Biometrics: Iris recognition

Yauheni Zviazdou
zviazyau@fel.cvut.cz, ezvezdov22@gmail.com

January 19, 2024

1 Introduction

In this report, I will describe Iris segmentation and comparison processes. I will identify the provided images in the given database and statistically determine the threshold for iris acceptance. As a bonus task, I will use my segmentation and comparison implementation on the iris images I have.

1.1 Abbreviations

- TPR (or TAR): True Positive Rate or True Acceptance Rate.
- TNR (or TRR): True Negative Rate or True Rejection Rate.
- FPR (or FAR): False Positive Rate or False Acceptance Rate.
- FNR (or FRR): False Negative Rate or False Rejection Rate.

2 Iris segmentation

For encoding the iris code, we need to find eye bounds on the input image. The simplest way is Circle Hough Transform. But for the best performances, we should process the image. In this work, I use the operations chain:

Brightness adjustment → *Smoothing* → *Edge detection* → *Circle Hough Transform* → *Circles selection*

Provided images (Fig. **??**) are cropped on the eye, so no need to do eye detection.

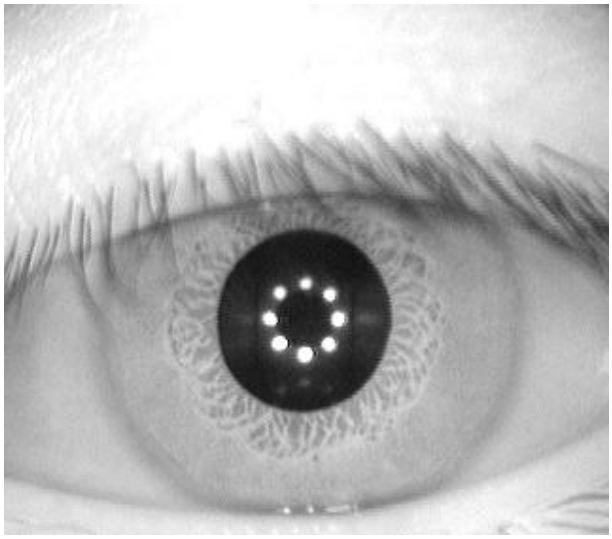

Figure 1: Example of provided image.

2.1 Brightness adjustment

Images from one camera may exhibit varying brightness levels. To standardize the images I change their brightness. I calculate image brightness as a mean of all pixel values. Then I adjust the brightness to value 50%. This normalization process ensures consistent brightness levels for all images, Fig **??**.

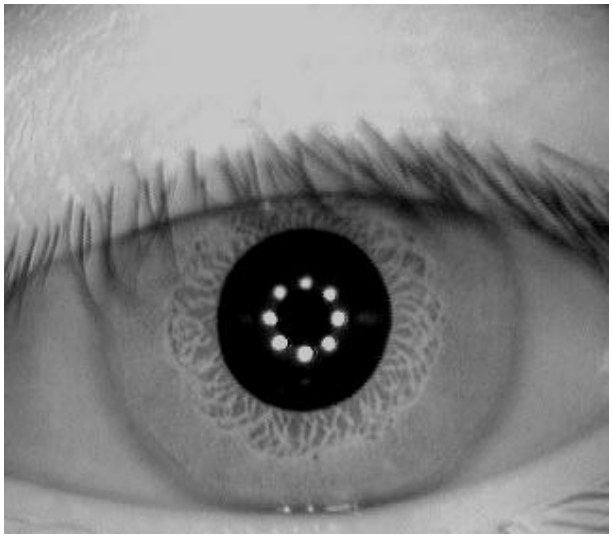

Figure 2: Image with changed Brightness

2.2 Smoothing

High-frequency objects in the image (for ex. eyelashes) can make a lot of noise and edges will be poorly detected. So to achieve better performances, it's good to smooth image. I use 12×12 Gaussian filter with standard deviation $\sigma = 15$. The result of the filter is seen in Fig. **??**.

Figure 3: Smoothed image.

2.3 Edge detection

To detect edges in the image I use a Canny edge detector with $threshold = 0.1$. The edge detector returns a binarized image with edges, Fig. **??**.

