Assigment 1, Graphics and Image Analysis

March 26, 2014

Alexander Kirk Jrgensen (akir@itu.dk)
Daniel Lujan Villarreal (dluj@itu.dk)
Mikkel Bybjerg Christophersen (mbyb@itu.dk)

Source code for this solution can be downloaded from:
https://github.com/DarkDecaydence/SIGB-Group67/tree/master/assignment1

Contents

1 Introduction

In this assignment we are given the task to detect various features of the eyes like the pupil, glints and iris. For this detection we will use several techniques, algorithms, and concepts we have learned in the course like thresholding, segmenting, and morphology. The assignment is divided into two parts. The outcome for the first part is to detect the center of the pupil and the glints on an video or image sequences. The outcome of the second part is to extend the previous implementation for detection of the pupil and the iris using the gradients of the edges in the sequence of images. The following sections describe the problem and solution followed by a discussion of these results and a conclusion.

2 Problem statement

The technical problem we are addressing in this assignment is the process of tracking features of a human eye in video data.
The human eye is perhaps a persons most important sensory organ, and as such identification of eye features has many applications.

- The movement of the eye, being indicative of the direction in which the person is looking, can be used in determining what a person pays attention to in a given situation.
- Some security systems rely on scanning the unique features of the eye to determine a persons identity.
- Humans look at the features and movement of each others eyes to determine the others emotional state and thought processes; it might be possible for machines to replicate this process.

In this assignment we will simply attempt to identify the position of certain eye features. We will perform no further analysis of these features, such as deriving meaning from their relative position and movement.

The features we are trying to locate are:

Pupil The pupil is the large, round black spot in the center of the eye. This is one of the eye's most distinctive features.

Glints The glints are small bright spots caused by reflection of light in the eye's surface. They are usually found close to the pupil, and those close to the pupil are the most pronounced.

Iris The iris, or limbus, is the colorful round section surrounding the pupil. It is usually partially covered by the eyelids.

Other pronounced features of the eye are the eyelids, eyelashes and eye corners. These are more complex shapes, and are more difficult to locate precisely. They will not be the focus of this assignment.

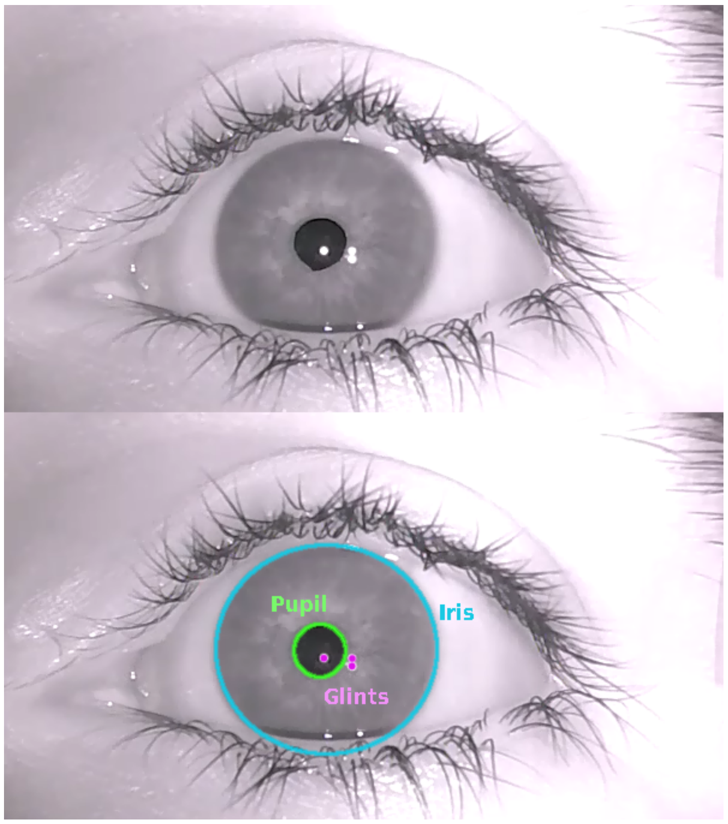

Figure 1: The eye features we wish to identify

2.1 Test Data

A set of test data have been provided to aid the assignment. These include 12 video files named from ”eye1.avi” to ”eye12.avi”, containing close-up video recordings of human eyes in a variety of conditions; brightness level, size of the eye and movement patterns vary. These are intended to offer proper challenges to our tracking solution, to ensure that it works in less-than-perfect recording conditions. A final recording named ”EyeBizaro.avi” contains one particularly difficult recording, where the eye moves around wildly under extreme lighting conditions.

