Project Image Analysis and Computer Vision

Cortese Pasquale Enrico (903147)

pasqualeenrico.cortese@mail.polimi.it

Politecnico di Milano

A.Y. 2019/2020

February 11, 2020

Sommario

1 Problem Formulation and Introduction

Eye movements provide a rich and informative window into a person's thought and intentions. Thus, the study of eye movement may determine what people are thinking based on where they are looking. Eye tracking is the measurement of eye movement/activity and gaze (point of regard) tracking is the analysis of eye tracking data with respect to the head/visual scene.

Among various possible applications of gaze tracking system, Human-Computer Interaction (HCI) is one of the most promising ends: eye movements are the only means of communication for some severely disabled people.

The goal of this project was to show how gaze tracking can provide assistance in the act of reading, detecting the instant when the reader has reached the End-of-Page in an electronic book in order to automatically turn the page. To do so, it was needed to overcome some challenges, like detection of face and eye features in the videoframe, their tracking in real time and the estimation of the point of gaze from the features.

"*eyeTracking&gazeEstimation*" is a Matlab program that localizes the eye centers and the other selected features and plots the estimated gaze on the screen using a Feature-Based approach. The program is thought to work in real-time with a single low-cost camera, such those you can find on PCs, tablets and smartphones.

2 State of the art

Several approaches and techniques have been proposed in the literature. For a complete overview of the state of the art please refer to the paper “A Survey on Eye-Gaze Tracking Techniques” [1].

3 Solution approach and code flowchart

The goal of this section is to show how the problem was addressed and in which steps the proposed solution was divided; it’s important to say first that two versions of the program were created: one to gather the data and another one to do the reading test.
After the initialization of the variables and the creation of video player and camera objects, the program begins with the declaration of a while loop; the loop keeps going until the video player window is closed or the target number of frame is reached in calibration window.
At this point three main steps follow:

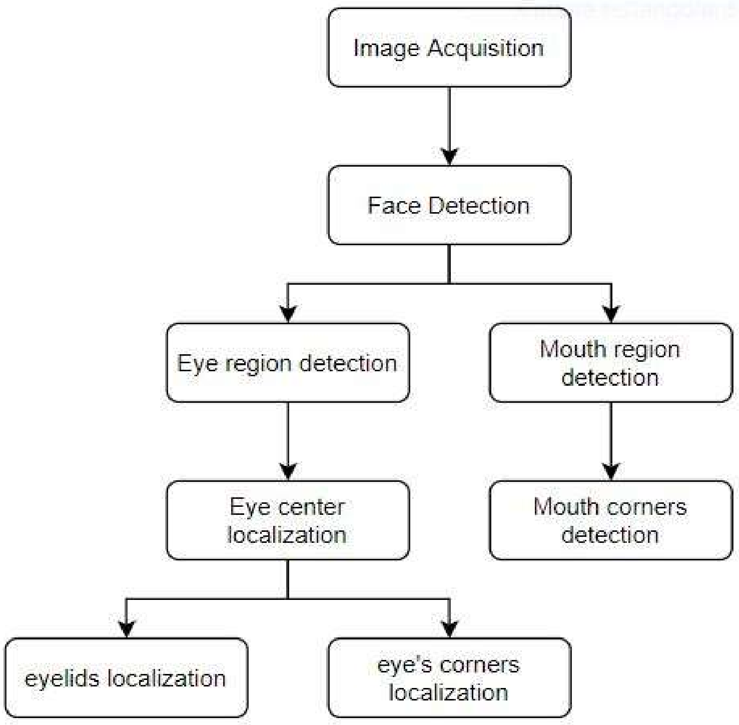


Figure 1 - Flowchart of the detection phase

3.1 Detection

As shown in Figure 1, the detection step is performed in different phases: first, the face is detected trough the Viola-Jones Algorithm [2]; the algorithm is available in Matlab as part of the Computer Vision Toolbox. The algorithm takes as input the gray-scale image where to find the face and returns a matrix of boxes containing all

the faces found; a function created by me is used to select the bigger box (that is more likely to be the right one), if no face is found the program just drops the frame and continue to the next iteration.
Once the face box is found, this is exploited to create before regions of interest where to detect, with the same routine above, the left and the right eye, and from the respective regions of interest the eye centers are localized, and after a region of interest where to find the mouth and thus the mouth's corners; eventually, from the eye centers the eyelids and the eye corners are localized.
The final result of this phase is shown in Figure 2; the algorithm for localization will be furtherly explained in the following.