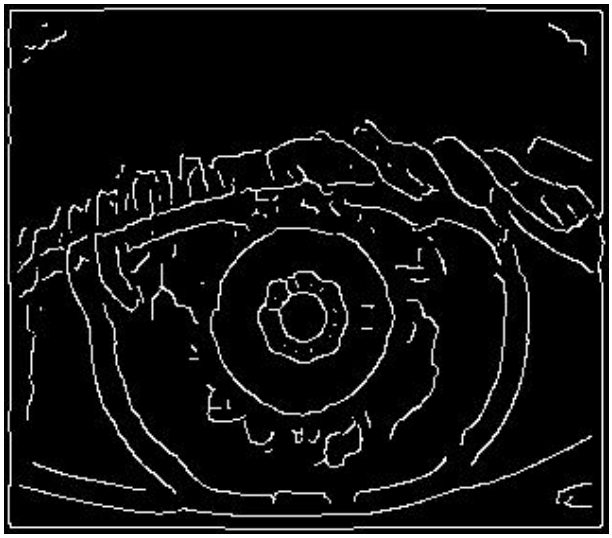

Figure 4: Edges of smoothed image.

2.4 Circle Hough Transform

Circle Hough Transform (CHT) method is based on iterating through the binarized image (where x and y are coordinates of a non-zero pixel) and changing parametric space by adding score $\frac{1}{r}$ to the all points that lie in the border of the circle with radius r and center x,y. If the radius is unknown, it should change from *min_radius* to *max_radius*. Example of parametric space, Fig. **??**.

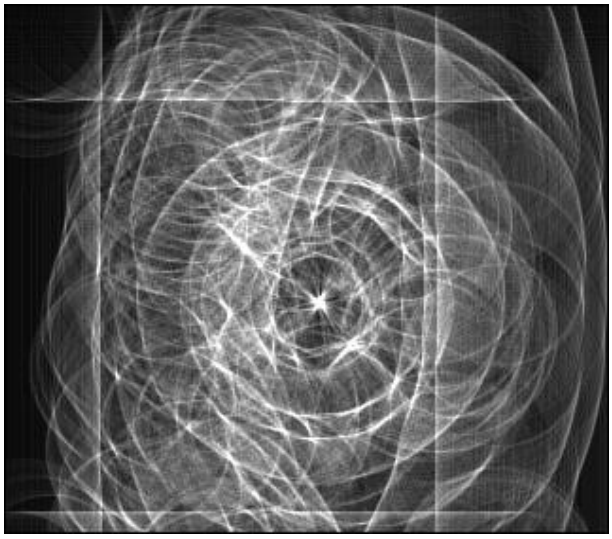

Figure 5: Example of parametric space slice with the best circle match.

My hyperparameters of CHT

- $min\_radius = 25$
- $max\_radius = \left\lfloor \frac{\min(max\_x, max\_y)}{2} \right\rfloor$
- $coordinates\_step = 1$
- $radius\_step = 1$

My implementation is not optimized, it needs 1.5 min to create an iris template.

2.5 Circles selection

The first circle is the circle with the highest rate in parametric space. The second circle is selected same way (2nd highest rate), but I made several rules to enhance the quality of detection:

- Difference between two circles should be more than *circles_radius_diff*. It helps skip circles that are almost identical to the 1st circle. I use value $circles\_radius\_diff = 40$.
- The centers of the circles can be spaced a maximum of *circles_center_diff* pixels apart on each side. This rule helps discard circles, that are made from noise. In my implementation I use $circles\_center\_diff = 10$
- Circles shouldn't cross, so I use function *circcirc* to detect crossing points of two circles. If coordinates of crossing are *NaN* then circles don't cross.

Resluts of the segmentation you ca see at figures **??**, **??**.