The recordings are in black and white, meaning that we cannot extract color information from the videos. We cannot, for instance, filter out colorful sections in an attempt to locate the iris.

In our solution we will treat these videos only as sequences of still frames, and carry out all analysis with the assumption that we are working with still images.

When assessing the functionality of our solution we will not assume that a satisfying solution is one that can identify all relevant eye features in all recordings. We will consider the solution satisfactory if it identifies the correct features and only those in a majority of the frames of each video. Our solution will thus be focused not on demanding that we locate the correct number of each feature, but simply on reducing the number of false positive and negative detections as much as possible with the available techniques.

3 Solution

3.1 Finding the pupil with Thresholds

To find the pupil, we have expanded on the *GetPupil(...)* function.
By default, the GetPupil(...) function gathers all contours in the current image. Our assignment was to filter the contours, hopefully ending up with only the pupil.

The optimal result is to have the function return only one contour; the pupil. We found that morphing the image would give us three premises that define a pupil sufficiently accurate: The amount of points making up the contour, its size, and its extension ratio (the contour's area divided by its bounding box)

Before we applied the morphing, we had a very large number of pupil candidates, as seen in the figure below.

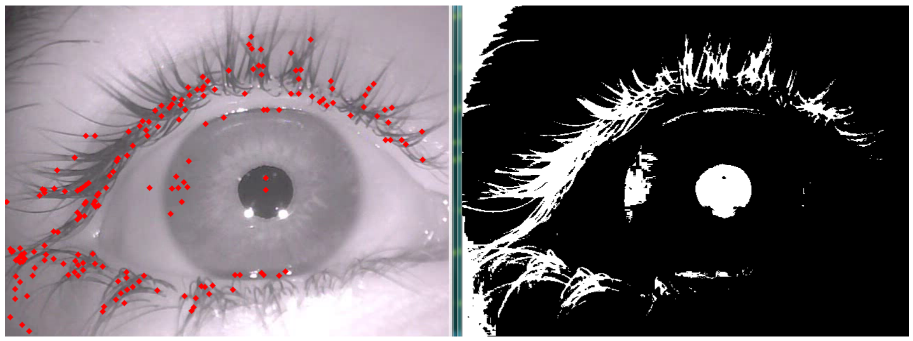

Figure 2: The binary image and pupil detection before morphology has been applied.

Since any contour is a list of points, connected by lines, we found that one can safely assume that the pupil is a contour made up of at least some amount of points. The method *cv2.fitEllipse(...)* refuses to fit an ellipse to a contour made up of less than 5 points, hence we setup a filter that disregards all such contours. Since a pupil is circular (or elliptical), a contour made from 5 points would have too few edges, and can therefore safely be disregarded as a pupil.

Our second filter checks whether the area of the contour is within the minimum and maximum size of the pupil, given by the sliders *minSize* and *maxSize*[1]. These values determine whether a specific contour has an area that is either too large, or too small. This filter is especially useful at filtering minor artifacts and noise that appears in crevices or the like. We found that a minimum of 500 and maximum of 5000 gave decent results for most, though not all, of the example sequences.

The third and final filter checks the contour's extension ratio. Assuming that the pupil is circular, we know that its area $A_p \approx r^2 * \pi$, and the area of its bounding box $A_b = w * h \approx (2r)^2 \quad | \quad w \approx 2r \wedge h \approx 2r$. Knowing this, we can find out what the standard extension of a circle would be:

$$x = \frac{A_p}{A_b} = \frac{r^2 * \pi}{(2r)^2} = \frac{r^2 * \pi}{4 * r^2}$$

Since the ratio $\frac{z*x}{z*y} = \frac{x}{y}$, we can declare the extension ratio to be $\frac{\pi}{4}$, or 0.785. We decide to put a lower bound on the extend equal to 0.5, though we could have put upper and lower bounds on the value closer to 0.785, to enforce closer proximity to a circle.

Alongside our filters, we reduce the number of pupil candidates drastically using morphology.

The morphing is done by dilating the binary image 10 pixels, and then eroding the image 10 pixels. This is known as an *open/close operation.* We do this to remove unnecessary detail that can obstruct our efforts to find the pupil. The open/close operation removes any minor noise and reduces fractioning of contours. It's effects can be seen in the figure below.