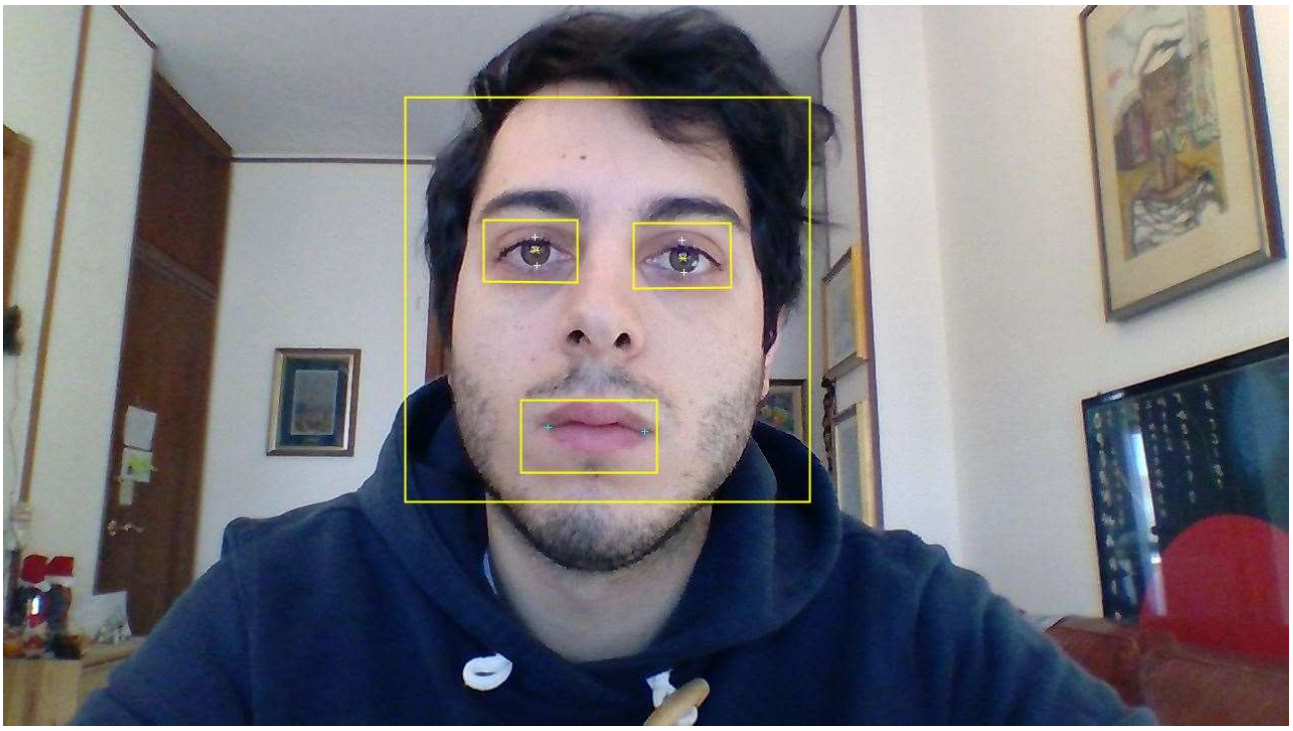


Figure 2 - Detection phase: it's possible to observe the yellow rectangles corresponding to the detection of face, eyes and mouth. In the figure are also shown the eye centers (yellow points), the eyelids (white) and the mouth corners (cyan)

3.2 Tracking

Once the desired points are detected, the two regions of interest for each eye plus the one for the mouth are used to initialize the Kanade-Lucas-Tomasi (KLT) tracking algorithm [3], using the detection features by Shi-Tomasi [13], as they are implemented in Matlab; thus five trackers are used. In Figure 3 it's possible to observe the patches tracked.
Each tracker is initialized using the entire frame image and as region of interest the ones previously found. As the point tracker algorithm progresses over time, points can be lost due to lighting variation, out of plane rotation, or articulated motion. To

solve this problem, the tracker is reinitialized when the visible points are less than the fixed threshold.
From the tracked region of interest the desired points are localized.