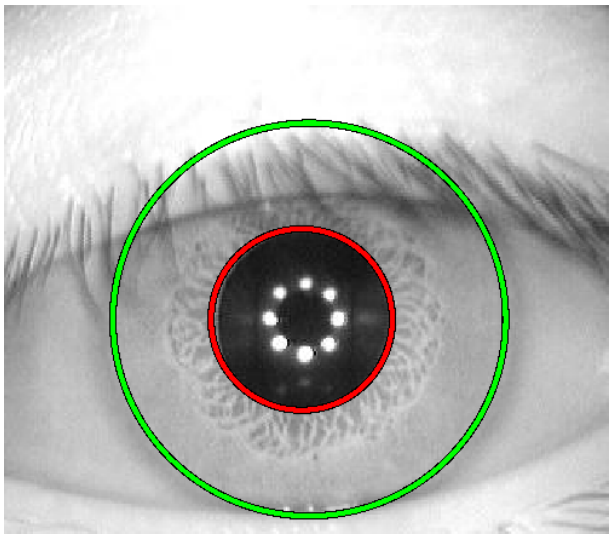

Figure 6: Segmented iris.

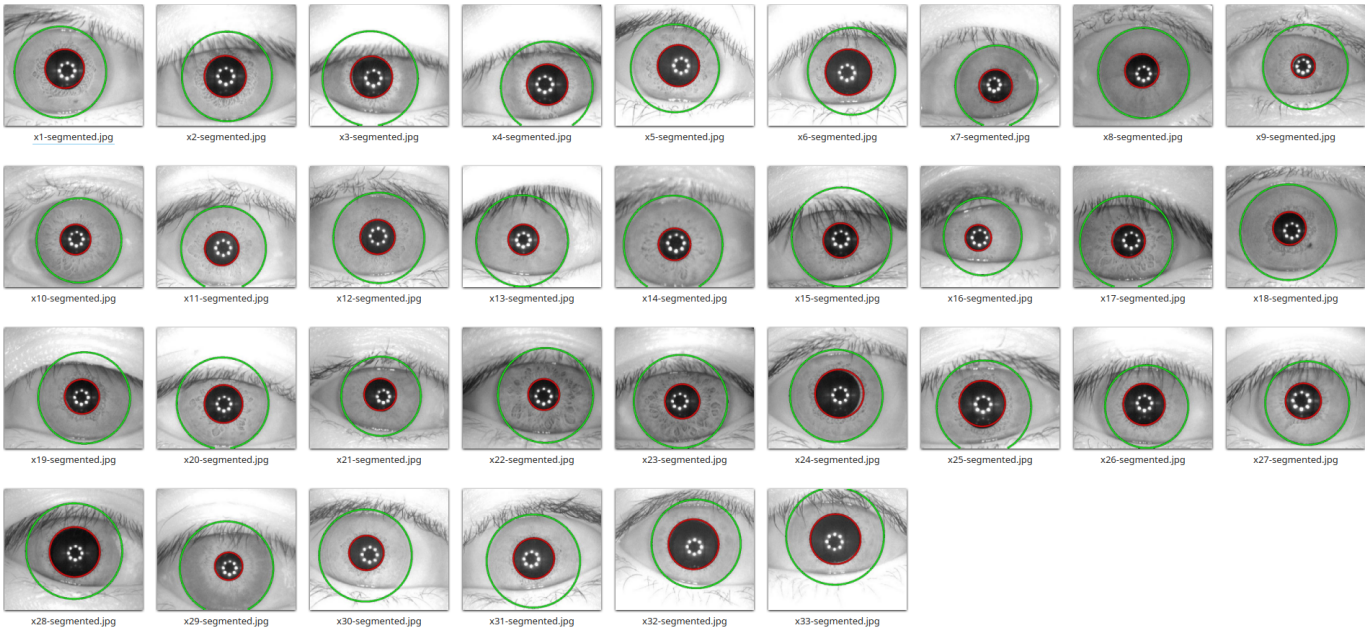

Figure 7: Provided images with iris segmentation.

2.6 Noise removing

Using the Hough Transform method is possible to detect noise in the segmentated iris, Fig. **??**.

