# UNIVERSITY OF RZESZÓW

# Image Recognition

## Theoretical project: pupil and movement detection of the eye

First Name, Last Name:

Paulino de la Fuente Lizcano

3rd year of computer science, semester 5
first-cycle engineering studies

Rzeszów 2022

# Contents

# Illustration table

# 1.Project description

This project arises from the need to be able to capture the human eye and detect it within an image. For that reason, for the subject of image recognition we have developed a python script which either through a webcam or through a video, it is able to detect the human eye. We will process the image to take the centre of the pupil to recognize where the eye is.

# 2.Requirements

To be able to execute the following script, we need a series of libraries installed in our computer, if not, it will be impossible to execute it. The necessary files are:

- Python (https://www.python.org/downloads/)
- Opencv (pip install opencv)
- Numpy (pip install numpy)

Once all the libraries needed are installed, we can start executing the program.

# 3.FindContours

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. To use them, first we must convert the image into a binary one, thanks to the function **cv2.threshold.**

- For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection.
- Since OpenCV 3.2, findContours() no longer modifies the source image but returns a modified image as the first of three return parameters.
- In OpenCV, finding contours is like finding white object from black background. So remember, object to be found should be white and background should be black.

This function needs 3 parameters, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs a modified image, the contours and hierarchy. contours is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x,y) coordinates of boundary points of the object.

If you pass **cv.CHAIN_APPROX_NONE**, all the boundary points are stored. But actually, do we need all the points? For eg, you found the contour of a straight line. Do you need all the points on the line to represent that line? No, we need just two end points of that line. This is what **cv.CHAIN_APPROX_SIMPLE** does. It removes all redundant points and compresses the contour, thereby saving memory.

In this image we can see the difference between them. Taking like 354 points against 4

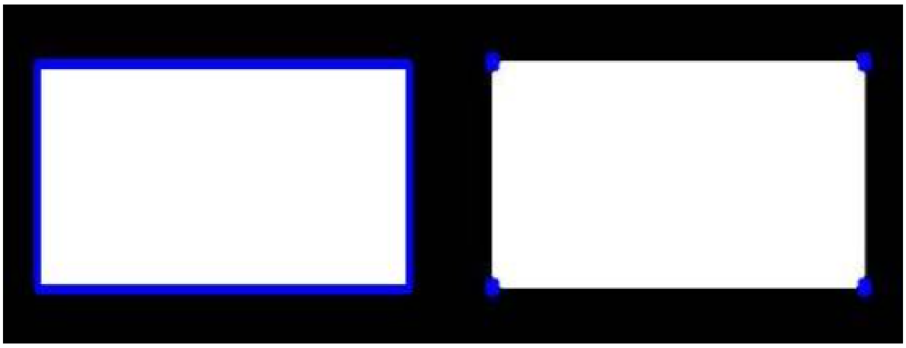


*Figure 1: Modes of processing points*

## 3.1 How does it work?

- When the computer is made to detect the edges of an input image, it then finds the points where specifically, there is a significant difference notice in the intensity of coloration, then simply those pixels are turned on. A stark difference noticed when the system is instructed to perform contouring.

- Contours are basically an abstract collection of segments and points that correspond to the reflective shapes of the objects that are present in the images that have been processed through the system. because of this, it is in our capacity to manipulate the contouring within the programs through which they are being accessed.

- This can be done in multiple ways, such as having a count on the number of contours in an image and then using that to categorize the object shapes, for segmentation of images or cropping objects from the image that is being processed and many more such similar functions.

# 4.Threshold

Thresholding is the binarization of an image. In general, we seek to convert a grayscale image to a binary image, where the pixels are either 0 or 255.

A simple thresholding example would be selecting a threshold value T, and then setting all pixel intensities less than T to 0, and all pixel values greater than T to 255. In this way, we are able to create a binary representation of the image.

