**Analyzing Visual Scan Paths of Professionals and Novices using Levenshtein Distance**

Zach Belou, Justin Clemmons, Rebecca Gravois, Norwood Hingle, and Jessica Wojtkiewicz

Louisiana State University

MATH 4020

April 30, 2015

**Introduction**

The purpose of this paper was to discover the effects of global and local clutter on visual scan search paths on navy pilot maps. A target icon was placed on a radar map that participants were asked to find. Two target groups participated in the experiment: professional pilots and undergraduate students with no experience of scanning naval pilot maps. The purpose of the experiment was to try to find similarities and differences among the professionals and novices when it came to scanning for the target icon. As soon as the participant located the icon on the computer screen, the participant clicked on the icon to show accuracy. This paper will explain the procedure and mathematics behind the different scan path techniques when it comes to locating the icon on the map.

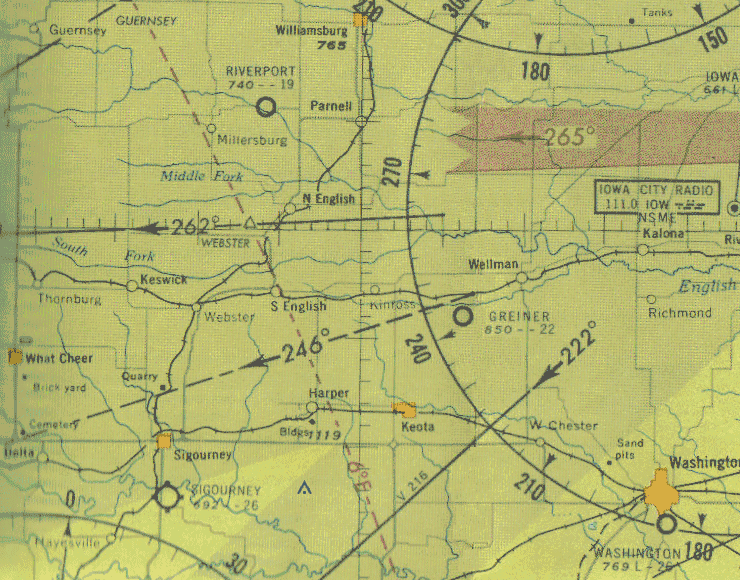


Figure 1: Example of a naval map

The two main characteristics of the naval maps used in the experiment, such as the one in Figure 1 above, were the level of local and global clutter. The local clutter was either high or low, or the target icon was missing from the map, which indicated no local clutter. Participants could select that an icon was not present in this case. Global clutter, or the amount of clutter in the entire map, was classified as low, medium or high. Each participant participated in 72 trials of the experiment with 72 different maps. Each trial had a 60-second time limit in which the participant could correctly locate the icon or correctly say that the icon was absent. The correctness of the participants’ selection as well as the visual scan path was recorded for each trial.

By nature of the experiment, participants were encouraged to find the target icon as quickly and as accurately as possible. The 60-second time limit acted as the incentive to make sure participants scanned quickly and tried to avoid being timed out. The variables of interest when looking at the scan paths of the participants were the number of fixations, the fixation length, and the distance between the fixation points. A fixation is simply when the participant lingered their eyes on one point of the map. The eye-tracking device was able to pinpoint exactly where each fixation took place and the path the eyes made when scanning from one fixation point to the next. The overall goal of the research is to see the difference between experts and novices in their visual scan path, as characterized by the fixation points. By retracing the fixation points in chronological order, the eye-path software can retrace the participants’ full scan path in real time.

Another variable of interest was the saccade length of the fixations. By plotting the visual map on a coordinate plane, the saccade length can be defined as the distance between any two consecutive fixations. Plotting the naval maps the participants looked at for the experiment onto a coordinate plane allowed us to divide the maps into grids based on the x and y-coordinates of the fixations. Plotting each fixation into a specific grid square allowed us to turn the visual map of sequential fixations into a character string for each scan path, comprised of the character assigned to each square in the grid. Then, by comparing character strings between trials, we were able to find the similarities between any two scan paths. This will be expanded upon in the Levenshtein Distance section starting on page seven.

This report will explain the data collected from the experiment, the conversion of the fixation locations into character strings, and the mathematical distancing algorithms used to compare the scan paths of different participants. Our final product will be a graphical user interface to filter different specifications of clutter and correctness for pilots and novices to compare fixation statistics and string similarities. The goal is to simply explain the process by which the final interface was created and explain the mathematics behind the string comparison.

