STUDENT INFORMATION

Name: Artemis Georgopoulou
Student ID: 235319
IMLEX, M2

ASSIGNMENT

Record your eye movements using your smartphone. Capture the saccadic eye movements while reading letters and the pursuit movement as you smoothly follow the motion of your fingertips. After recording, extract the coordinates from the video and create graphs with the horizontal axis representing time and the vertical axis representing position. Analyze these graphs to estimate the speed of each eye movement. Summarize your findings in a comprehensive report and submit it.

PROCESS SUMMARY

The steps I took to complete the assignment were the following:

1) Recorded 2 short videos of my eyes from my phone. One while reading a line of text, and one while following my thumb in a line.
2) Wrote code in python+OpenCV to extract x and y coordinates of the right eye (as it appears in the video) for each video. Every 10 frames, each video stopped and I clicked on the screen where the right eye is, to save the coordinates. Then, I also extracted the timestamps of the frames I measured the eye, and saved everything in numpy arrays for further processing.
3) Wrote some more code to make the plots of the timestamps and the position x and y coordinates, for each video. Additionally, to find the speed of the eye movement for each video, I used the gradient function of numpy of the coordinates and plotted it against time.
4) Lastly, I made this report and discussed the findings.

RECORDING

The videos were rather short (3-4 seconds). I used an online tool to crop the videos and center the eyes, and also time-crop the part that my eyes moved to start and end the video, which were irrelevant for our analysis.

Figure 1: Indicative frame of video and position of clicking

CODING

Here I will shortly explain the code I wrote in Python + OpenCV to efficiently extract position coordinates of the eyes while moving.
For every 10 frames, I show the corresponding frame of each video on the screen and a click_event function is called once I click at my right pupil (as the video appears, see Figure 1) that saves the (x,y) coordinates. The arrays 'coordinates_1' (for x coordinate) and 'coordinates_1y' are for the smooth pursuit video. And the 'coordinates_2' (for x coordinate) and 'coordinates_2y' are for the reading video. The corresponding time arrays are to save the timestamps of the frames that appear. In the end I save all the arrays as numpy arrays to process more in another script.
Below you can see the script for the coordinate and time extraction.
I think the script used for plotting the results is redundant, so I will not include it.

```
import cv2
from matplotlib import pyplot as plt
import numpy as np

# Arrays to store selected x and y-coordinates, and timestamps
coordinates_1 = []
coordinates_2 = []
coordinates_1y = []
coordinates_2y = []
```

```
time_1 = []
time_2 = []

def click_event1(event, x, y, flags, params):
   if event == cv2.EVENT_LBUTTONDOWN:
       coordinates_1.append(x)
       coordinates_1y.append(y)
       print(x, ' ', y)

def click_event2(event, x, y, flags, params):
   if event == cv2.EVENT_LBUTTONDOWN:
       coordinates_2.append(x)
       coordinates_2y.append(y)
       print(x, ' ', y)

# Open the videos
video_path_1 = 'smooth_persuit (online-video-cutter.com).mp4'
cap_1 = cv2.VideoCapture(video_path_1)

video_path_2 = 'reading (online-video-cutter.com).mp4'
cap_2 = cv2.VideoCapture(video_path_2)

# Create windows for displaying videos
cv2.namedWindow('Video 1')
cv2.namedWindow('Video 2')

# Set mouse callback function
cv2.setMouseCallback('Video 1', click_event1)
cv2.setMouseCallback('Video 2', click_event2)

# Variables for timing
frames_passed = 0

while True:
   # Read frames from the videos
   ret_1, frame_1 = cap_1.read()
   ret_2, frame_2 = cap_2.read()

   # Break the loop if any of the videos end
   if not ret_1 and not ret_2:
       break

   if frames_passed % 10 == 0:
```

```
        if ret_1:
            frame_1 = cv2.resize(frame_1, (1000, 700))
            cv2.imshow('Video 1', frame_1)
            time_1.append(cap_1.get(cv2.CAP_PROP_POS_MSEC))

        if ret_2:
            frame_2 = cv2.resize(frame_2, (1000, 700))
            cv2.imshow('Video 2', frame_2)
            time_2.append(cap_2.get(cv2.CAP_PROP_POS_MSEC))

        cv2.waitKey(0)

    frames_passed += 1

    # Break the loop if 'q' key is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release video capture objects
cap_1.release()
cap_2.release()

# Destroy all OpenCV windows
cv2.destroyAllWindows()

# Print and save the selected coordinates
coordinates_1 = np.array(coordinates_1)
coordinates_2 = np.array(coordinates_2)
coordinates_1y = np.array(coordinates_1y)
coordinates_2y = np.array(coordinates_2y)

time_1 = np.array(time_1)
time_2 = np.array(time_2)

# Save the coordinates as numpy arrays
np.save('coordinates_1.npy', coordinates_1)
np.save('coordinates_2.npy', coordinates_2)
np.save('coordinates_1y.npy', coordinates_1y)
np.save('coordinates_2y.npy', coordinates_2y)
np.save('time_1.npy', time_1)
np.save('time_2.npy', time_2)
```

