Eye Tracking
SIGB Spring 2014

Marcus Gregersen
mabg@itu.dk

Martin Faartoft
mlfa@itu.dk

Mads Westi
mwek@itu.dk

26. March 2014

1 Introduction

2 Pupil Detection

In this section, we will investigate and compare different techniques for pupil detection.

2.1 Thresholding

An obvious first choice of technique, is using a simple threshold to find the pupil, then do connected component (blob) analysis, and finally fit an ellipse on the most promising blobs.

Fig 1 shows an example of an image from the 'eye1.avi' sequence and the binary image produced by, using a threshold that blacks out all pixels with intensities above 93. This manages to separate the pupil nicely from the iris.

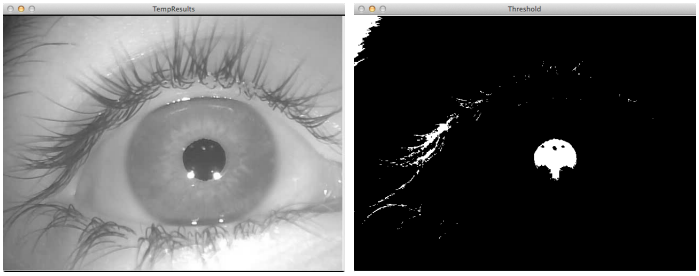


Figure 1: Thresholding eye1.avi

The next step, is to do connected component analysis, and fit an ellipsis through the blobs. As seen in fig 2, this succesfully detects the pupil, but is extremely prone to false positives.

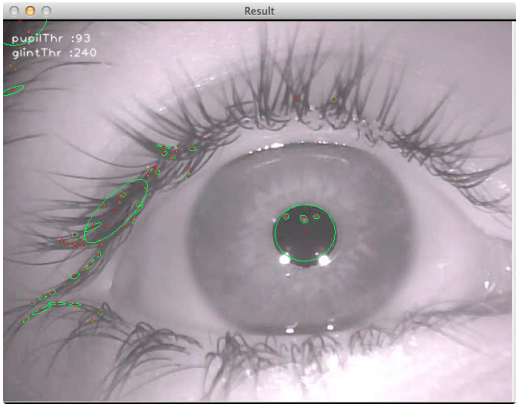


Figure 2: Fitting ellipses on blobs from eye1.avi (green figures are ellipses fitted through blobs, red dots are the centerpoint of each blob)

By experimenting, we find that requiring that the area of the blob lies in the interval $[1000 : 10000]$, and the extent between $[0.4 : 1.0]$, we eliminate most false positives on the entire eye1 sequence, while still keeping the true positive.

This approach has several problems, however. Note how the true positive on fig 2 fails to follow the bottom of pupil correctly. This is due to the glints obscuring part of the boundary between pupil and iris. It also makes some sweeping assumptions:

The pupil has size at least size 1000 If the person on the sequence leans back slightly, the pupil will shrink and we will fail to detect it.

A threshold of 93 will cleanly separate pupil from iris This is true for eye1.avi, but generalizes extremely poorly to the other sequences. If this approach is to be used across multiple sequences recorded in different lighting conditions, the threshold will have to be adjusted by hand for each one.

This problem can be mitigated somewhat with Histogram Equalization. A threshold of 25 on Histogram Equalized images, fares considerably better across several sequences. Note that this will still fail, if parts of the image are significantly darker than the pupil, thereby messing up the equalization.