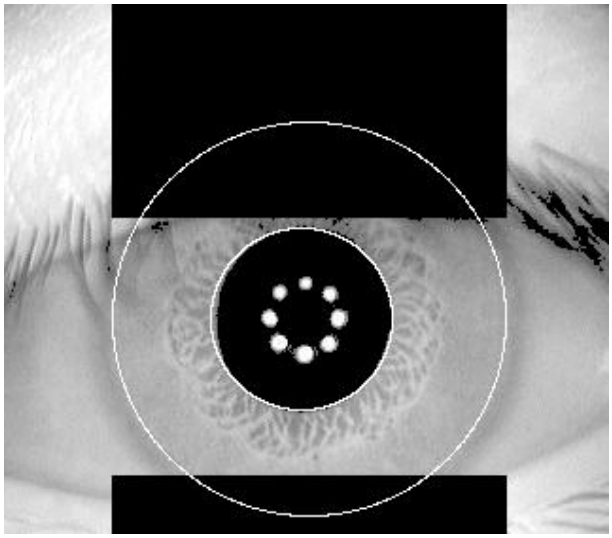

Figure 8: Removing noise from iris

3 Encoding

The next step is the normalization of the iris image (Fig. **??**), transforming into the polar coordinates (Fig. **??**) and creating a template using Gabor wavelet phase quantization (Fig. **??**).

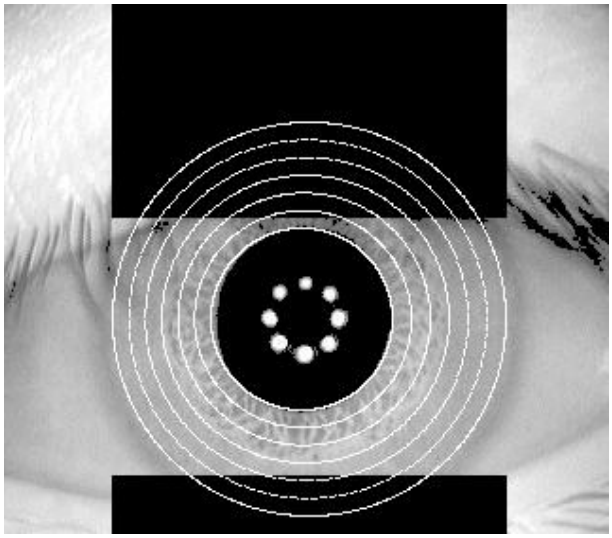

Figure 9: Iris normalization

Figure 10: Visualization of the iris in polar coordinates.

Figure 11: Iris code with noise mask.

4 Comparison

4.1 Hamming distance

Normalized Hamming Distance (HD) is used for iris comparison.

$$HD = \frac{\|(codeA \oplus codeB) \cap maskA \cap maskB|}{|maskA \cap maskB|} \tag{1}$$

4.2 Rotation

The eye can be rotated, so the iris code of the same eye can be different. For iris rotation, the *circshift* function was used. Because of phase quantization, makes sense to shift by the even numbers. In my Implementation I shift iris code by $\pm$ 12 bits with $shift\_step = 2$

4.3 Identification of provided iris images in the database

Image	Person ID	Eye side	HD
x2.jpg	1	L	0.100
x1.jpg	30	L	0.236
x3.jpg	22	L	0.318
x4.jpg	13	L	0.000
x5.jpg	1	L	0.000
x6.jpg	4	R	0.000
x7.jpg	27	L	0.222
x8.jpg	16	L	0.273
x9.jpg	13	L	0.000
x10.jpg	1	L	0.000
x11.jpg	1	L	0.000
x12.jpg	2	L	0.000
x13.jpg	29	R	0.167
x14.jpg	1	L	0.000
x15.jpg	1	L	0.363
x16.jpg	13	L	0.000
x17.jpg	4	R	0.335
x18.jpg	14	L	0.198
x19.jpg	9	R	0.223
x20.jpg	10	R	0.195
x21.jpg	13	L	0.176
x22.jpg	4	R	0.359
x23.jpg	4	R	0.332
x24.jpg	1	L	0.000
x25.jpg	29	R	0.100
x26.jpg	28	R	0.143
x27.jpg	4	R	0.000
x28.jpg	13	L	0.279
x29.jpg	28	R	0.216
x30.jpg	1	L	0.000
x31.jpg	2	R	0.000
x32.jpg	1	L	0.000
x33.jpg	7	L	0.000

Table 1: Identification of the provided images.

4.4 All-against-all comparison

I made 77562 comparisons. I didn't compare the codes that were made from the same image.

4.4.1 Distributions information

After all-against-all comparison it is possible to plot distributions on histogram, Fig. **??**. The parameters of these distributions you can see in table **??**.

