Iris Detector

This is a program which extract the image of the iris, from a video recorded with the Raspberry camera, in real-time. The software has been developed using **Python** and the library **OpenCV**. Everything has been thought to be used with the **Raspberry pi** and the **pi camera**.

How to use it

IrisDetection.py

To execute the program, from the command line, digit:

- **python IrisDetection.py**

To use the latter, remember to put the file **functions.py**, which contain some of the functions used by the main program. It is also possible to use the following parameters:

- **-O** followed by the path in which we want to save the picture and the images of the iris. If this parameter is not specified, the default one will be used, which is: */home/pi/Desktop* and the name of the picture, with additional alphanumerical extensions.
- **-p** followed by the number *1* or *2**,*** permits to specify which function is going to be used to calculate the ray of the pupils. If *1* is chosen, it will be used the function **getPupil()***,* and if 2 is chosen, **getPupil2()** will be used. Both these functions will be explained later.
- **-e** permits to specify the type of event which will start the picture shot. If not specified, the picture will be taken when the button "*q*" is pressed. If it's specified, and it's followed by the test "*face*", the picture will be taken when the face is close enough to the camera, for a certain period of time (button "*q*" will work anyway) and the program is recognizing both the eyes.

Example:

python IrisDetection.py -o /home/pi/Desktop/prova -e face -p 1

When the software is run, depending on the version of *OpenCV*, it could appear the error ***VIDIOC_QUERYMENU: Invalid argument*** more times.

```
pi@raspberrypi ~/Desktop/Raspberry $ python IrisDetection.py -o /home/pi/Desktop/prova -e face -p 1
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
```

This error can be ignore and the program will continue to work correctly just after the messages (after a few seconds).

The only information shown during the execution of the program is the time, in milliseconds, between 2 different frames.

```
time taken for detection = 0.000119654ms
time taken for detection = 0.000102331ms
time taken for detection = 0.000102267ms
time taken for detection = 9.87481e-05ms
time taken for detection = 0.000103463ms
time taken for detection = 0.000101671ms
time taken for detection = 9.70491e-05ms
```

When the picture is taken, the led on the camera will switch off for a while, then it will turn on again. During the time it will be on, keep the position of the face as steady as possible, and the eyes well open (to permit at the camera to capture the entire iris), till the led will switch off again.