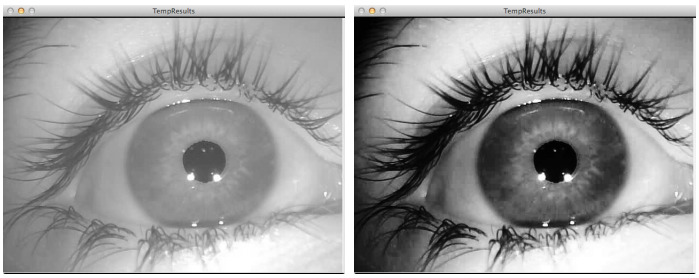


Figure 3: Eye1 before and after Histogram Equalization

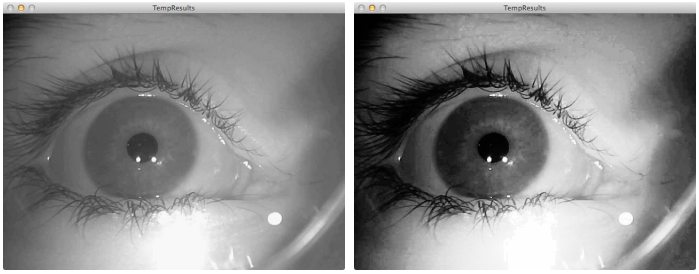


Figure 4: Eye3 before and after Histogram Equalization

Morphology Using Morphology, we can improve the detected pupil. The problem with the glints obscuring part of the boundary can be mitigated with the 'closing' operator - used to fill in holes in binary images. Fig 5 shows binary images before and after applying the closing operator. Notice how the noise inside the pupil is completely removed, and the glints are mostly removed. A downside to using the closing operation, is that adjacent, sparse structures may merge and resemble circles.

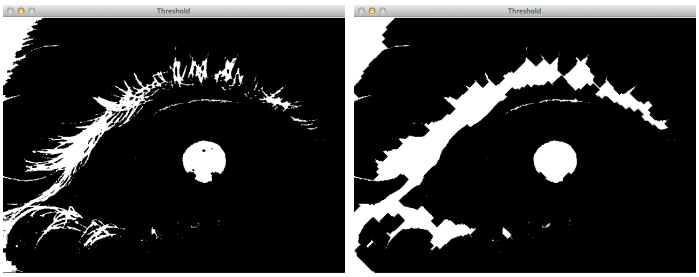


Figure 5: Eye1 before and after Closing (5 iterations, 5x5 CROSS structuring element)

Tracking The pupil tracker can be further improved, by using information about the pupil positions from the previous frame. We do it as follows:

1. Search within some threshold distance from each pupil in previous frame

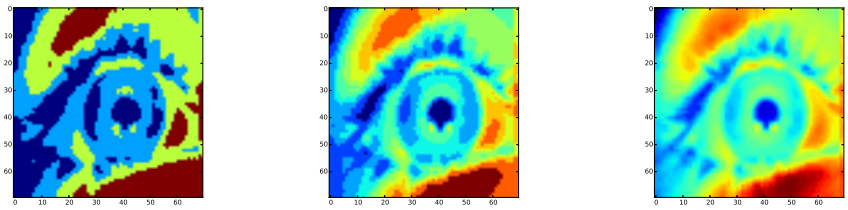
2. One or more pupils were found within the distance, return those

3. No pupils were found within the distance, search the entire image

Because of the fallback clause in '3', it is very unlikely that the true positive is not detected in each frame. The only case I can think of where this approach fails, is if the pupil is obscured for a frame (subject blinking for example), while a false positive is still detected. In that case, the pupil will improperly detected for as long as the false positive continues to be present.

2.2 Pupil Detection using k-means

A method to enhance the BLOB detection of pupil detection is k-means clustering. The method separates the picture in K clusters. Each cluster is a set of pixels, which have values closer to the cluster center, than to other cluster centres - a cluster center corresponds to mean value of the pixels in cluster. The value of K is arbitrarily chosen, so that for a sufficiently large number of K the pupil is evaluated as a single separate cluster. If the pupil is a single cluster a binary image can easily be created and BLOB detection would only need to look at the one object. The following figure illustrates how different values of K impacts the segmentation.

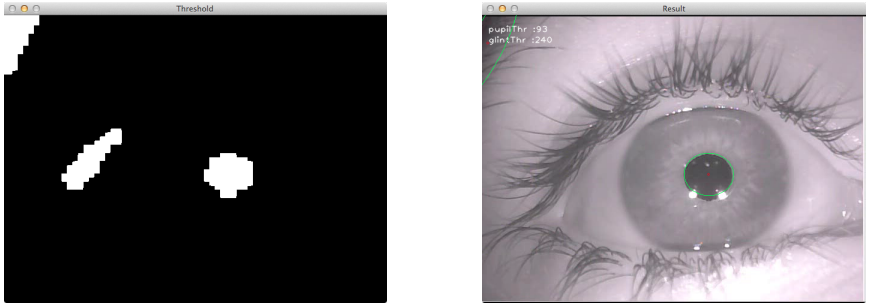


(a) $K = 4$ (b) $K = 10$ (c) $K = 80$

Figure 6: K-means for different values of K

For performance reasons, k-mean can not be calculated from the original image, the clustering is therefore done on a resized 70x70 pixel image. To reduce the impact of noise, the resized image is filtered with a gaussian filter. It is clear in figure 6 that even for a very high K, the pupil is not a separate cluster, Experiments has revealed that at a K value in the range of 10 to 20 gives a reasonably clustered image, while not having a massive impact on performance.

