Project on Gaze Detection

Anna Braghetto - 1205200

September, 8th 2019

Introduction

The estimation of the gaze direction is carried out through a program written in C++ that uses the OpenCV library. In particular the program return as direction: **straight**, **right** and **left**. Several steps are performed in order to detect the gaze direction of the faces present in an image: the most important ones are summarised below.

1. **Face and Eye recognition**
2. **Iris detection**
3. **Eye corners detection**
4. **Gaze estimation**

A detailed description of each step is exposed in the next sections.

1 Face and Eye recognition

The image is loaded in the program as a *cv::Mat* object and then it is analysed in order to detect the faces, and for each faces, the eyes.

The recognition, for both face and eye, is performed with Haar feature-based Cascade classifiers: the library OpenCV offers many pretrained classifiers. At first, the Haar Cascade for faces is applied to the input image, returning the rectangular region in which the face is detected. In each rectangular,the Haar Cascade for eyes is carried out: for each revealed eye, the rectangular region surrounding it, is returned.

The results of the Haar Cascade application are shown in Fig. 1 for different samples.

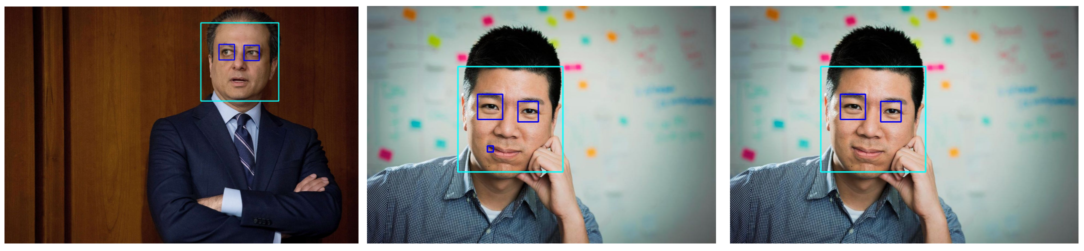

(a) (b) (c)

Figure 1: Results of Haar Cascade face and eye classifiers.

As shown in Fig. 1.b, sometimes the classifier missclassified a portion of the image: it recognizes an eye on the corner of the mouth. In order to discard regions that does not contain eyes, just the biggest ones are kept. An analysis on the dimension of each eye region is performed: the result is shown in Fig. 1.c, where just the correct detections are present.

2 Iris Detection

In order to detect the gaze direction it necessary to find the center of the eye: the iris is searched.

Each rectangular portion containing the eye is preprocessed and through the Hough Transform for circles, the circles inside the regions are found. Through a choice upon the circles, the iris is detected.

2.1 Preprocessing

The portion of the image is preprocess in different steps:

- Convertion the *Lab* color space where the histogram of L-*channel* is equalized.
- Convertion to the gray scale.
- Multiple thresholds and range resolution changes are then applied to the image.

The results obtained on the rectangular regions containg the eyes are shown in Fig. 2.1 for two sample faces.

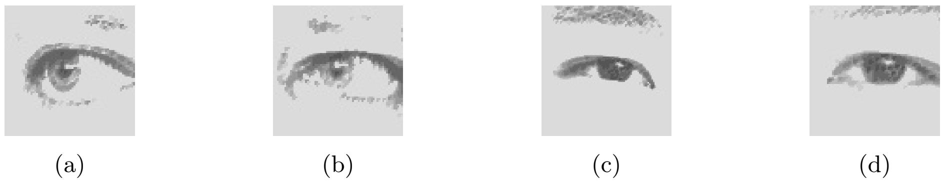


Figure 2: Results of preprocessing.

The image is ready for the Hough Trasform.

2.2 Hough Trasform

In order to get better results, before the application of the Hough Trasform to find the circles, the Canny algorithm for the edges detection is used. Then the circles are found with the function offered by OpenCV: by tuning the parameters the results change a lot. The results shown in Fig. 2.2 are obtained with parameters that work well for different input images.

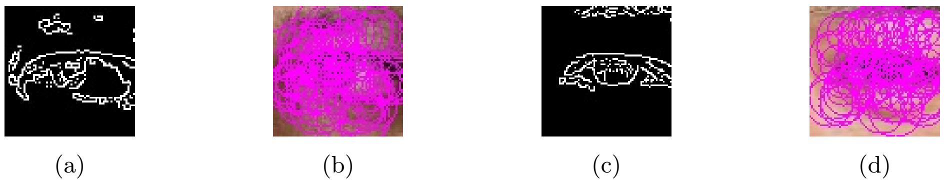


Figure 3: Results of Canny and Hough Trasform.

To find the iris, the best circle must be detect.

2.3 Choice on the circle

A score is assigned to each circle, based on the sum of the pixels contained within it and the frequency of points at maximum intensity: the best results is associated to the best circle. Since the iris is the darker region, without peaks at high intensity (as shown in Fig. 2.3), the algorithm works well. Both the quantity are normalized on the dimension of the image.

