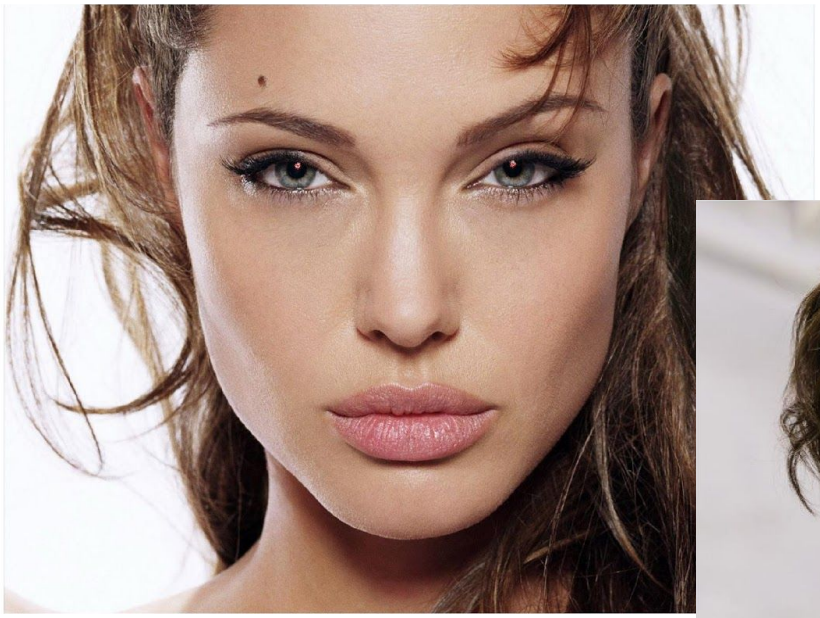
Michael Pedersen
09-06-16
Computer Vision

# Assignment 1 - Eye Detection

**Result Set:**



← Success! The red dots are right on the pupil!

Failure… The blue glasses confuse the image segmentation portion of the program.

← Limited Success: As long as the marker is on the eye I consider it a win.

**Description of the Algorithm:**

In order to process images faster I first resize every image to 1024 rows while saving the ratio between the new image and the old image resolution. The aspect ratio is maintained. My algorithm primarily relies on segmenting the eye region of interest from the rest of the image thereby reducing the detection of false positives due to noise. To achieve this I converted each image to the YCbCr and HSV color spaces. While first looking through the images I noted how each eye region had a disproportionately higher Cr (red-difference Chroma) and disproportionately lower Cb (blue-difference Chroma) compared to the skin or other regions of the images. These difference channels measure color more similarly to how the human eye perceives color. By constraining the Cr and Cb channels I was able to both remove the majority of the skin and background from most of the images.

In order to make this approach more generalizable I decided to use Otsu thresholding on the individual Cr and Cb channels. Due to the higher Cr value and lower Cb value of eye regions I can effectively use Otsu to cluster like colored pixels and discard the set which has low Cr values and high Cb values. In addition, I also used adaptive thresholding to segment even further based more on neighboring pixels than the image as a whole. By using logical and (the & symbol) I was able to merge these two techniques together in order to remove as much unnecessary information as possible. After constraining the YCbCr color space of the YCbCr converted image I am able to get a black and white image representing which regions conform to the constraints(white) and regions where at least one constraint fails (black).

Similarly, in the HSV (Hue-Saturation-Value) color space I recognized how eye color itself can only be a subset of all hues. For instance, it is probably a safe bet that not many individuals will have pure red eyes. Thus, by assuming that the vast majority of images will be for naturally colored eyes I can safely constrain the hue channel as well. Again, I generate a black and white image representing matching regions as white and non-matching regions as black.

After segmenting the image as much as I can off color I then combined the individual mask images using logical and (&). Then I clone the original color image into a new variable and convert any pixel locations where the mask is black to white in this new image. This allows me to have all the original eye information without the majority of the noise provided by non-eye regions.

Finally, to get the location of the eyes I use circular hough transform to detect possible circles (through the imfindcircles method). I then sorted these circles in descending order based off how close they are to a circular shape. Because eyes tend to be beside one another I also discard any circle that doesn't have a neighboring circle. To accomplish this I loop through each circle and find how many other detected circles are on a neighboring row (I use 30 rows centered on the circle's central point). This way I can eliminate many of the circles caused by any residual noise. To limit the guesses further I also check the region around the circle for a

high hue value. This is because the eye is typically more colorful than the rest of the image, especially after segmentation. After retrieving the circle's radius I use it to calculate the bounding box of the circle, centered on the center point of the circle. Using these bounding boxes I calculate the mean hue, from HSV, of the circle. I consider anything less than 10% to not be a good candidate. Then I simply choose the circle that survived and has the highest circular shape (based on the metric value).

**Difficulties:**

- I realized throughout development that color can be brittle - the variation between lighting conditions can be very difficult to work around. As a result I tried to mitigate this issue by liberally using morphological techniques like dilation followed by erosion (closing) to prevent accidental exclusion of eye information. I also tried to use thresholding as much as possible throughout the program in order to avoid hard coded values.

- The angle at which the subject is facing made this a much more difficult challenge. For just straight frontal portraits with good lighting I think my approach would work sufficiently but with the added variation it could quickly crumble.

- Currently there is nothing in the algorithm to detect the clustering of the candidate eye locations. As a result, an image could have the two best candidate locations be on the same eye thereby missing the other eye.

- Another limitation is the base assumption that there will only be two eyes in the photo. I have only just recognized this bias and consider it an effect of looking at the limited training set. Nothing in the instructions corroborate this bias and as a result my algorithm will most likely fail if more than two eyes are present.