Each pixel in the reduced picture is assigned to a cluster(label), selecting the pixels in the label with the lowest mean value, we can create a binary image, which now can be resized to the original image size. Because the cluster also contains pixels from other regions than the pupil area, the resulting binary image, on which we can do BLOB detection, is not optimal. Figure 7 shows this.



(a) Binary image (b) Resulting image

Figure 7: Pupil detection using k-means

Eyelashes and shadows become a part of the pupil cluster, and much like the issues with morphology, they often become connected components, which makes BLOB detection very difficult.

In conclusion the use of k-means did not yield a better result than ordinary thresholding, in many cases the result was actually worse, This is un-

fortunate because, if the pupil could be found as a single cluster, the amount of evaluation on the BLOB could be reduced and give better scalability on the position of the eye relative to the camera.

2.3 Pupil Detection using Gradient Magnitude

So far, we have been looking at the intensity values of the image. This has yielded reasonable approximate results, but is not as robust as we would like. In the following, we investigate what happens if we look at the change in intensity (the image gradient / first derivative), instead of the absolute intensity value at a given point. The gradients in the X and Y directions, are easily calculated with a Sobel filter. And from these, we can calculate the Gradient Magnitude as: $\sqrt{x^2 + y^2}$ (the Euclidean length of the vector x + y), and the orientation as: $arctan2(y, x)$. Fig 8 shows a subsampled cutout of the Gradient image of Eye1, featuring the pupil and glints.

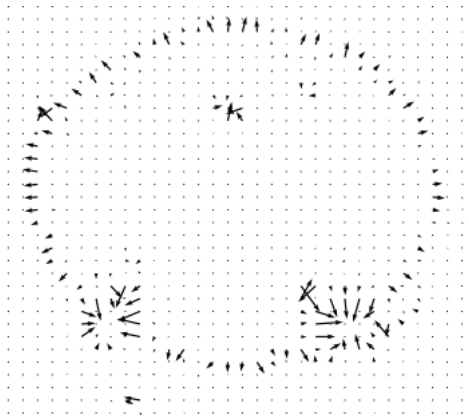


Figure 8: Quiver plot of Eye1 gradients (zoomed on pupil area)

Note that the pupil boundary is clearly visible on fig 8. We will attempt to use this information as follows: given an approximate centerpoint and radius for the pupil, scan in a number of directions, d from the centerpoint, find the location of the maximum gradient magnitudes along the line-segments that are described by the centerpoint, a direction from d and the radius. Use this set of points to fit an ellipse, and use that as improved pupil detection.

Figure 9(left) shows the lines considered, and the max gradient points found. Figure 9(right) shows the old and new pupil detections. This approach suffers the same problem as earlier. When the glints obscure part of the boundary, the pupil detection fails to follow the lower boundary. On top of that, there are also issues with noise, notice the red dots inside the top part of the pupil on fig 9, these are caused by non-system light reflecting off the pupil.

The noise issues can be drastically reduced with proper pre-processing.

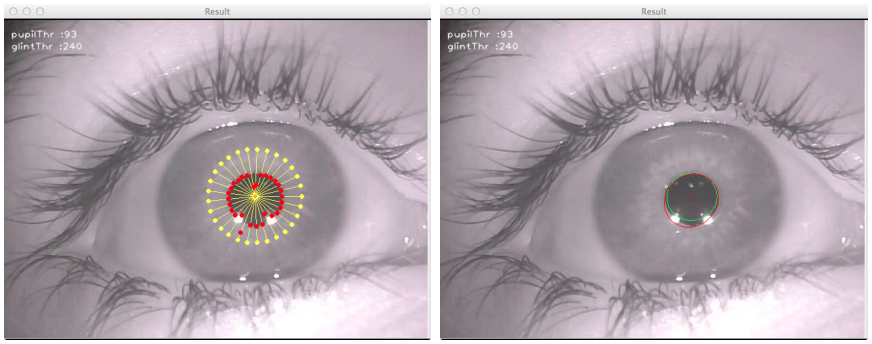


Figure 9: Left: Eye1 showing line-segments for gradient magnitude maximization (yellow) and maximum gradient values along the lines (red). Right: Eye1 showing pupil approximation (green), and new pupil detection (red)

