04-630: Data Structures and Algorithms for Engineers

Assignment I: Identifying Fixation Points for Eye Trackers

Deadline: 19:00, Friday, 9th February 2024

Problem Definition

Automatic eye trackers track the eyes movements and the user's point of gaze to determine the user's visual attention, the objects of focus, and the objects they ignore. The user's point of gaze is referred to as a fixation point. As the user's gaze moves from one fixation point to another, a scanpath is formed. A sample of a scanpath is shown in Fig 1 and Fig 2. Eye trackers capture information about the scanpath that includes the time spent at a fixation point as well as the coordinates of the fixation point.

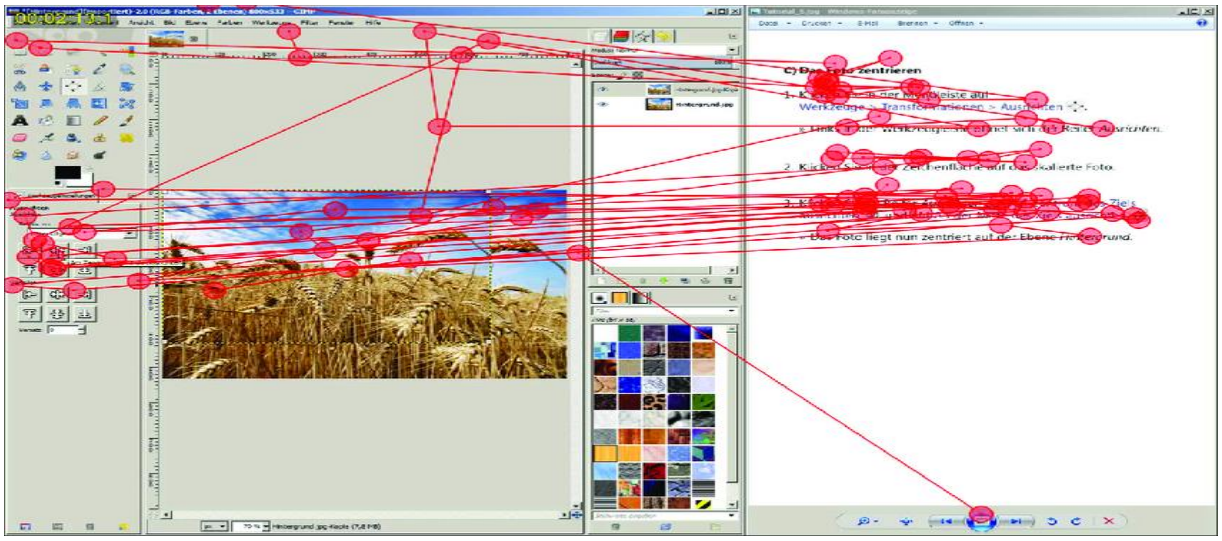


Figure 1: Sample scanpath generated from eye movement data of one study participant with concurrent presentation of tutorial and software application[1]

[1] Meng, Michael. (2020). Using eye tracking to study information selection and use: A research perspective.