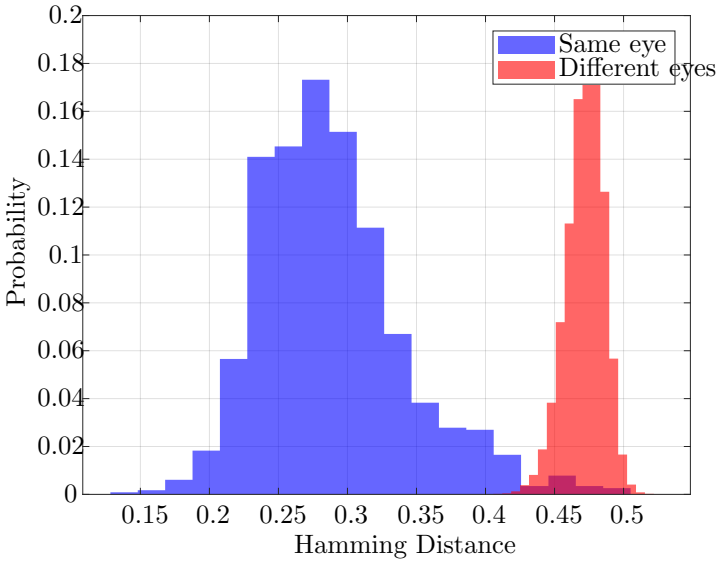

Figure 12: Distribution of Hamming Distances.

Distribution	Mean	Standard deviation
Same eye	0.2575	0.1028
Different eyes	0.4719	0.0136

Table 2: Parameter of the Normal distributions

4.4.2 Threshold selcection

I aim to create an acceptance system with the highest possible level of security. My focus is on minimizing the False Acceptance Rate (FAR) while maintaining a True Acceptance Rate (TAR) of over 95%.

- Threshold = 0.397956
- TPR = 96.34%
- FPR = 0%
- TNR = 100%
- FNR = 3.66%

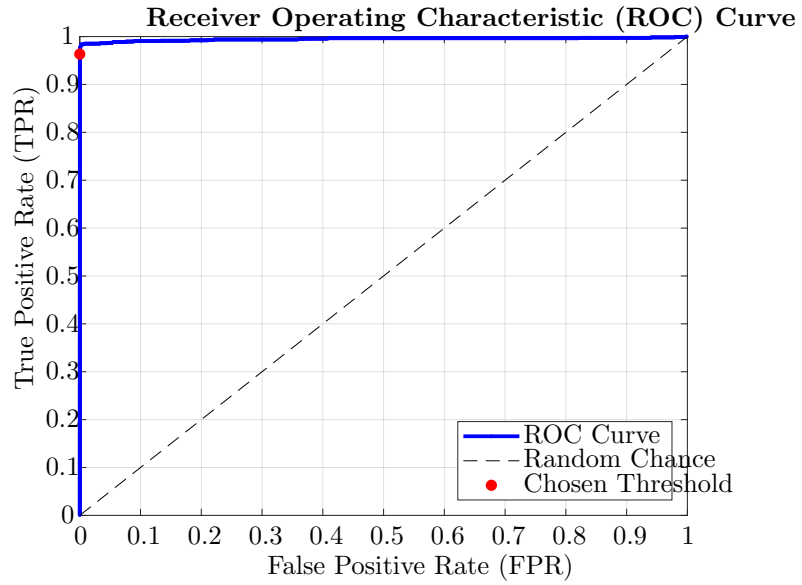

Figure 13: ROC curve of the logistic regression.

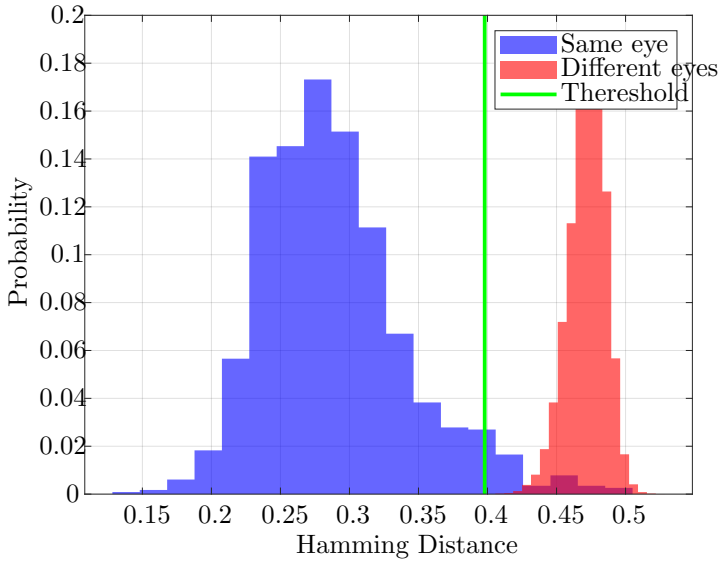

Figure 14: Distribution of Hamming Distances with the threshold.