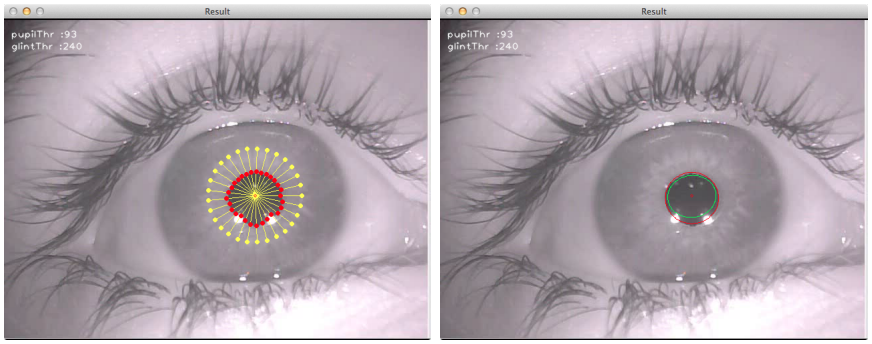


Figure 10: as fig 9, but pre-processed with a 9x9 Gaussian Blur

Fig 10 shows much improved results, when blurring the image beforehand.

We experimented with ignoring gradient points where the orientation was too far from the orientation of the circle normal, but did not see any improvements to the pupil detection.

2.4 Pupil Detection by circular Hough transformation

In an attempt to make our pupil detection more robust we now investigate the result of applying a circular hough transformation on the eye images.

The main challenge is finding the correct parameters for the process. We consider the following parameters:

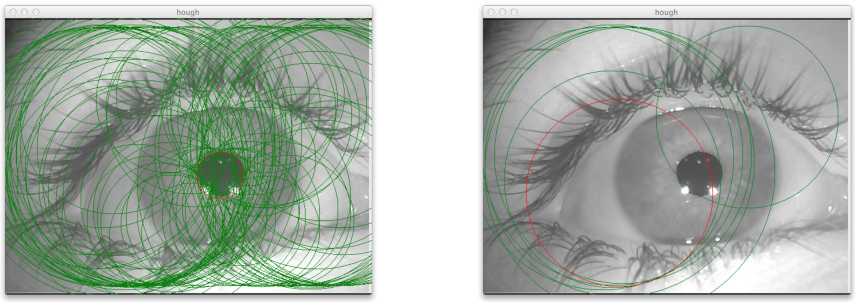
Gauss kernel size The size of the gaussian kernel that is applied to the image before the hough transformation

σ **- value** the standard deviation value used to construct the gaussian kernel.

Accumulator threshold The minimum cummulative vote thresshold in which some parameters for a circle are considered.

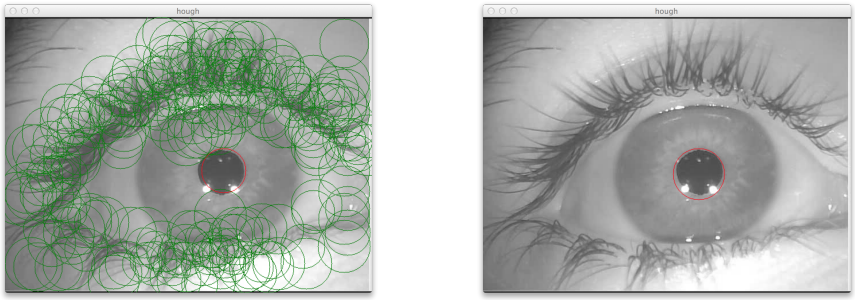
Minimum and Maxiumum Radius The minumum and maximum radius of circles to consider.

The next step is to experimentally find the parameters that yields the best result over all sequences.



(a) Accumulator threshold at 100 (b) Accumulator threshold at 150

Figure 11: Finding the accumulator threshold values



(a) Size=9, $\sigma = 9$ (b) Size=31, $\sigma = 9$

Figure 12: Preprocessing by smothing with a gaussian kernel

Discussion The circular hough transformation yields the most robust pupil detection we have been able to produce so far.

The intuition behind this is that in an ideal setting, where the eye for example is not distorted by perspective, the pupil is going to be near-circular, so the process of hough tranforming and then voting for circles is likely to succeed.

In a non ideal setting, for example in a frame were the eye is seen from the side the pupil is not going to yield the same circular properties, and the process is going to fail.

By preprocessing the image by smooting some of the noise is going to be filtered out. The idea is to choose a kernel that is rougly of the same size as the pupil. In this manner smaller features, such as eye lashes will be smothed away, but still maintaining the pupil feature.

