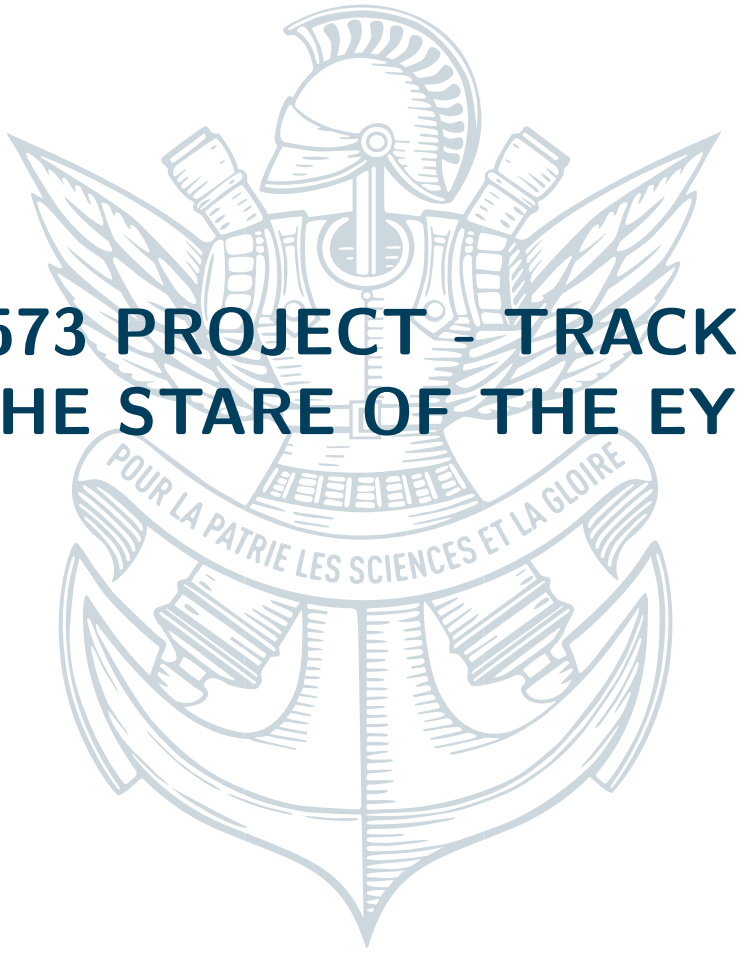

INF573 PROJECT - TRACKING THE STARE OF THE EYE

December 2022

Bohm Maximilien

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

The goal of this project was to track the stare of the eye in order to conclude whether the eye is looking at the screen or not. As a method of integrating some application, the plan was to blur the screen once the eye is looking elsewhere.

This project can be divided up in different steps, some of which turned out to be quite the challenge. The first step is to detect the face. After detecting the face, we need to target the eyes. Once all this is done, we have to somehow find a way to track the direction of the stare. Which revealed to be quite the challenge.

Many uses of eye tracking have been implemented however very few models have been developped to estimate the direction of the stare. Current methods are very computationally expensive.

1

DETECTING THE FACE

The first thought popping through my mind to solve this problem was to implement a Haar feature-detection algorithm to detect the face and different features however it turned out to not be the most reliable method to do so. On average, OpenCV's Haar feature-detection algorithm detects 1 false positive every $\tilde{3}0$ faces. Moreover, it was deemed not precise enough to be tangible to find the direction of the stare. Indeed, OpenCV's implementation only finds boundaries wheras other methods give specific landmarks
Therefore for the face detection I decided to use the algorithm provided by the library Dlib based on HOG with Linear SVM approach.

1.1 HOG with Linear SVM Approach

HOG otherwise known as Histogram of Oriented Gradients is a feature descriptor used widely in Object Detection. The principle is based on the idea of sweeping the picture in small local patches called cells and a histogram of the occurences of gradient orientations is performed. It is supposed in the theory that appearances of objects can be described through distributions of the gradient intensities and orientations.
Moreover, instead of comparing individuals gradients, HOG sums up gradient intensities within cells and therefore cells with more important gradients will have a higher weight.
After HOG, a classifier, Linear SVM, is used. For our case, Dlib's classifier is used. Dlib's classifier will be useful for our case as it provides a detection of 68 Landmark points of the face as shown below.