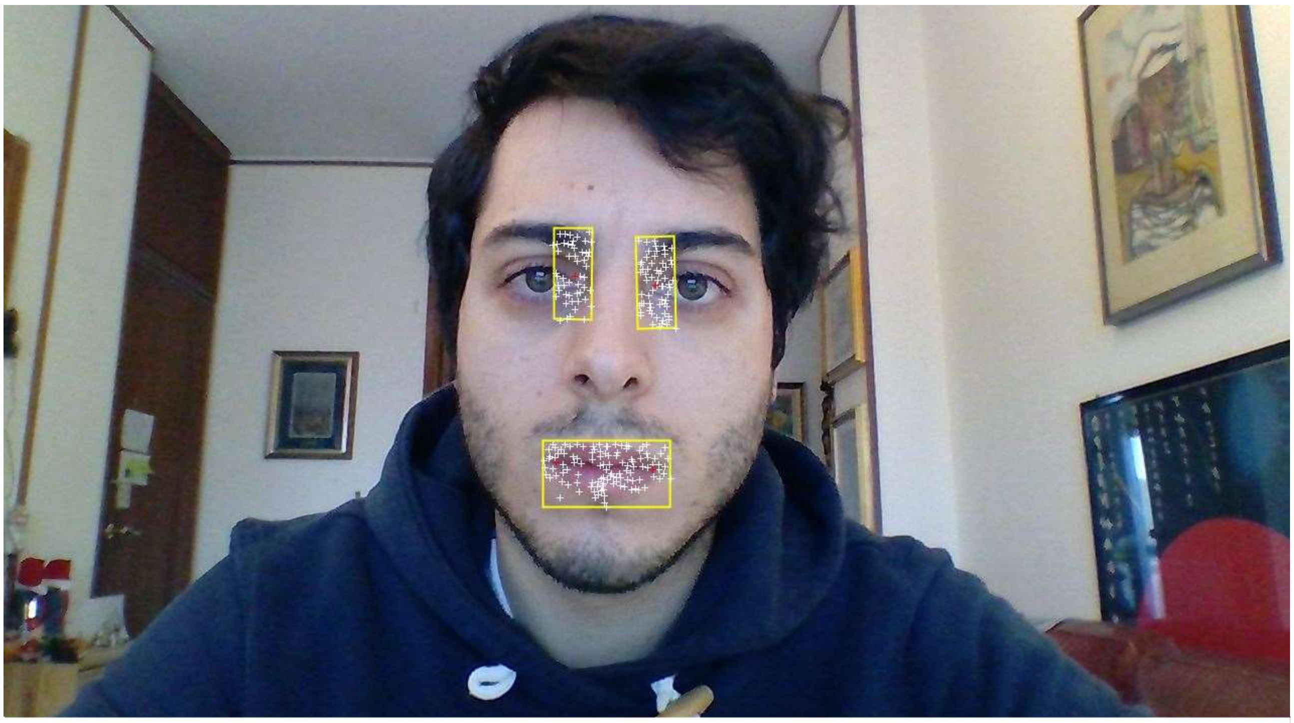


Figure 3 - LKT tracking: in yellow the patches tracked, in white the points tracked according to Shi-Tomasi [13]. It's possible to observe in red the eye corners, which are the centroids of their patches, and the mouth corners (for mor precision of the photo, the patches regarding the eyes are not showed in this figure).

3.3 Feature collecting and prediction

Once the desired points are tracked and its real time positions are obtained, the features of interest are collected and, depending of the script version, there's the prediction (in test version) of the gaze point or the packaging of the features array (in calibration version).

4 Points of interest localization

The goal of this section is to explain the localization algorithms of the points of interest in the program. These points are the eye centers, the eye corners, the eyelids and the mouth corners.

4.1 eye center localization

Given the region of interest from the Viola-Jones detection of the eyes, the localization of the eye centers is performed in a few steps. The following algorithm has been taken from [4].

First, the image cropped, correspondent to the region of interest, is converted from RGB to YCbCr color space. This latter color space is ideal to distinguish the skin area from the eye area. The eye area has a blue dominant resulting in higher values in the Cb channel. The skin area has a red dominant resulting in higher values in the Cr channel. Also, the separation of the luminance information makes the skin modeling more resilient to different lighting conditions and uneven illumination.
The $eyeMapC$ is the result of the combination of Cb and Cr channels to highlight the eye area. The $eyeMapC$ is defined as:

$$eyeMapC = \frac{1}{3}[Cb^2 + \overline{Cr} + {}^{Cb}/_{Cr}]$$

where Cb, Cr are normalized to the interval [0,1] and $\overline{Cr}$ means $(1 - Cr)$. The Cb, Cr division could produce inf or Nan pixel values. inf is saturated to the max numerical value and NaN is replaced by 0. Large values on the eye map are observed at the positions of the eye regions and eyebrows where the color difference from the skin pixels is maximized.
The irises present significantly lower brightness values than the sclera and the skin areas. So, we want to fuse the information of the $eyeMapC$ with the luminance channel. Therefore, we calculate a new eye map $eyeMapI$, that is, the division of $eyeMapC$ by Y, the luminance channel. The process is graphically represented in Figure X. The $eyeMapC$ has high values in the iris area while Y has low values in the same area, resulting in a new eye map that further enhances the iris area.
Moreover, the use of morphological operations such as dilation and erosion further accentuate the irises darker appearance in the Y component and the brighter appearance in the $eyeMapC$ component. We perform these operations with two at circular structuring elements called B1 and B2, for $eyeMapC$ and Y respectively. The radius of these circular elements is defined with respect to the iris radius.

$$EyeMapI = \frac{EyeMapC \oplus B1}{(Y \ominus B2) + \partial}$$

where $\oplus$ and $\ominus$ denote gray-scale dilation and erosion, respectively. ∂ is used to solve numerical problems deriving from a zero division. A small static number would suffice, yet, experimental tests exhibited improved results when a dynamic, data-driven value of symbol is used:

$$\partial = mean\ (Y \ \ominus \ B2)$$

The fast-radial symmetry transform (FRST) [5] is a transform that utilizes local radial symmetry to highlight points of interest within a scene. Its low computational

complexity and fast runtimes makes this method well-suited for real-time vision applications. The transform relies on a gradient-based interest operator that works by considering the contribution of each pixel to the symmetry of pixels around it. The implementation I'm using is a modified version of the Loy and Zelinski's approach [5] implemented by Kovesi [6]. First, the gradient of a gray-scale image is calculated computing derivatives in x and y via Farid and Simoncelli's 5 tap derivative filters. The results are significantly more accurate than Matlab's gradient function on edges that are at angles other than vertical or horizontal. This in turn improves gradient orientation estimation enormously. Then, we define a set of discrete radii N. The procedure is furtherly explained in [5].
The fast-radial symmetry transform algorithm is applied to the eroded luminance image $Y\ Eroded$ and to the $EyeMapI$. Then, the two transforms $S_{Y\ Eroded}$, $S_{eyeMapI}$ are summed together and the position of the maximum value pixel is the position of the eye center

$$(x_c, y_c) = argmax(S_{Y\ Eroded} + S_{eyeMapI})$$

Figure 4 reports the whole procedure.