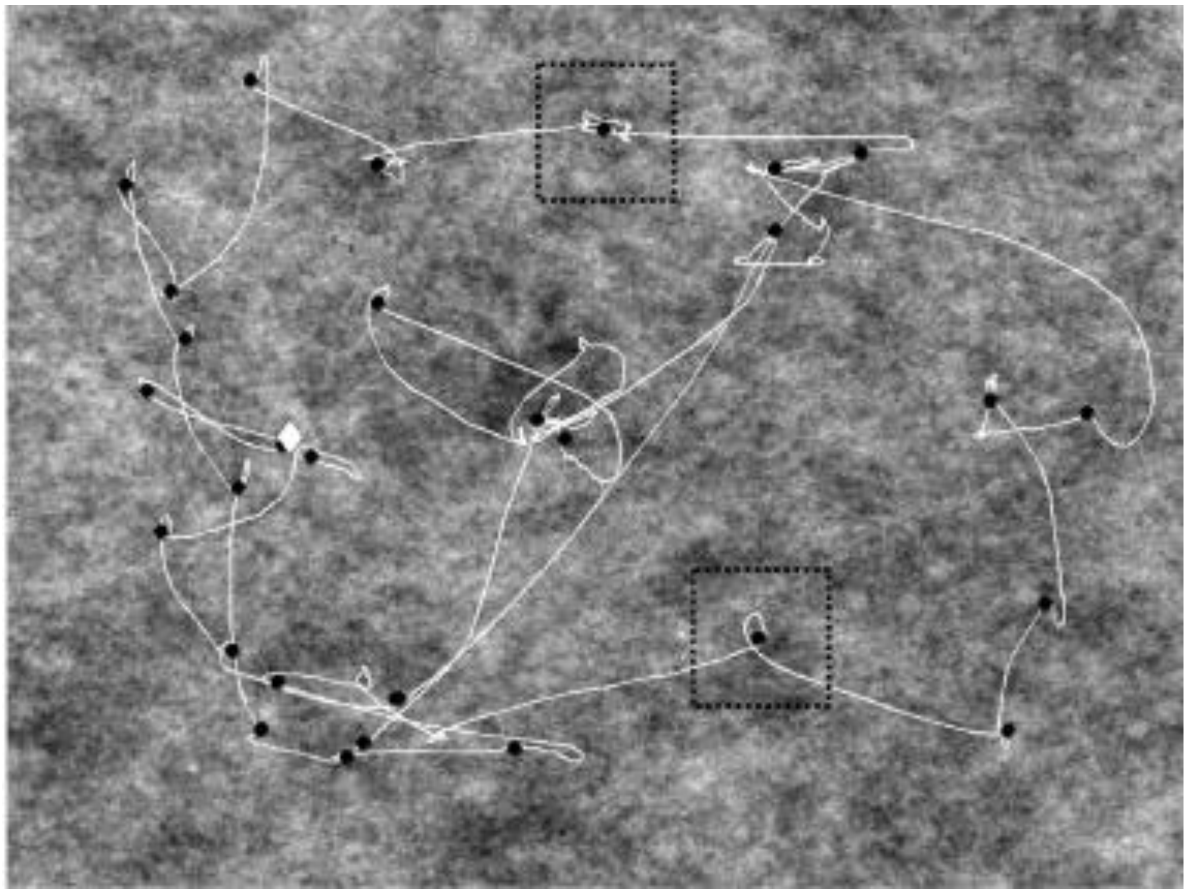


Figure 2: Dots indicate fixations. Diamond shows location of 'Triangle' target[2]

Eye tracking may be used to identify the objects a user focuses on in a scene, the sections of interest in a webpage, the amount of time the user spends on a fixation point indicating the length of visual attention, the number of times a user revisits a fixation point, and the order in which the fixation points are encountered.

Eye tracking technology has multiple practical applications, e.g., studying learner behavior in e-learning systems, gauging driver's visual attention in automotive research to help in layout design for dashboards, medical research and diagnosis of conditions such as Alzheimer's disease, attention deficit hyperactivity disorder (ADHD), and mental health monitoring e.g. Schizophrenia, etc.

The purpose of this assignment is to design and implement an algorithm to process a list of fixation points, each specified by an identifier, and its x and y coordinates. The algorithm will create a unique identifier for each distinct point and produce a list of distinct fixation points. A fixation point is distinct if its coordinates have not been encountered before. The unique identifier should be an integer number beginning with 1 and incrementing by 1 and based on the order in

[2] Rajashekar, Umesh & Cormack, Lawrence & Bovik, Alan. (2004). Point of Gaze Analysis Reveals Visual Search Strategies. Proceedings of SPIE - The International Society for Optical Engineering. 5292. 10.1117/12.537118.

which the distinct fixation points were encountered in the original list. To determine that a fixation point is distinct, only use the x and y coordinates. The algorithm should be implemented as a C/C++ program. The program will accept one or more test cases, with each test case terminated by the coordinates (-1,-1).