The drawback of setting a constant size for the gaussian kernel is that is is not scale independent. An ideal kernel in some frame may smooth away the pupil in some other frame is the subject moved closer or futher away from the camera.

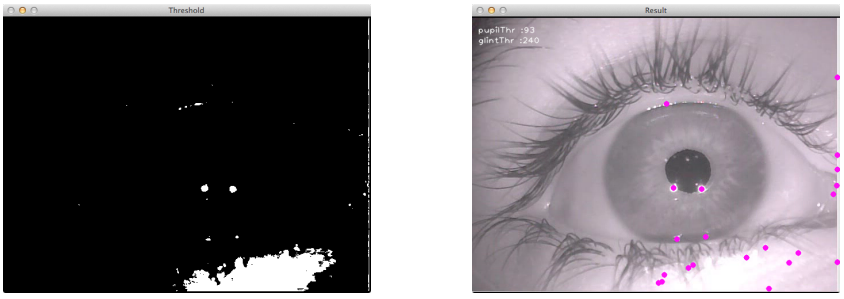
3 Glint Detection

In this section, we will investigate and discuss glint detection.

3.1 Thresholding

Like pupil detection thresholding seems like an obvious place to start. The methods are almost identical. By first creating a binary image from a threshold value, and then do a BLOB analysis, resulting in a set of points where glints are present.

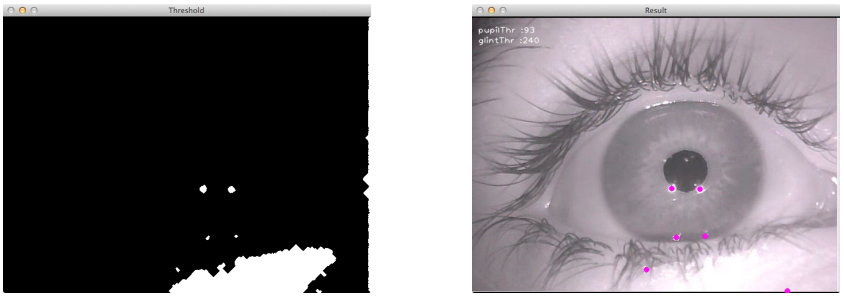
Figure 13 shows the glints in 'eye1.avi'. Because glints are very close to white, a fairly high threshold gives a good result. Furthermore by experimenting we found that, by requiring that the BLOB area lies int the interval $[10 : 150]$, we exclude a great deal of unwanted glint detections, it is although clear that there is still a good amount of false positives.



(a) Binary image (b) Resulting image

Figure 13: Glint detection with threshold of 240

Morphology To mitigate the false positives, we make use of morphology to remove small "spots" of white by first closing and then opening, we get rid of a lot of unwanted glints, as can be seen in figure 14



(a) Binary image (b) Resulting image

Figure 14: Glint detection with threshold of 240 - using morphology

Filter glints using eye features To further enhance glint detection we have added a function function that excludes glints that have an Euclidian distance from the center of the pupil greater than the largest radius of the eclipse representing the pupil. Figure 15

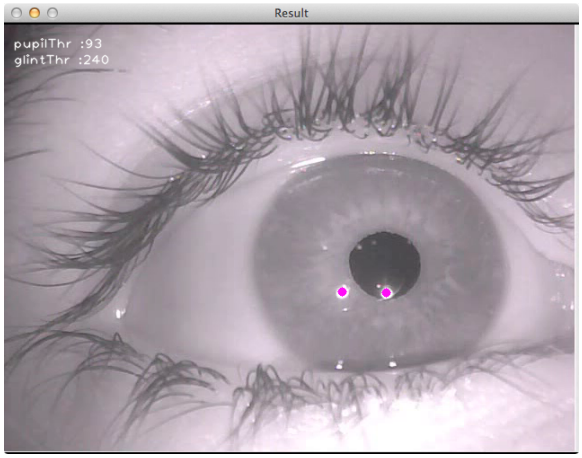


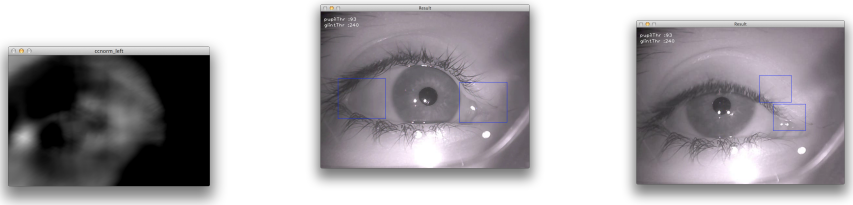
Figure 15: Glint detection with threshold of 240 - using morphology and filter

Discussion The method works very well, but fail when more than one pupil is detected, in other words the stability of glint detection is dependent the quality of the pupil detection.

