Aalto-yliopisto
Aalto-universitetet
Aalto University

ELEC-E7852
Computational Interaction and Design

Assignment A5b

Computational Rationality

Aitor Urruticoechea Puig
aitor.urruticoecheapuig@aalto.fi
Student N°101444219
November 2024

Assignment A5b
Aitor Urruticoechea Puig
November 2024

Acknowledgement

Parts of the code implemented in this work were generated with the assistance of GitHub Copilot. None of its proposals were, however, employed without refinement and necessary tweaks to adapt it to the nuances of the tasks at hand; making it impossible to identify and subsequently mark which lines of code were human or machine-made, for many were the fruit of the combination of both.

Contents

List of Figures

1 Introduction

In assignment A5a, the main proposal was to introduce empirically observed mean eye movement speed [1] as a variable to be considered in the simulation for saccades, rather than considering it a fully random thing. When the model was re-run, this resulted in *"longer saccades and understandably longer fixation durations"* when compared to the original model. This was *"understood to happen because with a larger target width, the distance between the fixation and the target increases"* (as per the presented report A5a). Now, in this assignment, it is time to test if this relation happens in real life as well, by collecting and analysing actual experimental, real-world, data.

2 Approach and Method

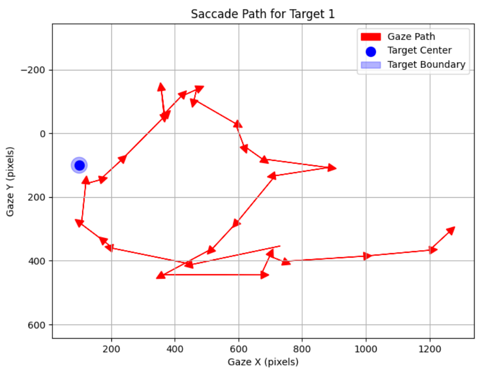
With the end of collecting the gaze data, the code used in assignment A3b using the provided webgazer.js tool has been adapted for this case. This time, after a quick calibration, five targets will be shown each for 3 seconds, and the eye movement of the user will be recorded. The recorded data is saved in .json format, so this time there is no need for a screen recording and manual data treatment. Instead, the recorded data can be directly interpreted by, for instance, a Python script to both show the eye movement and subsequently extract the actual relation between target width and movement time. Here one can consult some of the main parts of the code necessary to plot the figures and, previously, run the experimental data collection:

```python
# Create a figure for each target
for target_id, target_info in gaze_data.items():
    target_details = target_info["targetDetails"]
    gaze_points = target_info["gazePoints"]

    # Extract gaze coordinates and timestamps
    gaze_x = [point["gazeX"] for point in gaze_points]
    gaze_y = [point["gazeY"] for point in gaze_points]
    timestamps = [point["time"] for point in gaze_points]

    # Calculate movement time as the difference between the first and last timestamp
    if timestamps:
        movement_time = timestamps[-1] - timestamps[0]  # ms
        movement_times.append(movement_time)
        target_widths.append(target_details["size"])  # Use the target size as its width

    # Plot the gaze path with arrows
    plt.figure(figsize=(8, 6))
    for i in range(len(gaze_x) - 1):
        plt.arrow(gaze_x[i], gaze_y[i], gaze_x[i + 1] - gaze_x[i], gaze_y[i + 1] -
        ↪  gaze_y[i],
                  head_width=25, head_length=25, fc='red', ec='red', label="Gaze Path" if i
                  ↪  == 0 else "")
    plt.scatter([target_details["x"]], [target_details["y"]], color="blue", s=100,
    ↪  label="Target Center")
    # (continues)
```