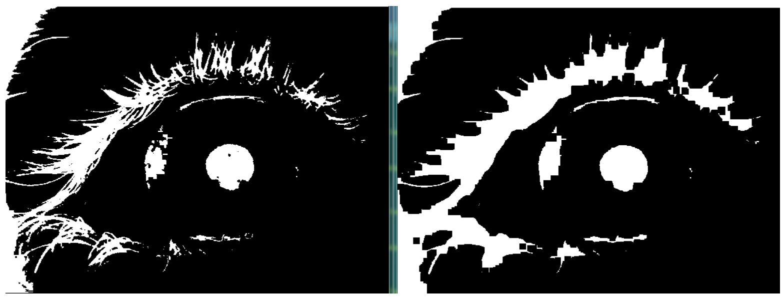

Figure 3: The effect of morphology on the binary image.

[1]minSize and maxSize are passed as arguments to the GetPupil(...) function.

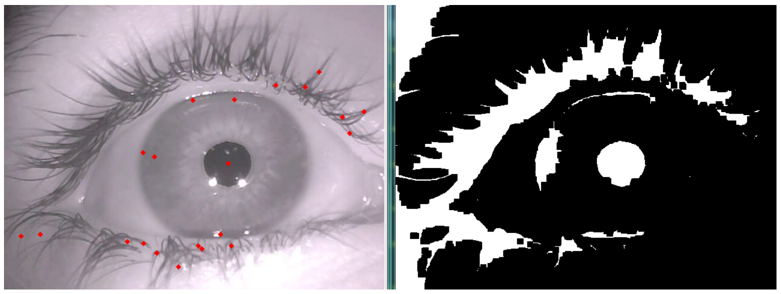

Figure 4: The effect of morphology on pupil tracking.

While our efforts does help us reduce the number of potential candidates for the pupil, it is not flawless, and still finds multiple candidates. This is likely due to our premises not being capable of isolating the pupil.

3.2 Finding the pupil with Clustering

The next step after finding the pupil and glints based on thresholds and morphology was to use clustering to segment the image into the regions of interest. For this we used the k-means method based on intensity and position as feature vectors. Playing around with the position or *distanceWeight* we found that changing this value does not give a significant improvement for the segmentation of the pupil. On the other hand, we found that a number of clusters equal to 5 gives a good separation of the pupil as one cluster for the *eye1.avi* sequence, but because of the changes in intensity the same value does not apply for all the sequences.

We were unfortunately unable to use the results of the clustering techniques to find a threshold for the pupil detection. This was due to a technical inability to implement this functionality in Python.
The concept behind the process would to pick one or more clusters with the lowest intensity, and using the value at which they separate from the remaining clusters as the threshold, in an attempt to ensure that the dark, closely collected area we are looking for (the pupil) is always put on the other side of the threshold than its surroundings.
We were unable to extract the necessary information from the clustering method, and it was decided not to spend more time on this issue in order to finish the assignment on time.

3.3 Finding the glints

To find the glints, we have applied the same techniques as used to find the pupil.

The main difference is that we have not used morphology when finding the glints, due to the fact that using an open/close operation on the binary image will always tend to remove finer detail, and when looking for the glints fine detail is exactly what we are searching for. When applying an open-operation we found that the glints tended to disappear, while a close-operation tended to fuse them together.

In addition, we use only one filter when finding the glints; an area filter. Since our pupil tracking morphology has already removed many potential candidates by removing most minor contours, we only put an upper bound on the area og the candidates.

We employ a filtering process wherein the glint candidates must be located within a certain distance of a pupil candidate in order to be considered. This allowed us to relax the constraints on the candidates a bit, as this measure proved enough to remove most false candidates by itself.

3.4 Finding the iris with blob detection

Our first approach to find the iris was to replicate the pupil detection method. By using different threshold values we tried to implement blob detection to find the center of the iris but this task was close to impossible. We found that the iris tended to either merge with other features, like eyelashes and pupil, or disintegrate into several components.

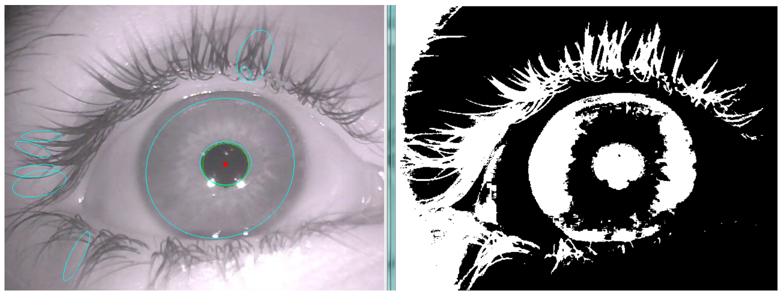

Figure 5: A rare instance where the iris is exactly identified.