4 Eye Corner Detection

We detect eye corners simply by template matching.

More intro text



(a) image

(b) A good template match of both the left and right patch

(c) A bad match of the left template

Figure 16: Template Matching

Examples The technique works best in a stable environment. Thus if the test subject moves his/her head on any of the 3 axis, or closer/farther away from the camera, the chosen template will of course produce a lower result at the position of the actual eye corner.

The changes in scale can be mitigated by using an image pyramid, i.e. convolving the template with multiple versions of a downsampled or upscaled version of the input frame.

Changes in rotation can be mitigated in a similar manner by rotating the template in a number of steps around its own axis.

Changes in perspective, i.e if the person turns his/her head, could possibly be mitigated by transforming the image by a homography.

The methods can be combined to detect a change in both scale and rotation. A problem with these techniques is that they introduce some computational complexity.

5 Iris / Limbus Detection

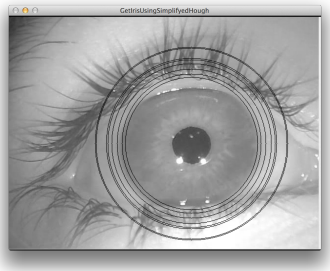
5.1 Hough Transformation with position prior

If we assume that we are somehow able to detect the pupil in a robust way, we can use this position as input to a circular hough tranformation.

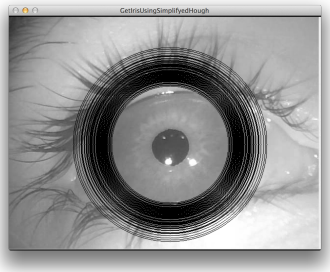
In this way the hough tranformation will be one dimensional since the only free variable is the radius of the circle.

The procedure is then to itearate over the circles in a range between some minimum and maximum radius with some step size. We then discretize the periphery of the circle and iterate over this discretization. We then increment the value of the current radius, in a parametric accumulator table, if the corresponding point in the Canny edge image is larger than zero.

The radius with the highest value in the accumulator table is thus the circle that yields the best fit.



(a) One correct circle and some false positives



(b) Multiple true positives

Figure 17: Detecting circles using hough transformation with position prior

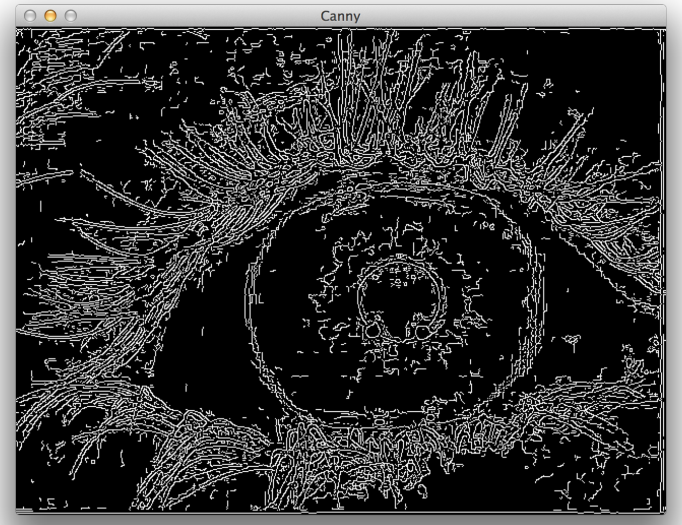


Figure 18: The canny edge image on which the circles in figure (a) are found

Examples In figure (a) the method positively identifies the limbus and some false positives.

In figure (b) the method yields more circles above some given threshold since the perspective distortion exhibits a limbus that appears to be more circular than it actually is.

Discussion The method works but is sensitive on numerous parameters. The threshold should be carefully chosen, and is dependent on the granularity of the Canny edge image it takes as input. A canny edge image with a lot

of edges yields an accumulator table with more votes, and thus more iris candidates.

Looking at Figure 18 we see that one possible improvement could be to smooth the image before producing the Canny edge image. This would remove the noise produced by eye lashes and other non-circular features.

6 Conclusion

References

[1] Foo

Appendix