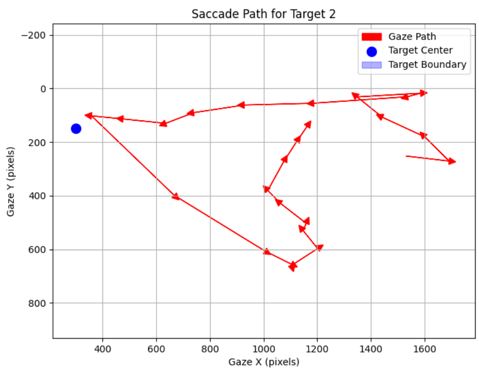
```
<script>
    window.onload = function () {
        //...
        // Function to update the target's size and position
        function updateTarget() {
            if (currentTargetIndex >= targetConfigs.length) {
                target.style.display = 'none';
                console.log("Experiment complete. Downloading gaze data...");
                downloadGazeData();
                return;
            }

            const config = targetConfigs[currentTargetIndex];
            target.style.width = `${config.size}px`;
            target.style.height = `${config.size}px`;
            target.style.left = `${config.x - config.size / 2}px`;
            target.style.top = `${config.y - config.size / 2}px`;
            target.style.display = 'block';

            // Initialize an array to store gaze data for the current target
            gazeData[`target_${currentTargetIndex}`] = {
                targetDetails: config,
                gazePoints: []
            };

            // Move to the next target after 3 seconds
            setTimeout(() => {
                currentTargetIndex++;
                updateTarget();
            }, 3000); // Show each target for 3 seconds
        }

        //...
        // Listen for gaze data
        webgazer.setGazeListener(function (data, elapsedTime) {
            if (data == null) {
                return;
            }

            const xprediction = data.x;
            const yprediction = data.y;

            // Update the gaze pointer
            gazeDot.style.left = `${xprediction - 5}px`;
            gazeDot.style.top = `${yprediction - 5}px`;
            gazeDot.style.display = 'block';

            // ...
        });
        // ...
            }
        });
    };
</script>
```

3 Results and Conclusions

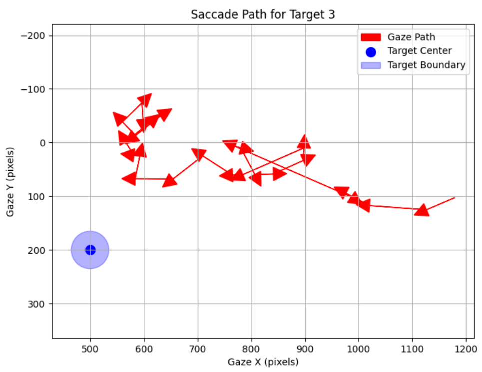
After running the experiment with the author's partner as a test subject, the results have been plotted (see Figure 1). This already gives us a view on one of the main issues that this experiment has run into. The method for tracking the eye movement is quite imperfect. Even with the preliminary calibration step, it is only able to provide for an approximate position for what the eye is actually fixating in. In the most extreme of cases (i.e. Figure 1e), the actual results seem to indicate that the test user did not even look to any point even close to the target. Overall, this adds a lot of question marks around the actual accuracy of the obtained results.



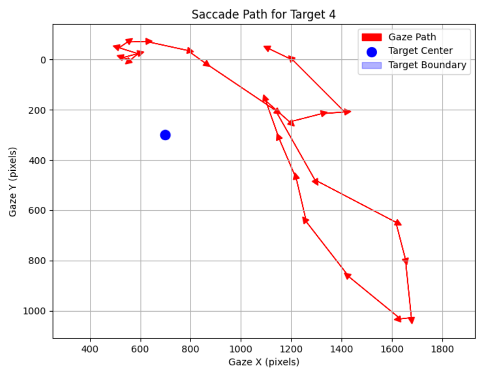
(a) Gaze results for target 1.



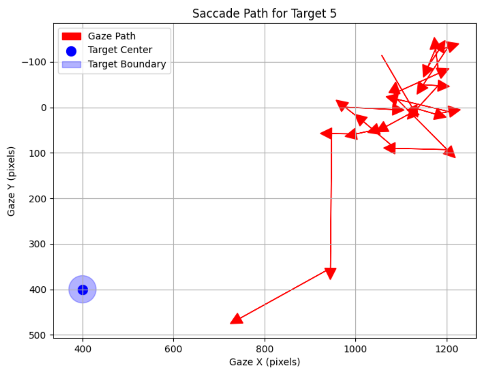
(b) Gaze results for target 2.