5 System with my iris

5.1 Caption

I captured the iris of both my eyes on the VISTA FA2 camera, Fig. **??**

Figure 15: Images of my iris.

5.2 Manual eyes detction

After that I cropped these images (manual eye detection), Fig **??**. During the cropping I understood, that all cropped images should be similar, even if the eye was zoomed more than others, the system can't determine its iris bounds.

Figure 16: Cropped images of my iris.

5.3 Segmentation

After tuning parameters a little bit, I have a segmentation algorithm that works with images taken on the VISTA FA2 camera (Fig. **??**).

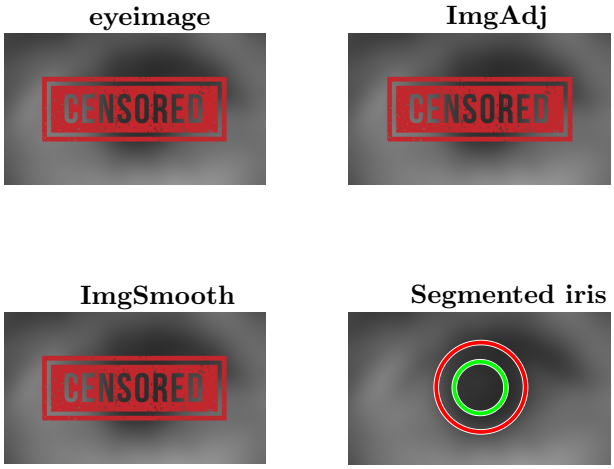

Figure 17: Segmentation process on my iris.

Changed hyperparameters:

- $irisConfig.loNoiseThreshold = 0.2$
- $gauss\_kern\_size = 15$
- $thres = 0.05$
- $min\_radius = 60$
- $circles\_center\_diff = 20$

All images was segmented perfectly, Fig. **??**.

Figure 18: Segmentation of all images.

5.4 Comparison

I have created a database containing my iris codes. Then I compared codes from database all-against-all. The results are plotted on the Fig. **??**.

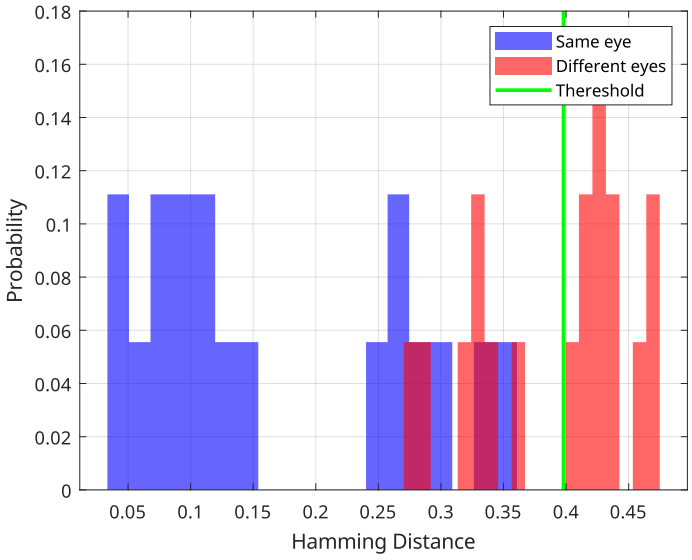

Figure 19: Distribution of my iris comparison.

The mean Hamming distance of the left and right eyes is 0.3907.

- TPR = 100%
- FPR = 38.89%
- TNR = 61.11%
- FNR = 0%

The high FPR could be because each camera has different quality levels. To tackle this, we may need to adjust the threshold for each camera individually to better match their varying qualities.