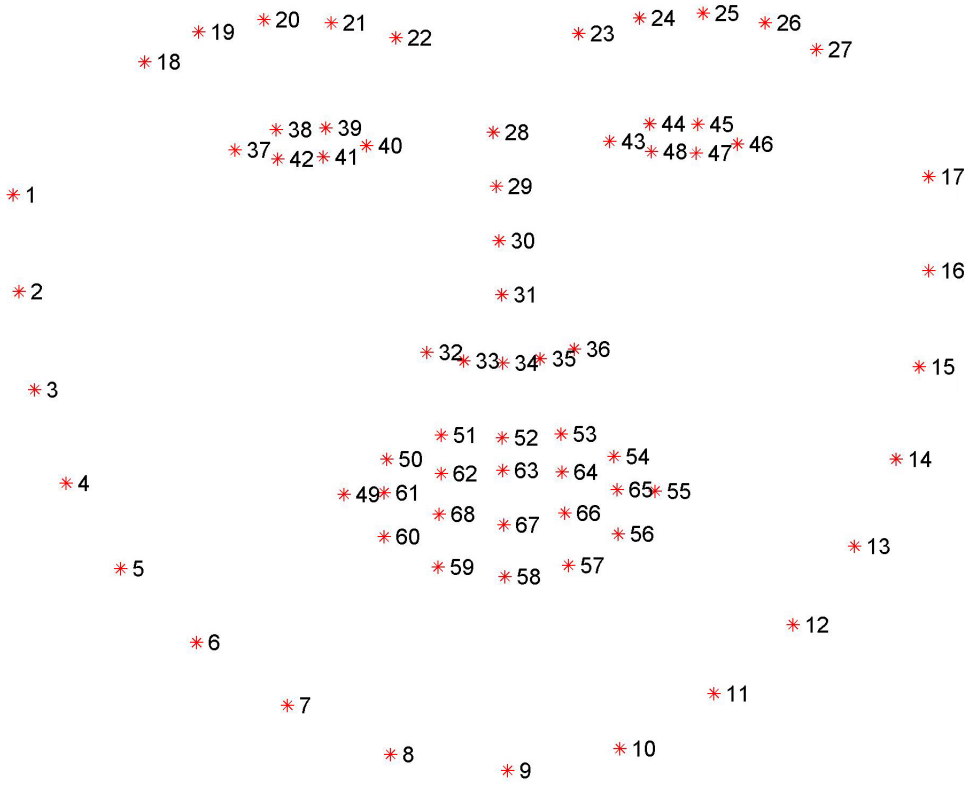

Figure 1: Overview of the 68 landmarks found in Dlib's classifier

1.2 Comparison with CNN Approach

Dlib's HOG approach is not the only one they have for face detection. Indeed, Dlib also proposes an algorithm based on a CNN Approach in order to detect faces.
It has been shown in different studies that what the CNN Approach thrives in is in detecting faces from lateral point of views.
However, in our case, if a face is sideways, there is no point in estimating the direction of the stare as you would not see the screen. Moreover, CNN is more computationally expensive compared to HOG and therefore does not bring anything interesting in our detection. Therefore we will continue with the HOG approach.

2

TARGET THE EYES

After detecting the face, now comes the time for us to detect the eyes.

2.1 Extract Eyes through Dlib Landmarks

Using Dlib's 68 Landmarks model, we extract a few specific landmarks corresponding to each eye. These landmark points include the contour of the eyes and most features in the face.
In our case the landmarks 36 to 42 for the right eye and 42 to 48 for the left eye represent the global shape of the eye.

2.2 Rudimentary way of seeing stare far off-field

From there, we extract the eyes by creating rectangles between the extreme landmarks for each eye. We then separate these rectangles in three distinct part (which will become the 'left', 'center' and 'right' stare). From these three distinct parts, we count the number black pixels in each part. We then take the one that has most black pixel and classify the direction of the stare as the being the region in which the eye is. This method is very rudimentary and simple however it is very computationally simple. It will only be able to glance at 'extreme' stare in the left and right direction however, it is still efficient.

A fun comment I will add is one challenge I had with this method. This method does require me to be in a room with lots of ambient light. Indeed, the brightness from my screen does reflect in the white of my eye and might mess up the pixel count. Hence ambient light needs to be there to counteract the screen's brightness. Therefore, my original idea to change the color of the screen depending on the direction of the stare was a miss as blue and red reflect strongly in the white part of the eye.

3

TRACK THE STARE OF THE EYE - [UNFINISHED]

Unfortunately due to some difficulties I did not manage to get this far. However I would like to share the idea I had on how I would implement tracking the stare of the eye.

First of let us find a better way to track the pupil of the eye.

3.1 Locate the pupil

Supposing that the pupil of the eye of dark color is surrounded by a bright background, we could use gradient direction at the boundary of this contrast to track the center of the pupil.
I believe some modified version of this formula could be used as an objective function in order to find the center,

$$(x^*, y^*) = \text{argmax}_{(x,y)} \left\{ \frac{1}{N} \sum_{i=1}^{N} w_i \left(\frac{(x_i, y_i) - (x, y)}{||(x_i, y_i) - (x, y)||_2} \phi(x, y) \right)^2 \right\} \tag{1}$$

where (x_i, y_i) are the boundary pixels andϕ would be some normalizing function that depends on the gradient at that point in order to disregard small gradient shifts.
Intuitinally, this formula seeks points where the direction from the point to the boundary points ressemblances the gradients.
This equation could then be used once having computed the gradient image of the cropped eye (maybe using Sobel filter), to localize the pupil center. This method is found in the works of Fabian Timm[1] however their method is more complex.
Indeed a problem that may occur using this method, would be in the case where the eye is half closed and therefore the pupil is cut horizontally down the middle and gradient orientation would not be of much help. Therefore in their paper, weights are calculated by using an inverted Gaussian filtered image.

Unfortunately, this method is great if the head is still and straight towards the camera however that is almost never the case. Therefore, calibrating the camera and geometrically estimating the head pose is very important.

3.2 Head pose estimation

Head pose estimation consists of two things: a position and a rotation.

My idea for solving this problem would be to use some of Dlib's landmarks, one of which is corresponding to the nose to locate some fixed structures of the face. Then through some calibration, we could estimate the distance between specific features like lips, eyes, contour etc... After such calibration, a rotation could be found by seeing what rotation would bring the distances found at time t with the ones measured during the calibration.

4

CONCLUSION

To conclude, this project did bring many challenges that unfortunately I did not manage to solve at this time. However, where challenges arise creative solutions follows. These papers on the methods for pupil location and head pose estimation bring a lot to the table and seem very interesting.

I had a lot of fun working on this project; it was very interesting getting to do some research on this topic and I wish to come back at a later point on this and try to fix this project to get a fully functional detection. Thank you for your time!

REFERENCES

[1] Fabian Timm and Erhardt Barth. Accurate eye center localisation by means of gradients. 2011.