**Introduction**

We were tasked with construction of a software eye tracker. In the following text we describe our approach to this challenge, the methods we used, and the results we have obtained.  We have built the eye tracker using Python and OpenCV as a toolkit. The purpose of the eye tracker is to correctly and accurately detect the eye (pupil, iris) and glints on the eye in every image where there is an eye and detect nothing if there is no eye present in the image.

**Adding Sliders to the Interface**

To be able to have a more easy to update workspace, the re-design of the base line code was needed. By separating the current code into classes into classes to make their use more easy and accessible from around the program. Adding more sliders to be able to calculate values and get the most accurate to separate the right pupil, Iris candidates[[1]](#footnote-1).

In the current framework there is three namedWindow[[2]](#footnote-2), as the original code it has a Temporal windows, controller and original window, since its simplicity to add sliders on depends of the function being called, this creates a robust way to get input back to the program.

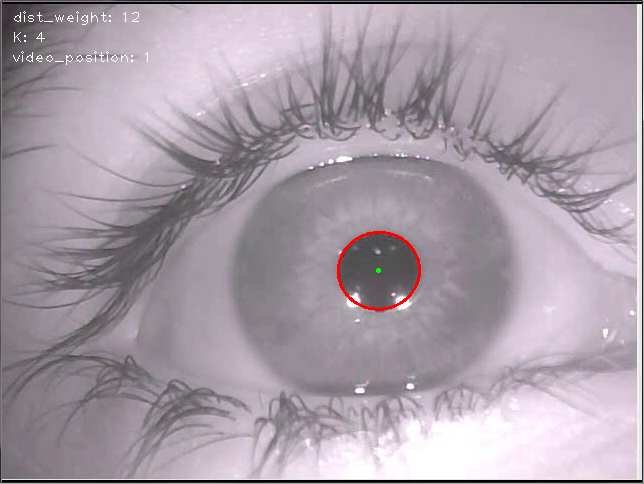
**Pupil detection**

Our approach to detect the pupil works in three stages:

1. Equalization of the histogram. This method improves the contrast in the image, in order to stretch out the intensity range.



1. The use of findContours, this open CV function retrieves the contours of a binary image.
2. Use a k-means histogram segmentation to refine the approximation of the pupil region, and find an initial approximation to the pupil center.
3. Refine the pupil center and find the elliptical outline, using the ellipse algorithm.



**Algorithms for Pupil detection.**

Our first assumption assumes that the pupil region, is either the dark

1. *Look at SGIBWindows.py for more details* [↑](#footnote-ref-1)
2. *http://docs.opencv.org/modules/highgui/doc/user\_interface.html* [↑](#footnote-ref-2)