The idea for this project came from Dr. Melissa Beck, from the LSU Psychology Department, and her experiment on comparing visual scan paths. Dr. Beck and her group tested pilots and undergraduate students’ ability to find a target icon in low, medium, and high global clutter as quickly as possible. Eye-tracking software tracked the scan paths of the participants as they tried to find the target before being timed out. The scan paths were formed from strings of consecutive fixation points, where participants’ eyes lingered for a few moments. The data provided by Dr. Beck was in the form of a spreadsheet with each row being one fixation. Data was recorded first by individual participant, then by trial number, then by fixation point. Our goalwith this project was to find a way to compare scan paths and create a user interface to compare the paths. Our methodology to find the scan paths was to convert the consecutive fixation points into character strings. This process is outlined in the Path-String Converter section beginning on page six. After converting scan paths to character strings, we used Levenshtein distance to compute the similarities between paths. This information, along with basic summary statistics on participant scan behavior is included in our graphical user interface. The interface allows users to specify which scan paths they would like to compare based on the level of local clutter, global clutter, and correctness.

**Data Structure**

We originally received the data from Dr. Beck in a large Excel file. This file contained numerous amounts of data including: a unique identifier for each pilot, trial count, global clutter, local clutter, trial fixation total, current fixation index, current fixation duration, start and end time of the current fixation, the x and y-coordinates of the current fixation, current map, reaction time, trial index, correctness, start and end time of the trial, saccade length, duration of the next fixation, and the x and y-coordinates of the next fixation.

Our first task was to determine the best way to import the given data into MATLAB. We decided to store it in a nested structure of arrays. In the top structure, we have an array of pilots, where each new element indicates a new participant. This structure contains the name of the pilot and an array of trials. Inside of the pilot structure is an array of trials, where each element indicates a new trial. This structure contains the information about the trial as described in the paragraph above as well as an array of fixations. Inside of the trial structure is an array of fixations, where each element indicates a new fixation point on the map. This structure contains the information about the fixation as described in the previous paragraph. Having the data in this intuitive format allowed us to access it more easily when it came to computing the statistics for the GUI. After inputting this data into the structure, we were then able to start manipulating it.

**Path-String Converter**

Before we could compare two fixation paths, we needed a way to describe the path. We chose to describe this path in terms of a string of characters. First, we overlaid the map with a grid. The area of the cells was determined by how close the fixations were to one another, based on their x and y-coordinates. Next, we assigned a character to each cell. Then, we were able to create a string by following the fixation path. We started with an empty string, and wherever a fixation point occurred, we appended the character for that cell to the string.



Figure 2: Example of path-to-string conversion

In Figure 2 above, we have a sample map overlaid with a 3x3 grid and a fixation path. The cells of the grid are numbered (x, y), where x indicates the row and y indicates the column, starting from the top left cell. To generate a string from this path, we first start with an empty string. The first fixation occurs in cell (3, 1), so we add the character ‘G’ to our string. The next fixation occurs in cell (2, 2). The character for this cell is ‘E,’ so our string becomes “GE.” It does not matter that we crossed into cell (2, 1) to get to this fixation. We only add a character to the string when we reach a fixation point. Finally, we found the target in cell (1, 2), so we append ‘B’ to our string. Now, we have our complete path string, which is “GEB.” We used this method to create a string for every trial. Now, we were ready to compare the paths.

**Levenshtein Distance**

The Levenshtein distance is what we used to calculate the distance between strings in our project. This helped us determine exactly how different or similar two strings were when comparing pilots to novices. This theorem is named after its developer, Vladimir Levenshtein, in 1965. It is one theorem amongst others used to calculate edit distance. It is typically applied in approximate string matching, which can be used in applications such as spell check. We also see this frequently in DNA sequence alignment theorems. The theorem also works well for our project.

The distance is computed by creating a matrix where two different strings are set as m and n. We set this up and computed the numbers in the matrix based on three operations: insertion, deletion, and substitution. Insertion and deletion have a cost of one, and substitution has a cost of two. As we went down each row, column, or diagonal, we added or subtracted the cost value if the coordinate of that location on the matrix did not match. If it did match, however, then we added zero. We chose the minimum value from the top, bottom, and diagonal and used that as the value of the coordinate, eventually building towards the last matrix coordinate, which is the distance between strings. This distance gave us an idea of exactly just how different the strings were from each other.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | E | X | E | C | U | T | I | O | N |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| I | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 6 | 7 | 8 |
| N | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 7 |
| T | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 8 | 9 | 8 |
| E | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 9 |
| N | 5 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 10 |
| T | 6 | 5 | 6 | 7 | 8 | 9 | 8 | 9 | 10 | 11 |
| I | 7 | 6 | 7 | 8 | 9 | 10 | 9 | 8 | 9 | 10 |
| O | 8 | 7 | 8 | 9 | 10 | 11 | 10 | 9 | 8 | 9 |
| N | 9 | 8 | 9 | 10 | 11 | 12 | 11 | 10 | 9 | **8** |