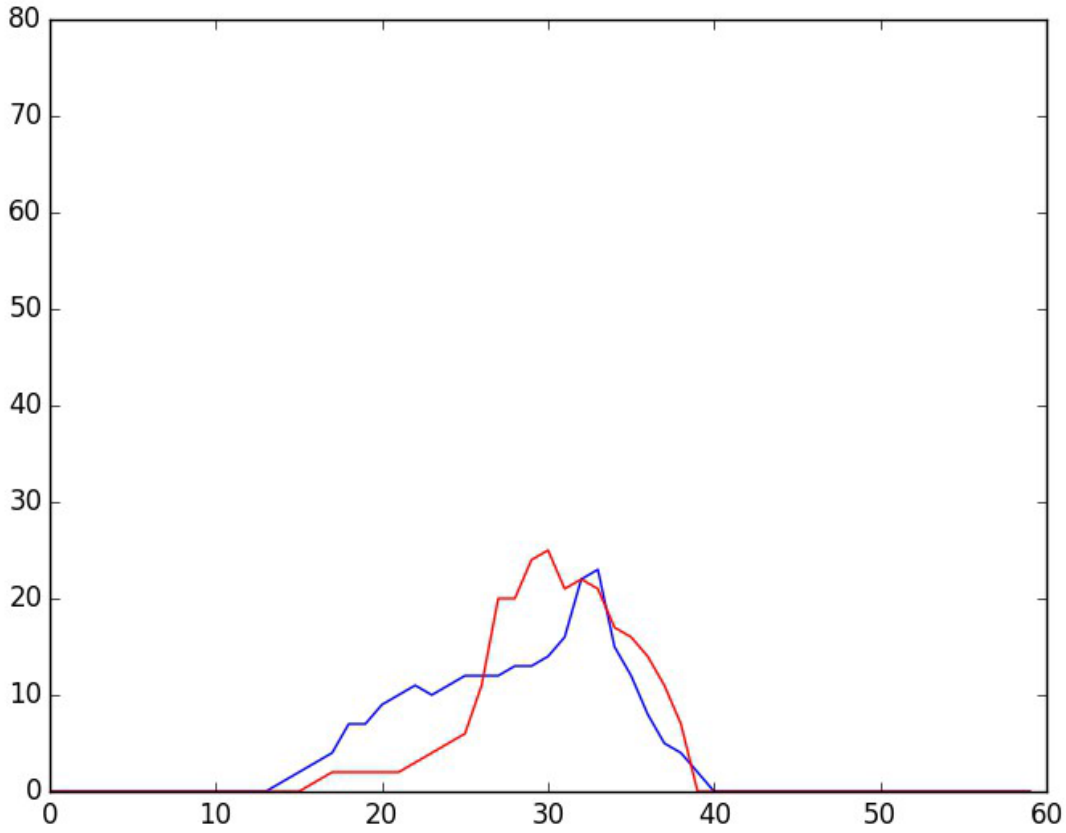
Description

Iris Detector is a software, written in Python, which allows the extraction of the iris, from a high definition picture, taken during the acquisition of a video of a face.
The program constantly acquires a low resolution video, and according to a certain event, it takes a high resolution picture of a face.
During the acquisition of a video, the software recognizes the position of the face, thanks to the function *detectMultiScale* (of the library *OpenCV*), which uses the classifier *HaarCascade*. Similarly, when a face is detected, eyes are extracted in the same way, from the rectangle containing the face. Eyes recognition is done with the use of a similar classifier.
When the capture of the high resolution picture is done, the library for the image capturing is changed, in particular the library *picamera* is used (after the command "*with picamera.PiCamera() as camera*").

At this point a picture, at a resolution of 2592x1944 (4:3) is taken. This picture will be used to extract the iris image from the eyes. A ROI of the picture is done, according to the coordinates calculated during the low resolution video acquisition. These coordinates are adapted to the resolution of the new picture, which has a higher resolution. From the ROIs of each eyes, the pupil is then extracted.
As first thing, from the ROI of an eye, an equalization of the image is done, with the function *equalizeHist()*, after that a binary thresholding process is applied to the equalized image, with the function *threshold()* and threshold level equal to 70. After this the center and the ray of the iris is calculated with the function *Canny()* (of the library *OpenCV*).
Then the function *getPulil()* (or *getPupil2()*) is launched, which takes the center and the ray of the eye as parameter, and returns the ray of the pupil.

getPupil()

This function, contained in the module functions.py, gets four parameters, which are the image of the eye, the coordinates of the center of the iris *x* and *y*, and the ray of the latter.
This function creates a ROI of the iris, according to the dimension passed as arguments, and it subtract an offset (a certain margin) from it. This can be done considering that the dimension of the pupil is certainly lower than the dimension of the iris. The latter image is normalized with the function *cv2.Normalize()*, highlighting the darkness of the pupil. This function get the pupil darker (values close to 0), and the rest of the iris with higher values in the gray scale. This operation is necessary, since after this, 2 histograms are going to be created, scanning the ROI vertically and horizontally. The width of the histogram should represent the width of the pupil.



Due to ambient and illumination condition, histogram could not correspond, or they could even have a not-bell shape. So the median and variance of the histogram could be different.
So width of the base of the histograms is calculated (the points in which those start, in the x-axis), and an average between them is done. This average represent the width of the pupil.

getPupil2()

This function is like the previous, but it tries to find the correct threshold for the generation of the histogram (adaptive thresholding). Practically the function *getHistograms()* is called, which iteratively generates histograms with an increasing threshold each time. As soon as the number of values equal to 0 is higher than a certain value, the iteration is stopped, returning the 2 histograms. The ray of the pupil is calculated as in the previous function.