3.5 Finding the iris with gradients

After the first approach failed we turned to gradients for iris detection. Boundaries in an image can be determined by a significant change of intensity between pixels. This change of intensity can be seen as a the slope, or gradient, at that point in the image. With this in mind we first calculated the gradients with respect to the x- and the y-dimensions. We then regarded these as coordinates of the gradient vector at each pixel, and used these to calculate the gradient

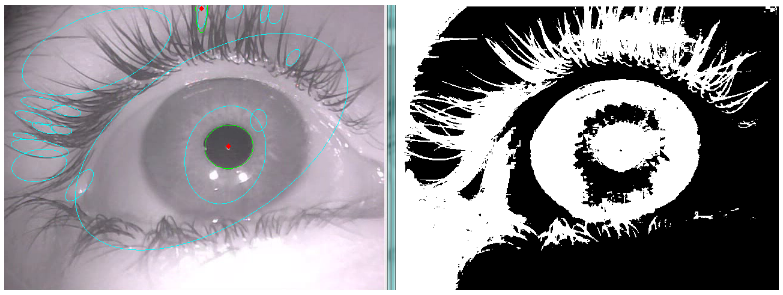
Figure 6: A case where the iris is fused with its surrounding, yet still is not filled.

magnitudes, and gradient directions. In general, the properties of these gradients where as expected. In areas where boundaries are present also grater gradients are present and their direction is towards the white area.
The edge of the iris was attempted found by finding the highest gradients along normals to a circle centered around the pupil. We fitted the length of these normals so that they generally stayed inside the eye, while not touching the pupil.
Even by trying to disregard pixels that where not aligned with the gradient curve the results where really affected by other features like the eyelashes.
The following figure shows the normals along which the highest gradient was identified. The blue lines are the normals along which gradients are found, and the blue orbs are the highest gradients along the line. The image shows that even with the entire iris boundary being located within the band searched, the edge detection still does not manage to find the boundary. The ellipse fitted to the points has little to do with the iris contour, and the edge detection generally seems happier locating the eyelashes and noisy detail in the iris than the softer boundary.

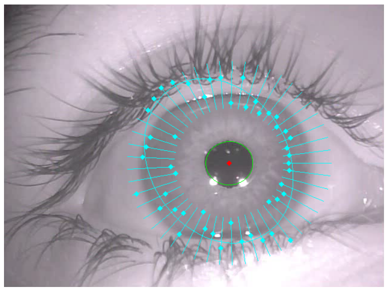
Figure 7: The gradient along the normals failing to do their job.

4 Discussion

In this chapter the results of the algorithm when applied to the dataset related to the assignment.

4.1 Pupil

The algorithm we have constructed shows very nice results when tracking the pupil. With a proper threshold defined, in almost every image in every sequence we have been given we have been able to identify the pupil and isolate it so it is the only candidate. The pupil-tracking's deficiencies are mainly concerning defining a threshold for an entire image sequence. Given that different images require different thresholds to isolate the pupil from its surroundings, it only makes sense that for a sequence containing changing conditions will not be possible to define a single threshold for the entire sequence. We have not been able to implement automatic definition of thresholds.
The following videos demonstrate the algorithms pupil tracking capabilities. For each sequence a threshold has been selected that gives optimal results. In the filtering process the minimum area for a pupil component is 500, and the maximum is 4800.
The numbers in the file names refers to the example recording processed in each video, and not to any ordering of the results.

Pupil1_good.avi	This recording shows the algorithm operate in near-perfect conditions. The pupil is large and clearly identifiable, and the algorithm finds it consistently. Very few mistakes are made, but a few components are falsely identified as pupils. Threshold is 15.
Pupil9_light.avi	Here we see how even extreme lighting conditions are analysed correctly, as long as they are consistent throughout the recording. We attribute this to the histogram equalization performed before processing; this ensures that the pupil, being the darkest object in the video, is rendered as nearly entirely black in the image being processed regardless of lighting. Here the conditions are so extreme that a very low threshold can be defined. Threshold is 3.
Pupil12_ellipsis.avi	Here we see how the ellipsis-fitting algorithm employed to simplify the representation of the pupil component manages to follow the shape of the actual pupil very exactly under good conditions. Threshold is 15.
Pupil4_negatives.avi	In this recording the pupil is not identified in certain sections (false negatives), due to it being too bright and the threshold too low to capture it. Threshold is 10
Pupil4_positives.avi	This shows the same recording as the previous, but with a slightly higher threshold. Here we see how the pupil is now being consistently tracked, but

a great number of false positives are being captured around darker areas. This demonstrates how some videos will have conditions too differing to define a single threshold for them.
Threshold is 18.