RESULTS ANALYSIS

In the plots of figure 2 we can visualize the x and y position against time, as well as the speed. The timestamps are annotated in milliseconds.
The speed (= change in y / change in x) was calculated in python as follows (example, for a set of coordinates and a set of timestamps):

```
speed = np.gradient(coordinates, time)
```

On the top left, we can see the x coordinates of smooth pursuit video (blue line) and reading line video (orange line) against time. It's easy to observe that in the smooth pursuit the line is also much smoother, indicating less pronounced changes in speed and not very big saccades. On the other hand, the reading line is much less smooth with the eyes doing saccades, jumping from word to word. The x position of reading doesn't have a big span, as the line I recorded myself reading was smaller in distance than the trajectory my thumb followed.
The same results apply to the speed plot of x coordinates, on the bottom left. The changes in speed of the smooth pursuit are smoother, almost resembling a parabolic function (my eyes were faster in the middle of the pursuit and slowed down in the start and end). More specifically, it is increasing in a negative way in the middle, and going back closer to zero in the beginning and end. In the line reading however, there are variations in speed, showing again the presence of pronounced saccades.

Now let's take a look at the y coordinate as well, although I think the x coordinate carries more useful information for the eye movements in a horizontal movement (in both videos, my eyes moved along a horizontal line more or less).
We can see that the y values don't have a big range (approximately 400-440, Figure 2 top right). In the line reading it's even less. That shows that the eyes followed indeed, mostly a straight line. In the smooth pursuit there is a slight increase in the end, and I believe it is because I unconsciously raised my thumb a bit, so my eyes followed. I don't think there is anything conclusive coming from looking at the y-coordinate speed changes (Figure 2 bottom right), as it probably comes because of my slightly unstable head position

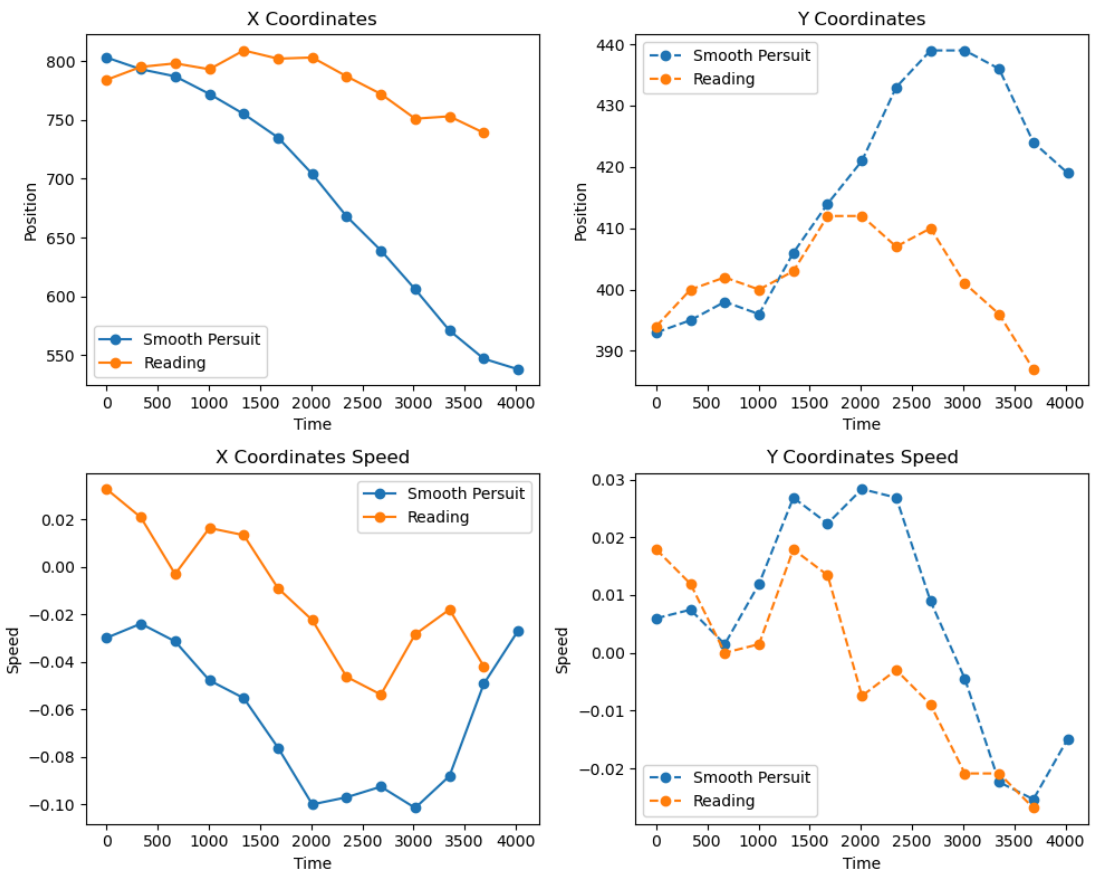

Figure 2: Position and Speed plots of x and y coordinates

CONCLUSION

Overall, I think we can see the difference in saccades between the smooth pursuit and the line reading. The results were mostly expected, as in the smooth pursuit the eyes follow along a line without jumping. In the line reading on the other hand, we expected to see more jumps, or saccades, corresponding to my eyes jumping from word to word.

The specific data and plots however might be a bit corrupted because of my slight head movements while capturing the videos.