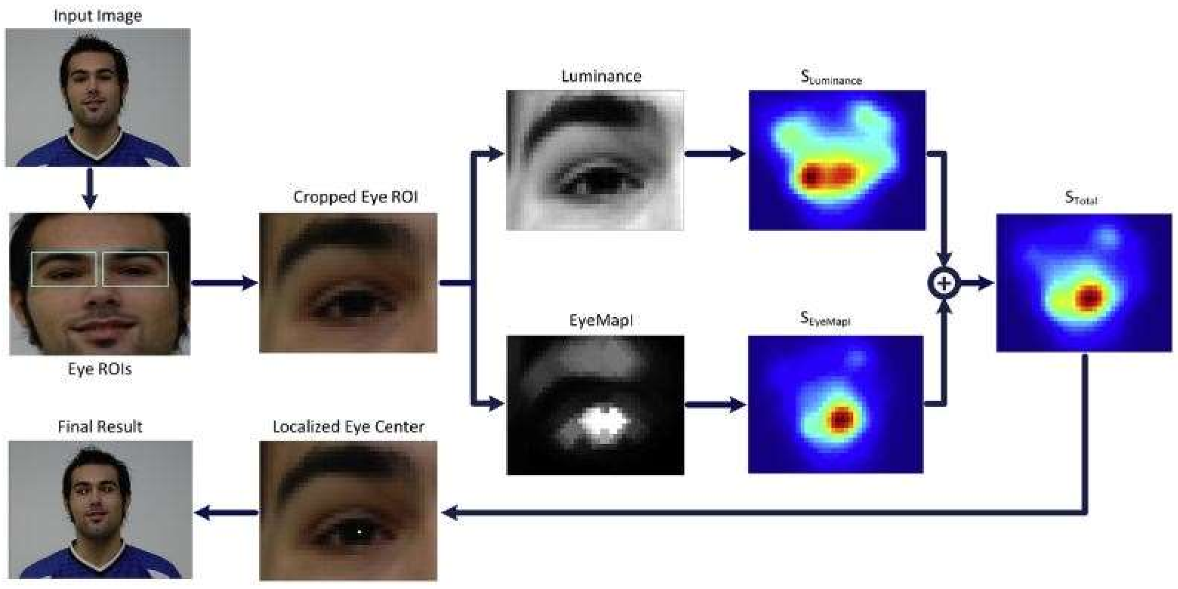


Figure 4 - Overview of the proposed system. For clarity purposes, the images showing radial symmetry transform results are pseudo-colored. Image taken from [14].

4.2 eye's corner localization

The most common approaches consider inner and outer eye points as anchor points. Instead of tracking isolated points, here is used the more efficient alternative of tracking a rectangular image patch and consider as anchor points its center coordinates. We build one patch for each eye. These two patches are located near to the inner sides of the eyes, the ones between the eye's corners and the nose. A patch contains the inner eye corner and the eyebrow edge, therefore comprising a highly textured area containing edges which are easy to track robustly. The dimensions of the patches depend on the interocular distance. The interocular distance is simply the distance between the two detected eye centers.

4.3 eyelids localization

The positions of the upper and lower eyelids provide information about the degree of eye opening [7] and greatly contribute in defining the gaze along the vertical axis. The y-positions of the eyelids correspond to the horizontal boundaries between two homogeneous areas, i.e. the iris area and skin area. The x-position of the eyelids is regarded the same as those of the corresponding eye centers. Starting from the localized eye center we define a rectangular Region of Interest (ROI) in which the eyelids are searched. The distance between the eyeball centers, also known as interocular distance, is used as the reference distance. Assuming that the iris diameter roughly corresponds to 10% of the interocular distance, the width and height of the ROI is defined as $0.1 * d_{ioc}$ and $0.3 * d_{ioc}$ correspondingly (d_{ioc} stands for the interocular distance); each vertical side is at a distance of $0.05 * d_{ioc}$ from the eye center so that only the iris area (not the sclera) is enclosed, thus constituting a homogeneous area, and the distance of each horizontal side is $0.15 * d_{ioc}$ from the eye center, so that the eyelid boundary is certainly included, regardless of the eye state. In order to detect the boundary of these distinct regions, integral projection functions are used. Image projection functions have been proven to be effective methods for extracting boundaries between different areas, representing the image by 1-dimensional orthogonal projections usually along the vertical and horizontal axes [8,9]. However, in view of the specific application, head rotations may change the boundary orientation on other directions rather that the horizontal one. To this end, the integral projection function is generalized to detect projections on different angles. Suppose $I(x, y)$ is the intensity of a pixel at the location (x, y). The integral projection along a direction ϑ for a rectangular area is defined as

$$IP(I,\rho) = \int_{-\frac{H}{2}}^{\frac{H}{2}} I(x_0 + \rho cos\theta - hsin\vartheta, y_0 + \rho sin\vartheta + hcos\vartheta)\, dh$$

where $(x_0; y_0)$ is the rectangle center (eye center), $\rho = 0,1, \ldots, W$, with W being the width of the rectangle, and H represents the height of the rectangle or, equivalently, the number of pixels to be integrated for each ρ. Given the search ROI for the eyelids, denoted here after as $I(x, y)$, we first perform an edge detection on the cropped image, through canny filter. The integral projection function of Eq. (3) is computed for $I(x, y)$ with ϑ being the inclination of the line connecting the two detected eye centers, which represents the rotation of the head.
The y-coordinate is determined finding the first value to be above a fixed threshold; the lower side position is calculated in a similar manner.