Pupil5_bright.avi In this recording the person leans forwards towards the light source, and the pupil is overly lit for a short duration. The threshold has been set so that it correctly identifies the pupil for a short duration before and after the extreme condition. The threshold is thus correct for a narrow section of lighting conditions, and the inability to locate the pupil before, during and after the leaning shows once again that changing conditions does not allow for a static threshold to be defined, even if it can for individual frames.
Threshold is 100.

Pupil8_dark.avi Here we see the opposite problem of the previous recording. Here, a large section of the recording becomes very dark towards the end, and the algorithm is unable to find the pupil. Part of the blame for this problem is on the histogram equalization process, which reduces contrast between the pupil and iris in response to large, dark areas.
Threshold is 40.

PupilBizaro_decent.avi This recording shows the "Bizaro"-video, being the hardest challenge in the provided dataset, and shows decent results in pupil tracking. The pupil is found during the average lighting conditions in the recording, but once again fails when the conditions shift.
Threshold is 5.

From these results we conclude that our algorithm can identify the pupil in almost any image, given the optimal threshold, but that changing conditions throughout a video can ensure that no single threshold works for the entire video.

4.2 Glints

The algorithm is very capable of locating glints the recordings, given that the pupil is correctly identified, which as we have demonstrated, is generally the case. The results have been similar to the pupil identification, although the glints have tended to be easier to identify under a range of conditions with the single threshold of 245. The main concern have therefore been elimination of false positives, while retaining the true glints. It has in many cases been necessary to select a proper upper bound on the glint area in the individual recordings, much in the same manner as the threshold in the pupil detection.
The following recordings demonstrate our results. The minimum distance from a pupil is set to 50 in all examples.

glint3_good.avi Here we see our algorithm working under good conditions, with equal results. We see a lot of potential false positives in the numerous bright

areas around the persons eye, but none of them are picked up on due to their high distance to the pupil.
Max area is 50.

glint4_distance.avi Here we observe false negatives when the true glints are too far removed from the pupil. Increasing the maximum distance would only cause false positives, so the problem is unsolved with our current solution. We also see how the glints are sometimes missed due to their changing size.
Max area is 50.

gline6_positives.avi Here we observe how false positives occur in the bright spots on the upper eyelid, when these move close enough to the pupil.
Max area is 40.

glint9_positives.avi In this recording we see clusters of false positives accompanying false positives in the pupil detection. This shows a weakness in relying as heavily as we do o finding the pupil in order to locate glints.
Max area is 50.

glint7_dependency.avi In this recording we see the opposite problem, where the glints are successfully found in all cases, except cases where the pupil is not identified. Under these conditions it is clear that the pupil tracking is the single point of failure for the glint tracking. This is not indicative of a robust system.

We conclude that the glint tracking works decently well, but only for cases where the pupil detection can be relied on to provide an indication of the position of the true glints.

4.3 Iris

Our iris detection is not working satisfactory. The edge detection algorithm employed does not locate the boundary of the iris sufficiently accurately to allow for proper iris detection.

iris1_failure.avi This recording shows how the boundary of the iris is not properly tracked. The ellipsis indicating our results jumps around a lot, and does not follow the iris. The only reason the iris is tracked as well as it is, is that it follows the correctly identified pupil, and that we have set the radius of the circle along which the iris contour is searched for to match the radius of the iris.

As far as we can tell the problems arise in the decision that we are identifying the point of highest gradient along the circle normal. This approach is much too prone to noise; a noisy area where 2 adjacent pixels in a monotone area differ wildly will be chosen over a smooth but consistent edge. The fact that the boundary of the iris is not a hard edge between single pixels is reflected in the results, as the algorithm shows no sign of locating the edge. The algorithm proves prone to picking up the eyelashes as iris edges. This confirms our assumption, as the eyelash area contains many harder edges, and in this case is analogous to noise.

iris1_failure no tracking follows pupil
speed

5 Conclusion

This two part assignment helped us to implement the knowledge acquired through the course to find important features of the eyes like the pupil, glints and iris. For the first part of the assignment, through morphology and physical characteristics of the pupil like area and extension ratio, we were able to find and track the pupil in most of the cases but our method still has some imperfections. For the second part of the assignment the implementation was extended for detection of the iris using the gradients in the image, although this was less than successful.