ELEC-E7852 – A3b – Fall 2024

Ida Parkkali

Experimental method:

The experiment was done using WebGazer [1]. The WebGazer was downloaded using Brew and node.js. There were some problems with dependencies, but these were fixed using an older version of node.js. The WebGazer was integrated to a JavaScript program which showcased the UI (photo of a Fazer Café website).

The participant was asked to be part of an experiment for a university assignment about eye tracking. The participant was asked to first calibrate the eye tracking software by looking at the dots on the screen at the same time they are clicking them. After that, the participant was asked to look at the screen however they wanted. After 10 seconds the instructor stopped the experiment. This was repeated three times for the different iterations of the UI view, but the participant was not told about this. The participant allowed their picture to be part of this report [Figure 1].

The JavaScript program took care of the calibration process and tracked the movement of the eye to a CSV file. The logic of eye tracking from a view with calibration was based on the calibration.html file example that was provided in the WebGazer file. Making of CSV files, making the view to only have the background picture, and debugging was done using ChatGPT to meet this experiment's needs.

The gathered data was modified using Python on Jupiter Notebook. ChatGPT was used to normalize the datapoints and to add plot lines, because milliseconds were not practical for the scatterplot. 0 is closer to the beginning, 1 is closer to the ending of the experiment. It was ran one by one for each CSV file.

Figure 1, picture of the participant completing the task

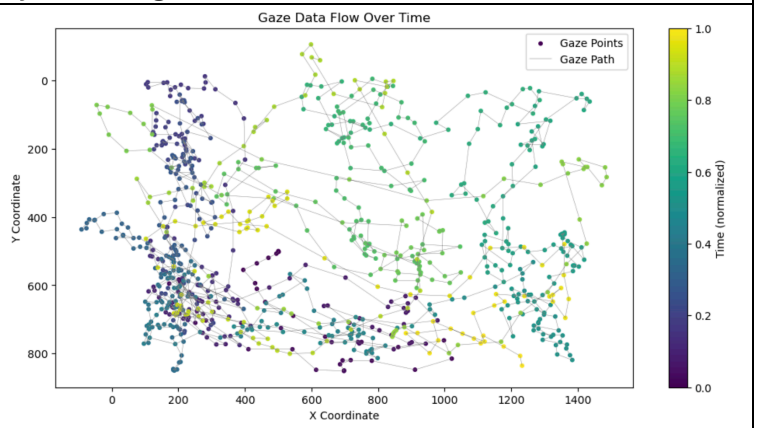
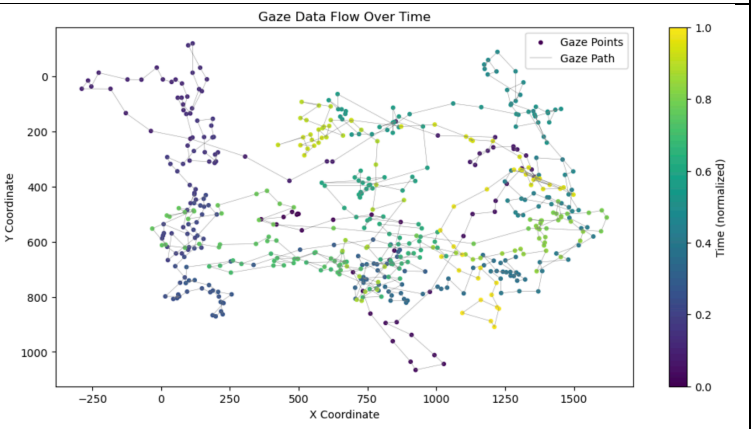
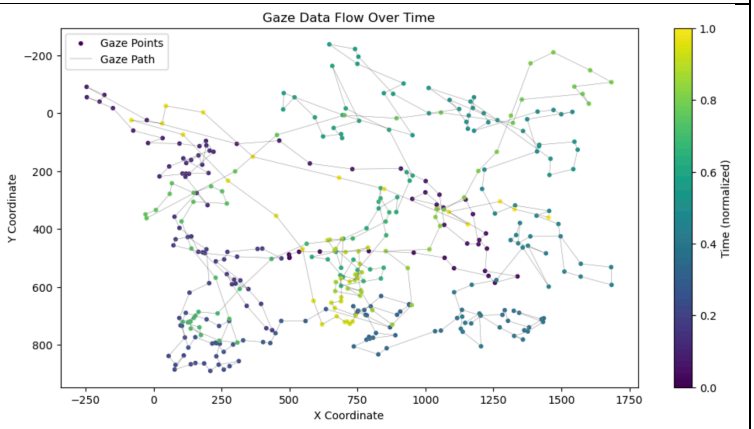
```
<script>
  let gazeData = []; // Array to store gaze data

  window.onload = async function() {
    await webgazer.setRegression('ridge')
      .setGazeListener(function(data, elapsedTime) {
        if (data == null) {
          return;
        }
        const x = data.x;
        const y = data.y;
        const timestamp = Date.now();
        console.log(`Gaze coordinates: (${x}, ${y})`);

        // Update the position of the gaze indicator
        const gazeIndicator = document.getElementById('gazeIndicator');
        gazeIndicator.style.left = (x - 10) + 'px';
        gazeIndicator.style.top = (y - 10) + 'px';

        // Store the data point as an array of CSV-friendly values
        gazeData.push([x, y, timestamp]);
      })
```