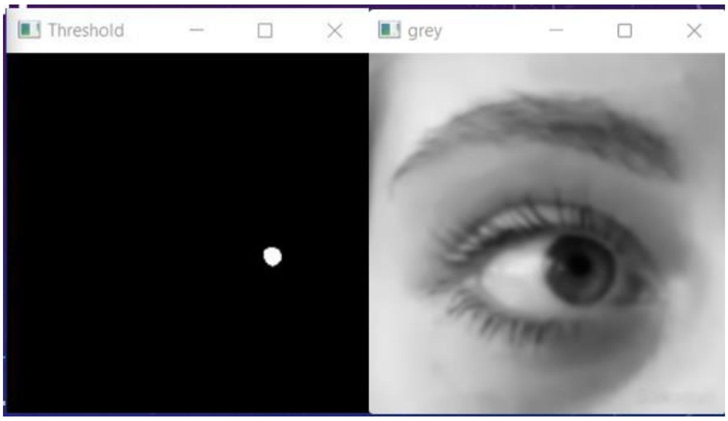


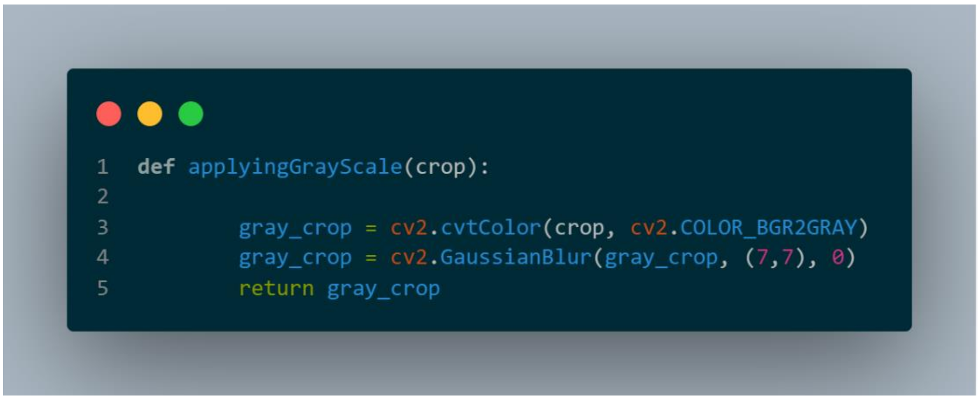
*Figure 2:Threshold*

On the left, we have the gray eye image that has been converted to grayscale. And on the right, we have the thresholded, binary representation of the eye.

## 5.Processing the image

In this section we are going to talk about how this project process the image and how the algorithm works to detect the eye.

First, we must understand some facts of the eye, there are 3 colors inside the eye, white, black and iri´s color, being the pupil is the darkest point in the eye. To start working on it, we first turn the image into a gray scale with the function showed in the picture below.



```
def applyingGrayScale(crop):

        gray_crop = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
        gray_crop = cv2.GaussianBlur(gray_crop, (7,7), 0)
        return gray_crop
```

*Figure 3:GrayScale method*

When we apply this method, we get a gray scale image, but this Is not enough for recognizing the pupil, this is just a required step to go over the problem.

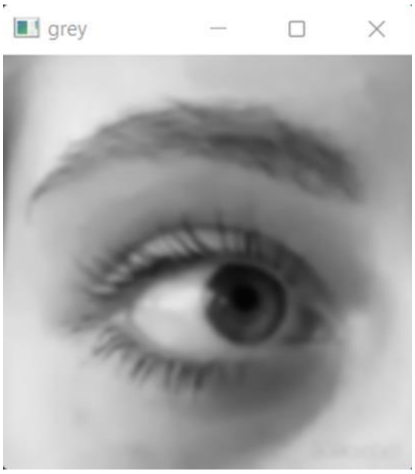
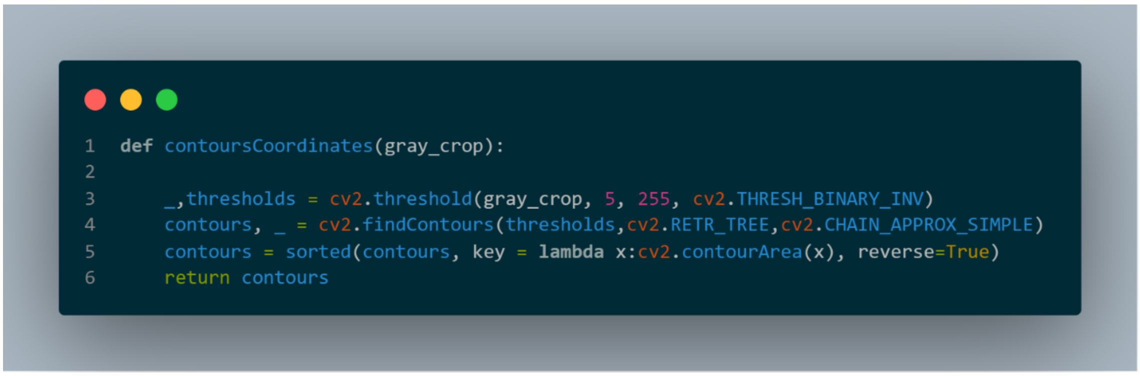


*Figure 4:Grey eye*

We can see that the darkest thing in the image is the pupil, but how can we distinguish between the pupil and the rest of the eye? The answered is using the function **cv2.threshold**

As I described before, this allow us to focus on the pupil ignoring the rest of the eye.

Once we obtain the thresholds, we need to identify the pixels that represent the pupil. To do so we will use the function **cv2.findContours()**, that returns us a list, or tree of lists of points. The points describe each contour, that is, a vector that could be drawn as an outline around the parts of the shape based on its difference from a background.



```
def contoursCoordinates(gray_crop):

    _,thresholds = cv2.threshold(gray_crop, 5, 255, cv2.THRESH_BINARY_INV)
    contours, _ = cv2.findContours(thresholds,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key = lambda x:cv2.contourArea(x), reverse=True)
    return contours
```

*Figure 5: Contorus method*

If we represent the contours on a 2-D plane, we will get that all the points are grouped together, simulating a circle. In this case we have a little noise on the image but is not so relevant because is due to the iri´s color that is brown.