(c) Gaze results for target 3.



(d) Gaze results for target 4.



(e) Gaze results for target 5.

Figure 1: Gaze tracking results for 5 different targets, with different positions and width.

To actually prove or disprove the theory that supported the model change in assignment A5a, it is of interest to plot the actual found relationship between Target Width and eye Movement Time (Figure 2). Note that for this plot, the results gathered for the fifth target (Figure 1e) have been omitted as they are not even close to reliable enough to draw any meaningful conclusions.

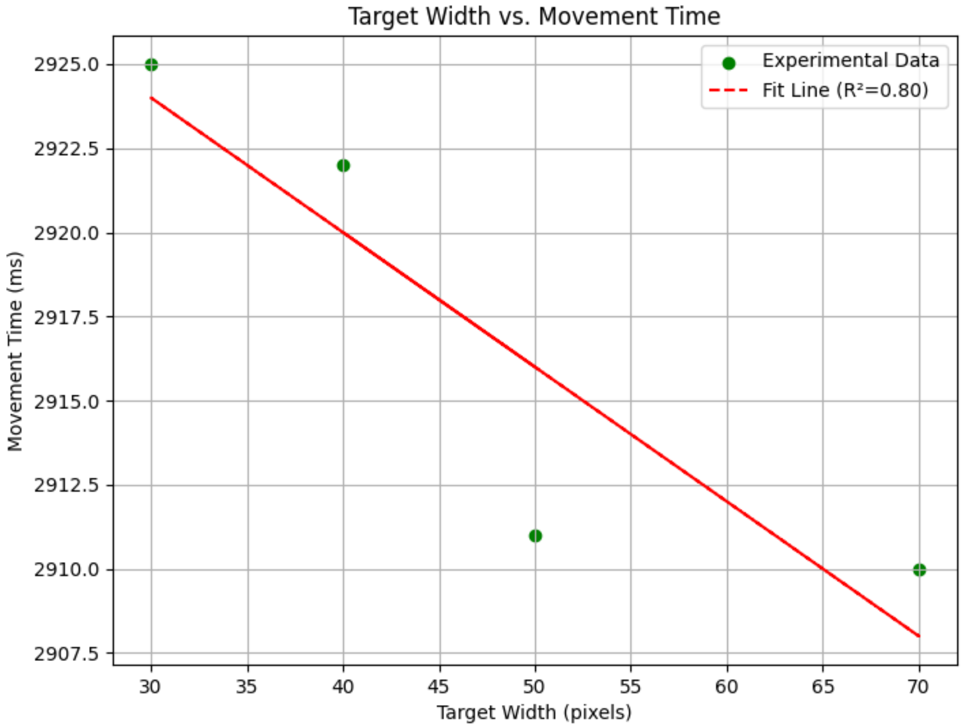


Figure 2: Target width versus movement time.

With the, albeit doubtful, experimental data collected in this short test run, these preliminary results seem to indicate that the original model approach with randomness is able to better reflect the observed real relationship. There indeed seems to be an actual inverse relationship between width and movement time (unlike the predictions based on saccade movement time). Nonetheless, this is quite a limited experiment to draw any meaningful conclusions. Not only the obtained data is far from meticulous enough, but there are also other moving variables that should be considered further. For instance, targets were shown not only with different sizes, but also in different positions in the screen, trying to play around with the effects this would have on human peripheral vision. However, this has not been taken into consideration when looking at the direct comparison between Target Width and Movement time.

References

1. ROBINSON, David A. The mechanics of human saccadic eye movement. *The Journal of Physiology*. 1964, vol. 174, no. 2, pp. 245–264. Available from doi: 10.1113/JPHYSIOL.1964.SP007485.