Figure 2, the JS program to start eye tracking and to gather data

```
# Made by Ida Parkkali 1010662

import pandas as pd
import matplotlib.pyplot as plt



# Load CSV data
df = pd.read_csv('gazeData (3).csv')

# Normalize the timestamp to a range between 0 and 1 for color mapping
df['time_normalized'] = (df['timestamp'] - df['timestamp'].min())
/ (df['timestamp'].max() - df['timestamp'].min())

# Create a larger figure
plt.figure(figsize=(12, 6))

# Plot the points with color gradient
plt.scatter(df['x'], df['y'], c=df['time_normalized'], cmap='viridis', s=10, label='Gaze Points')
# Plot lines connecting the points
plt.plot(df['x'], df['y'], color='gray', linewidth=0.3, label='Gaze Path')

# Set titles and labels
plt.title('Gaze Data Flow Over Time')
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')
plt.gca().invert_yaxis()  # Invert y-axis to match screen coordinates

# Add a color bar for time progression
plt.colorbar(label='Time (normalized)')

# Add a legend
plt.legend()

plt.show()
```

Figure 3, the Python code for plotting

Results:

	Visual flow from a previous assingment	Eye tracking
1		
2		
3		

Discussion:

Looking at the eye tracking data, it is clear that the participant looked at the middle/bottom of the view first (the dark blue color in the plots). However, the results are not clear enough to indicate that the different iterations have a drastic difference. In all of the eye tracking experiments the participant also looked at the log in button after looking at the middle buttons (the turquoise color) in the upper right corner.

Additionally, it should be noted that the first eye tracking CSV has more datapoints, because on that try the participant was taught how the calibration works. Therefore, if we look at test 2 and 3 for eye tracking, we can see that the participant looked at the bottom of the view more in the test 3, which might indicate that the more colorful layout caught the attention more.

One point that was surprising is that how much the participant also looked at the right side of the view compared to the visual flow paths. This might be due to the visual flow algorithm not having a long enough span of attention to predict. Additionally, this might be because of bias due to the participant learning the view from the previous tests. Also, the calibration view still showed the Fazer Café view in the background, which is not idea for bias too. It should be also pointed out that in test 3 for some reason I got negative values, which might be due to the eye tracking software not being precise enough.

Conclusions:

Eye tracking concluded that the participants actually looked at the buttons in the middle, but it could not determine confidently that the different iterations had a difference between each other. To make the experiment more reliable, there should be more participants for the experiment and the eye tracking software should be more precise.

References:

[1] https://github.com/brownhci/WebGazer