Figure 3: Example of Levenshtein distance between two words

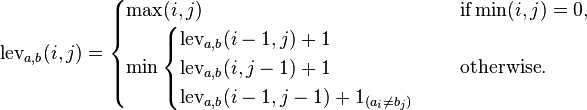


Figure 4: Algorithm for Levenshtein distance

We give an example of using Levenshtein distance to compare two strings in Figure 3 above. In this case, we are comparing “EXECUTION” with “INTENTION.” Using the algorithm depicted in Figure 4, we are able to complete the table. We can also see, through a trace back, the similar parts in the string, based on the values of the diagonals.

**GUI**

In addition to creating a structure within MATLAB for the data given to us by Dr. Beck, our group created a Graphical User Interface (GUI) in MATLAB to portray the work that we completed throughout the semester. Our GUI organizes the data and presents the information in a visually appealing and easy-to-understand manner, which allows a user to see specific trials completed during an experiment without hassling with a complicated spreadsheet in Excel. In our GUI, there are three main parts: data, filter, and Levenshtein distance distribution. Figure 5 below is a screenshot of the GUI that we created.

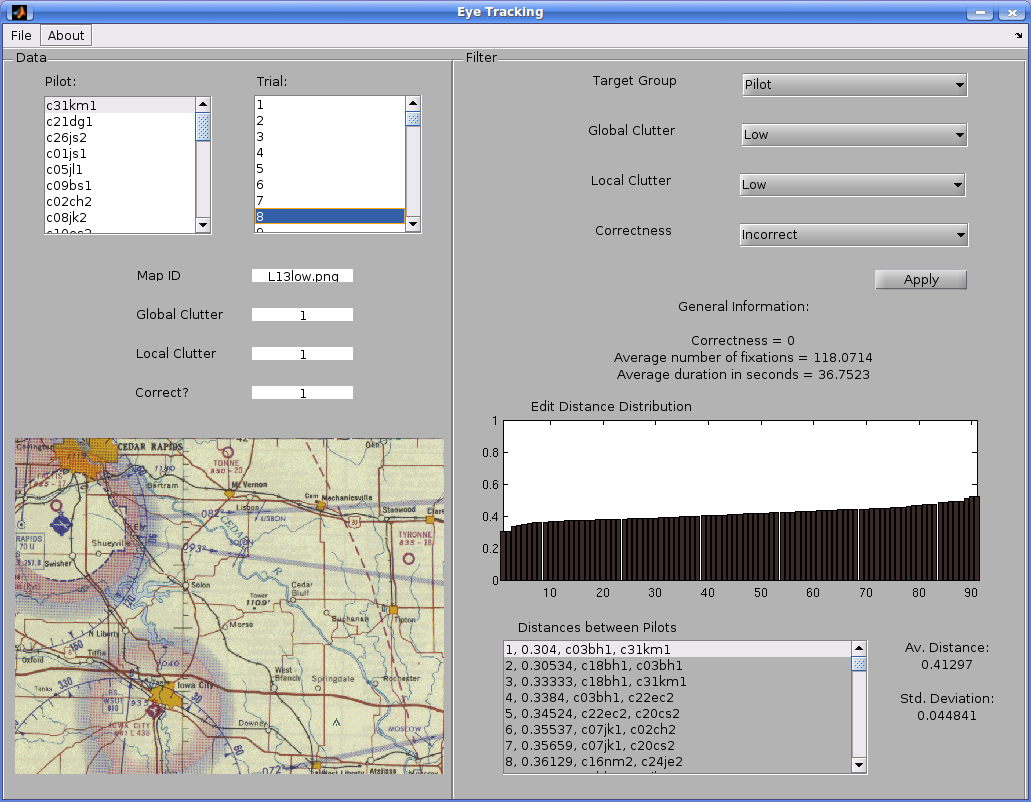


Figure 5: Screenshot of the GUI

On the left side of the GUI is a column that directly correlates to the spreadsheet given to us by Dr. Beck. The user must click on one of the 12 pilots in the first list box as well as one of the 72 trials completed by the chosen pilot in the second list box. Once the user does this, the map shown during the chosen trial, the amount of global and local clutter, and whether the pilot was correct or not is shown on the left column of the GUI. For global clutter, the value one corresponds to low global clutter, two to medium global clutter, and three to high global clutter. For local clutter, the value zero corresponds to an absent target, one to low local clutter, and two to high local clutter. For correctness, the value one means the pilot was correct during the trial chosen, and zero means the pilot was not correct.