To determine the sum of pixels, called *area* and the frequency of the intensity, called *white*, the image is cropped inside the rectangular surrounding the circle. The *area* is computed by summing all the pixel values, then the *white* is determined by calculating the histogram and taking the value at the maxima intensities.

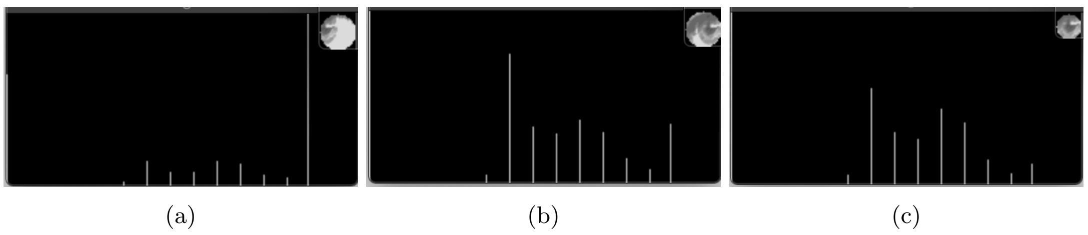


Figure 4: Crop and histograms of three samples.

An initial ranking is built by ordering the quantities (*area* and *white*) in ascending order: for each quantity a mark equivalent to its rank position is assigned to the circle. The final score of each circle is the sum of the two marks defined above. The circle with the lowest score corresponds to best tradeoff between *area* and *white*.

The results obtained for two sample image is exposed in Fig. 3.

This algorithm works well for different images of eyes and it is able to discard "false positive".

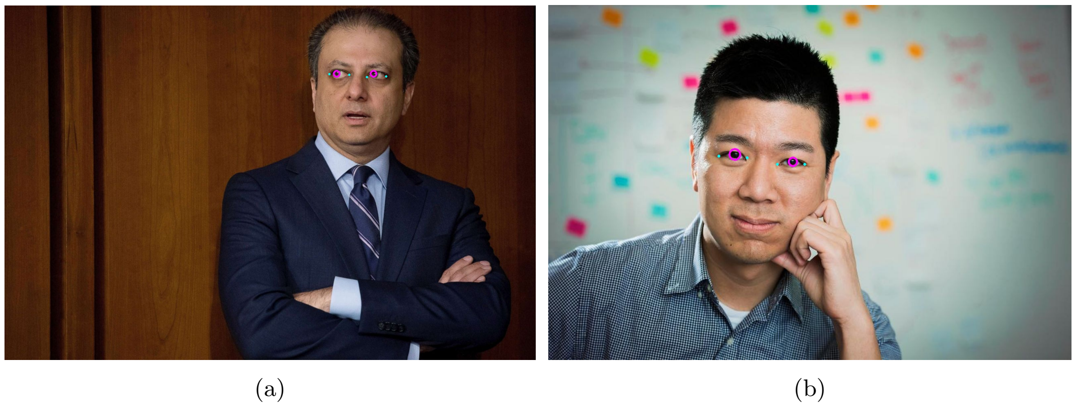
To estimate the gaze it is necessary to know the position of the eye corners.

3 Corner detection

The corners of the eye are detected by using the Harris method. Since the method returns many corners, a simple thresholding is not enough to discriminate the real corners: the best corners are those whose coordinates correspond to the outermost points.

Since on the image there are other features besides the eye, like the eyebrow, a threshold on the distance along rows (Δy) is defined. In order to avoid bad recognition in images where the head is not straight, its inclination is computed from the centers of the iris of both eyes: the image is rotated and the best corners retrieved.

The results of the corners detection is shown in Fig. 3.



(a) (b)

Figure 5: Best eye corners .

Thanks to coordinates of the corners and the center of the iris it is possible to estimate the gaze direction.

4 Gaze Estimation

The gaze direction of one eye is estimated by threholding the distance between both its corners and its center. The gaze direction of the face is the defined from the gaze direction of both eyes.

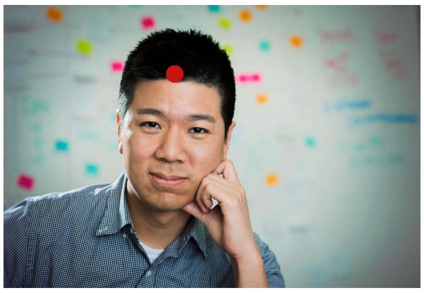
The results are dump on the screen and also plot on the image as the following legend:

1. Red circle = look straight
2. Blu arrow = look to the right
3. Green arrow = look to the left

5 Results

At the end the program shows the results at different steps while the gaze estimation on many different images are shown in Fig. 6.

(a)


(b)

(c)

(d)

(e)

(f)

(g)

(h)

Figure 6