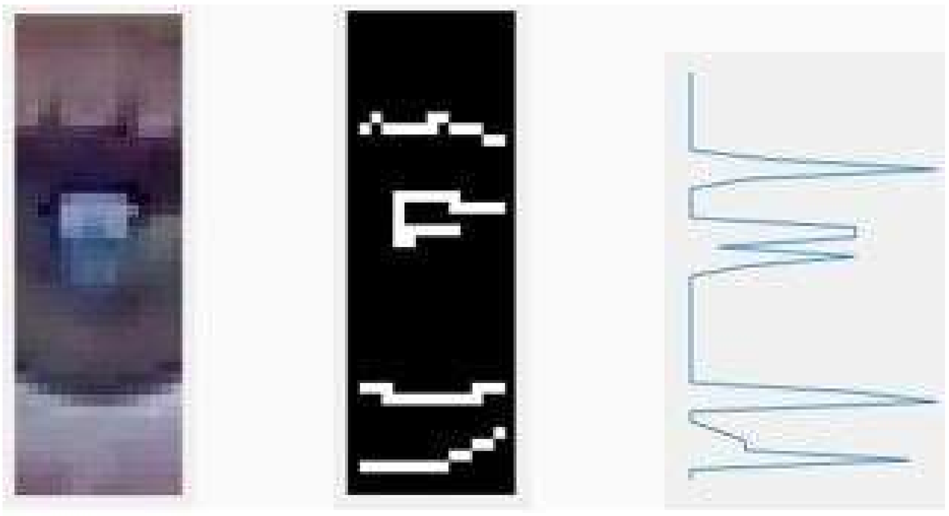


Figure 5 - Eyelid localization in different steps

4.4 mouth corners localization

The mouth corners localization algorithm was inspired by the paper [10].
To improve the speed and accuracy of mouth detection, we set the ROI based on the characteristics of the mouth distribution in the face region. Saeed and Dugelay [11] proposed that the mouth ROI was the lowest one-third of the detected face region. The lower one-third of the face image is extracted and recorded as the image I_2, and the middle half of the image I_2 is extracted and set as the mouth ROI, as shown in the green box in figure.
Based on the difference between the colors of the lips and the skin, the mouth region is precisely positioned according to lip segmentation, and we split the lips

according to the value of chromatism s of the RGB space [12]. The value of chromatism s is defined as follows:

$$s = 2 * tan^{-1}\left(\frac{R-G}{R}\right)/\pi$$

Once the chromatism value is obtained, it is thresholded and integral projection, first on y axes and then on x axes in target coordinates, is used to find the mouth corners.

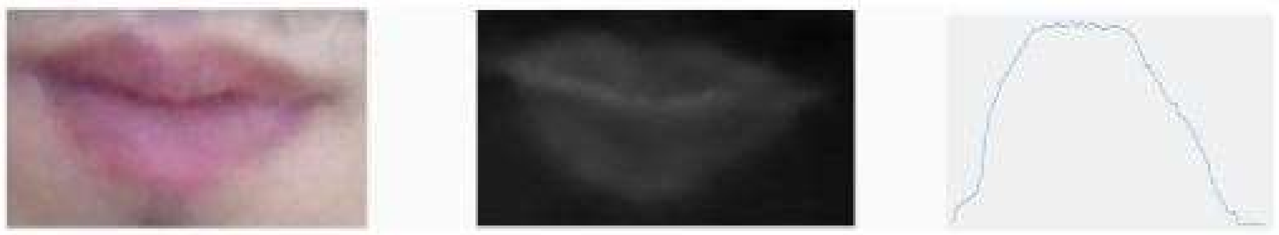


Figure 6 – Mouth's corners localization: first the Roi is cropped and the chromatism values s is obtained; then the integral projection is used to obtain first in which points s exceeds a fixed threshold and those two points are used to obtain the x coordinates, secondly integral projection is used again in the x-coordinates to obtain the relatives y-coordinates.

5 Gaze estimation

The goal of this section is to show how the prediction model is obtained, following the collection of the position of the point of interest previously described. The prediction model is the piece of code that map image data to screen positions, that is, the gaze direction. Here a linear regression model is used. Second-order polynomial equations are commonly used for 2D mapping and are generally useful to correct curved distortions and to smooth scaling along the screen. However, non-linear terms may introduce big errors especially when approaching to screen borders. As a result, errors during calibration can lead to much larger errors on screen coordinates estimations. Moreover, the more the coefficients to learn, the more the training examples should be. In order to have a fast calibration procedure and for the reasons above explained, a linear regression approach is used. Two mapping functions, one for each direction (along x and y axes), are learned. Given the assumption of independence of gaze estimation in the two axes, two separate feature vectors are formed as horizontal and vertical distances between moving and anchor points.

$$Y_h = x_0 + h_1 * x_1 + h_2 * x_2 + h_3 * x_3 + h_4 * x_4 + h_5 * x_5 + h_6 * x_6$$
$$Y_v = x_0 + v_1 * x_1 + v_2 * x_2 + v_3 * x_3 + v_4 * x_4 + v_5 * x_5 + v_6 * x_6$$