Input

The first line of input comprises an integer *N* (1<= *N* <=10); this indicates the number of test cases to be processed, i.e. the number of scanpaths, in the input file. It is followed by *N* sets of fixation points. Each fixation point comprises three numbers: the fixation point number, the *x* coordinate of the fixation point, and the *y* coordinate of the fixation point. The last fixation point in each set is signified by *x* and *y* coordinates equal to minus one, i.e., the (-1,-1) coordinate. You can assume that there are no more than 1000 fixation points in one scanpath sequence and that the coordinates *x* and *y* are in the range 0<= *x, y*<= 2000. Finally, you can assume that all input provided is valid and contains no errors. Therefore, there is no need for input validation.

Output

The output should begin with your Andrew Id. For each test case, write out a sequence of three numbers comprising a unique fixation point identification number, the x coordinate, and the y coordinate. Write each set of these three numbers on a separate line. The output should not include the terminating fixation point (i.e. the one with x and y coordinate of minus one) and there should be no duplicate fixation points, i.e. fixation points with the same coordinates. Note that the identification numbers should be contiguous, beginning with 1. Furthermore, the fixation points should be listed in ascending order by identification number. Finally, each list should be terminated by a line of dashes.

Sample Input

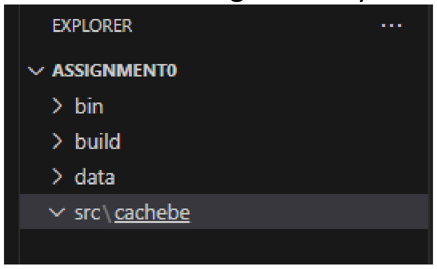
```
2
1 382 353
2 484 328
3 695 241
4 715 242
5 695 241
6 695 241
7 710 245
8 715 242
9 695 241
10 -1 -1
1 0 0
2 1 1
3 2 2
4 1 1
5 0 0
6 4 4
7 -1 -1
```

Sample Output

```
cachebe
1 382 353
2 484 328
3 695 241
4 715 242
5 710 245
**********
1 0 0
2 1 1
3 2 2
4 4 4
**********
```

Instructions

The input should be read from a file **input.txt** and output should be written to a file **output.txt**. Both files are to be located in the data directory. For an assignment, say assignment0, you should have the following directory structure. Assume cachebe is your Andrew ID:



Submit the following in a zip file named with your **Andrew ID** by the deadline shown above.
1. The source code: *.c, *.cpp, and *.h file(s)
2. The cmake file: CMakeLists.txt
3. The test input file: input.txt
4. The test output file: output.txt
Do not include any other files. Submit only the source code files, the CMake file, and the input & output files. Do not include subdirectories.

The source code should contain adequate internal documentation in the form of comments. Internal documentation should include the following.

- Author of the program.
- A summary of the way in which the code was tested.
- Functionality of the program.
- Format of input and output to the program.
- Solution strategy: a summary of the algorithm, e.g. using pseudo-code.
- Test strategy, i.e., description of the different test cases.
- Complexity analysis of the essence of the algorithm.

Place this documentation at the beginning of the application file.

Grading

Functionality: 70 points.

Does the program produce the same output as the instructor test output for the unseen test cases? Zero marks will be assigned for any test case that does not produce the required output. If the program does not compile and the marker cannot fix the problem in less than five minutes of effort, a total mark of zero will be assigned.

Test Cases: 30 points.

Does the student's test data in the file input.txt contain the following?

- Tests of greater strength than the two examples provided above. [15 Points]
- Tests that check boundary cases, e.g. several instances of the same fixation point. [15 Points]

Internal Documentation.

Marks for functionality and test cases will be awarded if and only if acceptable internal documentation is included. What constitutes acceptability is up to the examiner, but any reasonable attempt will be considered acceptable. Simple one-line descriptions will not be considered acceptable. Therefore, getting zero is plausible.

Use of ChatGPT

The use of generative AI e.g., ChatGPT is prohibited for this assignment. Therefore, any use of generative AI even to generate internal documentation will be considered unauthorized use and will lead to instant penalties as per Academic Integrity Violation (AIV) guidelines.