Architecture of microprocessors mini project

Interactive eye tracker

Gligor Catalin Andrei

Ciopec Andrei Sorin

Cuprins

1. Introduction:

The eye tracking project is designed to monitor the gaze direction of a user in real-time. It requires the ability to detect the user's eyes in a video feed, isolate the iris within each eye, estimate the position of the pupil, and then determine the direction of the gaze based on the position of the pupil relative to the iris.

To solve this problem, our project uses a combination of facial detection algorithms to locate the user's face and eyes, some image processing to isolate the iris from the rest of the face, and some machine learning algorithms to estimate the position of the pupil. The gaze direction is then determined by comparing the position of the pupil to the position of the iris.

One of the innovative aspects of our project is the use of real-time video for gaze tracking, which allows for dynamic interactions, like realizing a certain desktop related task when looking in a certain direction.

The interactive eye tracker project in Python combines innovative technology with creative functionality, offering a user-engaging experience. The system influences eye-tracking technology to detect specific gaze directions and responds dynamically to user input. When the user looks up, a YouTube video pops up and starts playing, providing an intuitive way to enjoy video content. Upon directing their gaze to the right, the Turtle module from python will launch and will start to visually draw the Pikachu character. Additionally, looking left triggers a terminal based dice rolling game. This approach underlines the versatility of an eye-tracker to implement diverse functionalities, making it an innovative piece of code.

2. Theoretical background:

The theoretical foundations of gaze tracking are found in multiple topics, from machine learning to image processing.

In our project, the face and eyes in a picture can be identified using the Haar cascades or other algorithms of a similar nature. The iris inside the eye is then separated using image processing techniques. In order to determine the iris' margins, this involves converting the image to grayscale, setting a binary threshold, and utilizing edge detection techniques.

Next, an estimate of the pupil's location inside the iris is made. Finding the darkest area of the iris, which represents the pupil, is how this is done. The direction of the gaze can therefore be concluded from the relative location of the pupil within the iris.

Furthermore, gaze tracking accuracy can be increased by the application of machine learning and from mathematical calculations. For instance, a machine learning model can be taught to identify the pupil and iris in a picture or to determine the direction of gaze based on the pupil's location. The mathematical part comes into play when finding the direction of the gaze using the vertical and horizontal ratios, used in our project.

These theoretical ideas are used by the project's gaze tracking technology to follow the user's gaze in real time. It locates the eyes using the webcam and machine learning, isolates the

iris and estimates the pupil's position using image processing techniques and some mathematics, and enhances the gaze estimation accuracy with machine learning algorithms.

3. Implementation:

Example.py

It uses a webcam to track the user's eye movements in real-time. Here's a brief summary of its functionality:

- It initializes the GazeTracking object and sets up the webcam using cv2.VideoCapture(0).
- The script enters a loop where it continuously reads frames from the webcam and sends them to the GazeTracking object for analysis.
- The GazeTracking object analyzes the frame to determine the user's eye movements. It can detect the gaze of the user's eyes, or whether they are blinking.
- The script prepares an annotated frame that shows the user's eye movements and displays a text message indicating the detected eye movement.

Gaze_tracking.py

This script is responsible for determining the direction of the user's gaze. It provides methods to check if the user is looking right, left, up, down, or even blinking. Here's a short description of its functionality:

- **is_right(self)**: This method checks if the user is looking to the right. It does this by checking if the horizontal ratio of the pupil's coordinates is less than or equal to 0.4 (these values were obtained in mathematical ways and with the trial and error method).
- **is_left(self)**: This method checks if the user is looking to the left. It does this by checking if the horizontal ratio of the pupil's coordinates is greater than or equal to 0.6.
- **is_center(self)**: This method checks if the user is looking at the center. It does this by checking if the horizontal ratio of the pupil's coordinates is between 0.45 and 0.55.
- **is_up(self)**: This method checks if the user is looking up. It does this by checking if the vertical ratio of the pupil's coordinates is less than 0.6.
- **is_down(self)**: This method checks if the user is looking down. It does this by checking if the vertical ratio of the pupil's coordinates is greater than or equal to 0.5.

Each of these methods checks firstly if the coordinates of the left and right pupils have been detected. If they have, it then calculates the horizontal or vertical ratio of the pupil's coordinates to determine the direction of the gaze.

Dice.py

Dice.py is just a short and basic script for a simple dice rolling game, where the faces of the dice are drawn beforehand using the keyboard and placed in an array.

When the ***roll_dice_game()*** function is called, it generates a random number between 1 and 6, and then print the string representation of a dice showing that specific number.

Pikachu.py

Pikachu.py is a script that uses the Turtle module in Python to draw a representation of Pikachu, the character from Pokémon. The *turtle module* is a part of Python's standard library. It's a useful tool for simple draws for example and it's also a powerful enough module for high level programmers to use for graphics-related tasks.

Pupil.py

The pupil.py script contains a Pupil class that is responsible for detecting the iris of an eye and estimating the position of the pupil. The class is initialized with an eye frame, which is an array containing an image of an eye, and a threshold value used for image "binarization". The Pupil class has a method image_processing which performs operations on the eye frame to isolate the iris.

Eye.py

Eye.py is a portion of a procedure that handles an eye inside a specified frame. Using the points variable, it first establishes if the eye is the left or right eye. The *_blinking_ratio* method is then used to compute exactly what the name tells us, making use of the eye's facial landmarks to identify whether the eye is closed, in this case signaling a blink. The eye in the original frame is then isolated using the given landmarks by calling the _isolate function. It evaluates the side of the eye and the frame if the calibration is incomplete. After retrieving the side of the eye threshold from the calibration, it builds a Pupil object using the separated eye frame and the threshold.