Features for both axes are calculated as the following:

- Horizontal features:

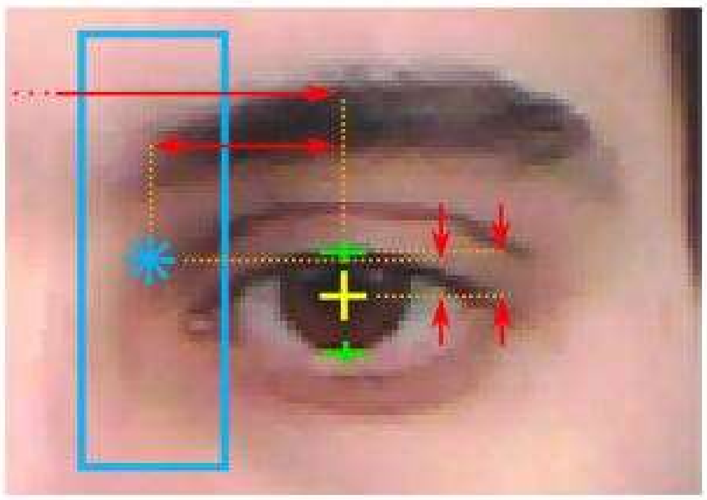


Figure 7 – Particular of the features collected, regarding the right eye. Image taken from [4]. The moving points are indicated with the yellow cross (iris center) and the green crosses (eyelids). The blue rectangle denotes the image patch which is tracked through out the image sequence and from which the anchor point (blue asterisk) is derived. The extracted features are depicted as red arrows.

- horizontal distance between left eye center and left eye corner;
- horizontal distance between right eye center and right eye corner;
- horizontal distance between right eye center and eye corner;
- horizontal distance between left eye center and left mouth corner;
- horizontal distance between right eye center and right mouth corner;
- horizontal distance between left eye center and right mouth corner;

- Vertical features:

- vertical distance between left eye center and left eye corner;

- vertical distance between right eye center and right eye corner;
- vertical distance between left anchor point and left eyelid;
- vertical distance between right anchor point and right eyelid;
- vertical distance between left mouth corner and left eye center;
- vertical distance between right mouth corner and right eye center;

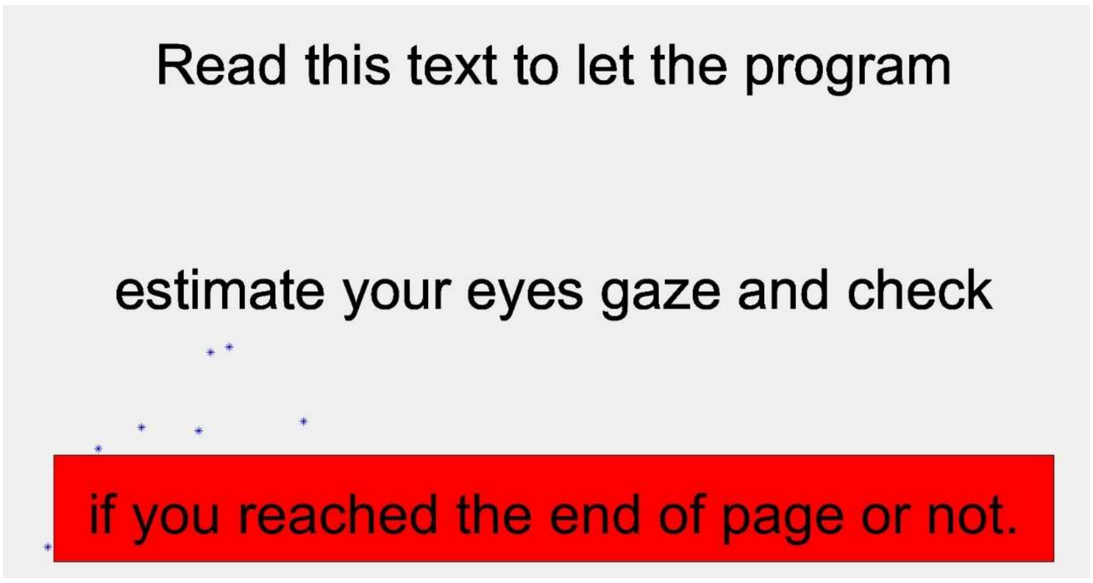
In order to achieve the final result, two phases have to be followed.

