tate

Assignment 1: Eye Detection Algorithm

Algorithm Description:

A standard Gaussian filter is applied to the input image before anything else is done. This is primarily done to prune out some weak edges. The image is then trimmed down, cutting 15% of the pixels from all sides. This is to reduce processing time as well as eliminate any false positives near edges since eyes are unlikely to be over there anyway. A Sobel edge filter is applied and saved for later use. The input image is converted to HSV space from RGB since I find it much more intuitive to work with HSV. A kernel whose size is based on image size proportions is dragged over the image and cuts out patches to score using an evaluation function. This is where the heart of my algorithm lies.

The evaluation function attempts to place HSV values into color bins such as black, white, red, green, blue etc... It also examines the number of edge pixels in the cropped image. All of these values are normalized for fair comparison. The score is determined by seeing how similar this data is to data from other images of eyes I trained earlier. Some of these training images were from the validation set, and others were some similar ones I found online. I hardcoded their bin count for colors and edges since I don't think we're allowed to use .mat files. The final score is the maximum score received when compared to the training data. Lower is better.

After the kernel is finished iterating through the image, the top scores are looked at, and some spatial reasoning is applied to prune out detecting the same eye twice or identifying eye locations that make no spatial sense (too far/too close). When a reasonable match is found,

the left eye is the one with the smaller column value and the right eye is the other one. The estimate for the coordinates is just the center of the image patches chosen. If no reasonable matches are found, coordinates located in the upper middle of the image are selected as a guess for where the eyes may be.

Validation Set Results:

The Good:

Normalized Distance = .0094

Normalized Distance = .0110

The Mediocre:

Normalized Distance = .1054

Normalized Distance = .0635

The Bad:

Normalized Distance = .2545(from algorithm guessing)

Normalized Distance = .0525 (from algorithm guessing)

It seems that blue/green eyes with the iris clearly shown perform better than those with an occluded/washed out iris. Brown eyes can be a bit of a challenge too since brown hair is common and may confuse the algorithm. One flaw with my approach is estimating the size of an eye based on image proportions. If this estimate is bad, the rest of the algorithm can suffer from it. Another challenge is trying to generalize what an eye looks like based on color. Lighting can really influence how this algorithm performs. Also, many eyes look different so it's hard to find a cookie cutter color template for an eye that will work every time. If I could do this assignment again, I would probably try out more image filters before starting the algorithm just to see if I could prune out some more areas unlikely to be eyes. I'm not sure how slow this

algorithm will be compared to the rest of the class, but I hope it's not thrown out because it takes too long. To test all 10 validation images, it takes about one minute and forty seconds on my computer. Most of that time is spent on the third and seventh images which are much larger than the others.