Calibration.py

The calibration.py code is responsible for calibrating the gaze tracking system. It determines the optimal threshold values for distinguishing the iris from the rest of the eye in a certain frame.

The evaluate method improves the calibration by considering the given eye frame. It first finds the best threshold for that specific eye frame using the *find_best_threshold* method. This method computes the size of the iris for different threshold values and returns the threshold that results in an iris size closest to the average iris size.

The evaluate method then appends the best threshold to the list of thresholds for the appropriate eye (left or right), as indicated by the side argument. This allows the system to adapt to changes in lighting conditions and other factors that could affect the appearance of the iris in the frame.

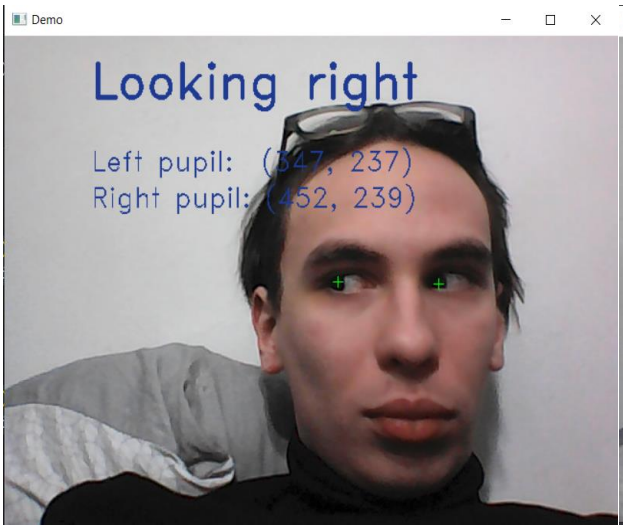
4. Conclusions:

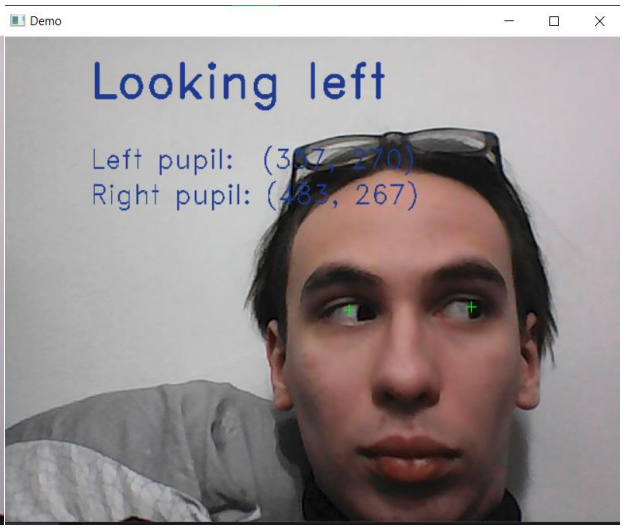
The eye tracking project is a pretty advanced program that tracks a user's gaze direction in real-time by using computer vision and machine learning algorithms, along with the user's own

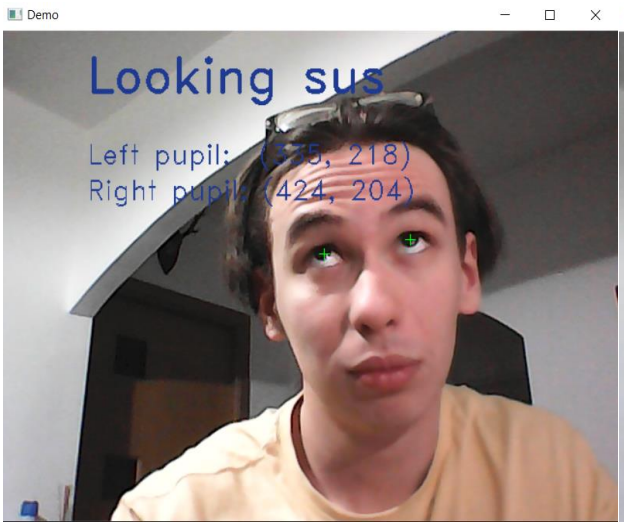
webcam. It locates the user's eyes in a video feed using facial detection methods, separates each eye's iris using image processing techniques, and estimates the pupil's location using machine learning algorithms. Finally, by matching the pupil's position to the iris' position, the gaze direction is established.

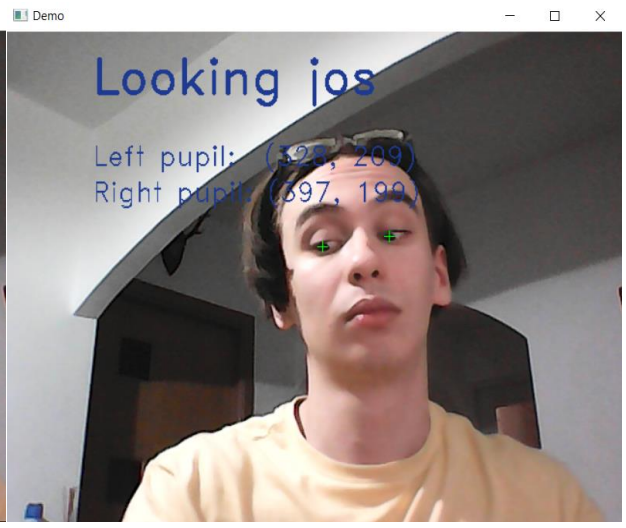
5. Annexes (Functionality):

In the next pictures we can see the functionality of the eye tracker:









6. Bibliography:

https://github.com/antoinelame/GazeTracking

https://pythondex.com/awesome-python-turtle-codes

https://www.youtube.com/watch?v=x-Ag2_bJ40Y&pp=ygUQcHl0aG9uIGRpY2UgZ2FtZQ%3D%3D

https://chat.openai.com/