5.1 Calibration

In the calibration phase some points, randomly chosen, are showed to the user, lighting them one by one and registering the feature vectors in that known positions. After calibration I have correspondences between known points on the screen and the relative features, allowing me to train the linear regression model. This operation is performed by the script "*regression_model.m*" inside the "data" folder.

5.2 Test

As explained, with the calibration completed, the new input vector arrives, and it's mapped to the estimated screen point with the obtained model parameters. To approximate a real situation where this project can be used, a text is inserted in a figure: when the user arrives to the end of page the desired action for the specific application can be triggered, which in the test script was set to signal the end of page was reached.


Read this text to let the program

estimate your eyes gaze and check

if you reached the end of page or not.

Figure 8 – Test image with plotted predicted gaze points (in blue).

6 Discussion and future work

The proposed gaze tracking system extracts facial points in order to derive gaze-related features which are subsequently mapped into screen coordinates.
The solution used to approach the formulated problem presents both weak and strong points: the use of color information to locate the eye center is far more reliable than geometric interpolation methods, and it is proven that this method doesn't require high resolution images, a fact that can be exploited to achieve a faster execution of the program. Also, the color information is immediately robust to change of the pose, differently to geometric interpolation.
On the other hand, uneven illumination, shadows or reflections, like in the case of the user wearing eyeglasses, may trick the program, thus one first enhancement of the program could be making the program robust to this aspects.
One aspect I worked on in the making of the project was to use the detection just in the initialization step and in the re-initialization ones, lifting the weight of the program on the tracking; although, tracking is subjected by noise and, for the problematics told above, this could result in a wrong localization of the points of interest: a second improvement to the program could be the implementation of a policy about noise reducing and continuity check, calculating the distance of the same point between two consecutive frames and fixing a range for this to be kept, modified or discarded.
As told before, the program is good in tracking the points of interest during movement of the head, like translation or rotation, but the detection should be improved, giving the possibility to re-initialize the point to track when the face is not right in front of the camera.
In conclusion, the fixed goal was reached: to record the gaze point on the screen and to trigger an action in the program when the end of the page was reached.

7 Table of figures

8 References

[1] Chennamma, H. R., and Xiaohui Yuan. "A survey on eye-gaze tracking techniques." arXiv preprint arXiv:1312.6410 (2013).

[2] Jones, Michael, and Paul Viola. "Fast multi-view face detection." *Mitsubishi Electric Research Lab TR-20003-96* 3.14 (2003): 2.

[3] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674.

[4] Skodras, Evangelos, Vasileios G. Kanas, and Nikolaos Fakotakis. "On visual gaze tracking based on a single low cost camera." *Signal Processing: Image Communication* 36 (2015): 29-42.

[5] Loy, Gareth, and Alexander Zelinsky. "Fast radial symmetry for detecting points of interest." IEEE Transactions on pattern analysis and machine intelligence 25.8 (2003): 959-973.

[6] https://www.peterkovesi.com/matlabfns

[7] Bour, L. J., M. Aramideh, and B. W. Ongerboer de Visser. "Neurophysiological aspects of eye and eyelid movements during blinking in humans." Journal of Neurophysiology 83.1 (2000): 166-176.

[8] Feng, Guo-Can, and Pong Chi Yuen. "Variance projection function and its application to eye detection for human face recognition." Pattern Recognition Letters 19.9 (1998): 899-906.

[9] Zhou, Zhi-Hua, and Xin Geng. "Projection functions for eye detection." Pattern recognition 37.5 (2004): 1049-1056.

[10] Ji, Yingyu, et al. "Eye and mouth state detection algorithm based on contour feature extraction." Journal of Electronic Imaging 27.5 (2018): 051205.

[11] Saeed, Usman, and Jean-Luc Dugelay. "Combining edge detection and region segmentation for lip contour extraction." International Conference on Articulated Motion and Deformable Objects. Springer, Berlin, Heidelberg, 2010.

[12] Pan, Jing, Yepeng Guan, and Shuangcheng Wang. "A new color transformation based fast outer lip contour extraction." JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE 9.9 (2012): 2505-2514.

[13] Shi, Jianbo. "Good features to track." 1994 Proceedings of IEEE conference on computer vision and pattern recognition. IEEE, 1994.

[14] Skodras, Evangelos, and Nikos Fakotakis. "Precise localization of eye centers in low resolution color images." Image and Vision Computing 36 (2015): 51-60.