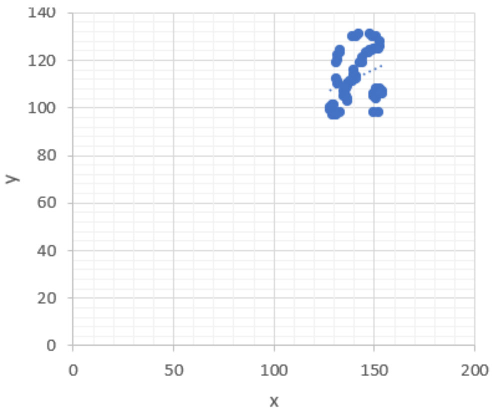
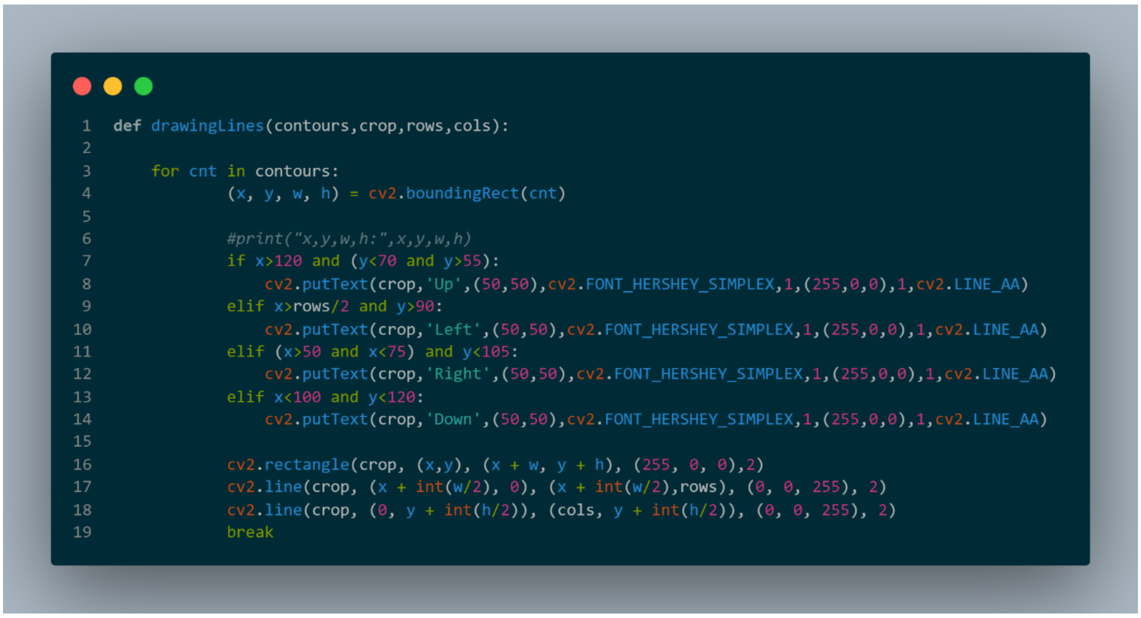


*Figure 6: Graph*

Once we have that vector, we need to get the biggest area to locate the pupil, by sorting those values, we will be able to detect the and follow the movement of the pupil. Those order values will be used to draw on the image.



```python
def drawingLines(contours,crop,rows,cols):

    for cnt in contours:
            (x, y, w, h) = cv2.boundingRect(cnt)

            #print("x,y,w,h:",x,y,w,h)
            if x>120 and (y<70 and y>55):
                cv2.putText(crop,'Up',(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0),1,cv2.LINE_AA)
            elif x>rows/2 and y>90:
                cv2.putText(crop,'Left',(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0),1,cv2.LINE_AA)
            elif (x>50 and x<75) and y<105:
                cv2.putText(crop,'Right',(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0),1,cv2.LINE_AA)
            elif x<100 and y<120:
                cv2.putText(crop,'Down',(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0),1,cv2.LINE_AA)

            cv2.rectangle(crop, (x,y), (x + w, y + h), (255, 0, 0),2)
            cv2.line(crop, (x + int(w/2), 0), (x + int(w/2),rows), (0, 0, 255), 2)
            cv2.line(crop, (0, y + int(h/2)), (cols, y + int(h/2)), (0, 0, 255), 2)
            break
```

*Figure 7: Drawing Image*

This function is the one use to determine where the eye is looking at, if we take into a count the Figure 6. We can supposed that the eye is in a coordinate axis so depending where the pupil is, we can calculated how far away the center of the image is from the pupil and we can determine where the eye is looking at.

The better the camera is, the better it will work, because if there is too much noise, the program will get crazy and don't detect the pupil. I upload an example of an eye being tracked so if you want to try with another video, just change the file define inside the function "videoCapture(name of the file)", if you want to use your own camera, you can put "videoCapture(0)" which means taking the image from the camera.

## 6.Conclusions

Developing this project has change my mind about image recognition, I thought that this was easy but not at all. You must take care about the quality of the image in order not to find too much noise and the program doesn't work properly. The hardware is very important so we can get a good image and applying the filters is not so easy, well not applying them but trying to find some patterns or some sections of an image. This field in computer science has created an interest on how the image can be processed. Thank you for your attention and hopefully you will like this project.