On the right side of the GUI, the user is able to apply a filter to the data within the structure by selecting the amount of global clutter, amount of local clutter, and correctness of the trials. The user may choose any combination of global and local clutter and correctness, including not applying a filter to the individual parameters. After applying the filter chosen by the user, the percent of correctness, the average number of fixations, and the average duration in seconds is shown on the interface. The Levenshtein distance distribution and the distances between all pairwise combinations of pilots that have passed the filter are also shown in the axes and list box on the bottom-right side of the GUI. The average distance and the standard deviation between all combinations of the pilots listed are also displayed.

Overall, our GUI allows a user to view the data collected by all experiments conducted by Dr. Beck in a way that is easy to understand. This GUI will be especially helpful whenever a user wants to know specific information about a certain trial completed by a pilot or a group of pilots based on their performance on various trials.

**Conclusion**

During our project, we were successfully able to read the fixation data into a structural array, convert the fixation paths of each trial into character strings based on the grid pattern specified, use Levenshtein distance to compare two scan paths to each other, and create a graphical user interface to display our findings. The structure of the array that now reads the data will make statistical analysis easier in the future. The Levenshtein distance is a promising lead to compare scan paths, but other distance formulas can be looked into in order to compare a greater number of paths at once. Our project focused on reading the data and being able to present the data in an easy-to-understand format, but future work can focus on fixation duration, correctness, and saccade length. By transforming the scan paths into character strings, the data has been converted into a form similar to DNA sequences. It might be useful to look up methods used for DNA comparison and categorization. This will be worth looking into in future work on this experiment.

In order to make the data easier to read, the intuitive importation of the data from a spreadsheet into an array structure made for simple variable calculation. The setup of the array structure made it easy to define where to search for information, whether we wanted to look at all fixations for one person, for one trial, or each fixation separately. As described previously, summary statistics can be calculated within each trial by specifying for one person, then all trials within that one person’s array. The filter on the graphical user interface allows statistics to be read directly from the multiple arrays depending on the specification.

As far as preliminary results for the experiment are concerned, the major determining factor for correctness accuracy was the level of global clutter. As the level of global clutter increased, the pilots separated themselves from the novices by more accurate scores. There is no difference in the level of correctness for low and medium levels of global clutter. Interestingly, the pilots timed out more often than the undergraduates when searching for the target icon. This is most likely due to undergraduates trying to answer as quickly as possible, as opposed to pilots trying to systematically find the icon, preferring correctness to speed.

As one would expect, the intensity of global clutter directly corresponded to the number of fixations it took to find the target icon. The contrast between a map with low global clutter and a map with high global clutter is shown in Figure 6 below. With more fixations to look for the icon, it followed that higher levels of global clutter led to more incorrect answers and more time outs. Pilots had longer fixation lengths, meaning that pilots lingered on each spot longer when searching for the target icon. One explanation is that pilots take the job of reading the map seriously and want to be sure that the target is not in a given space when searching. Undergraduates, on the other hand, will scan in a less systematic and more intense way as to spend little time on each fixation if the target icon is not directly in their line of vision.

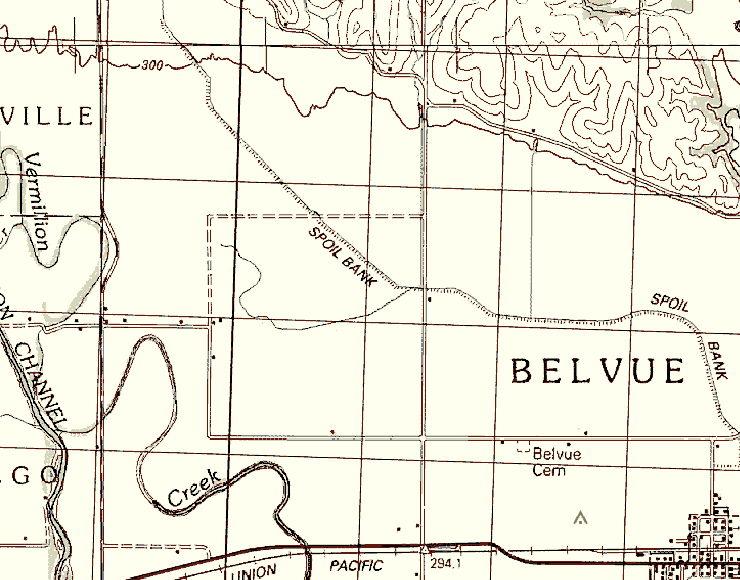
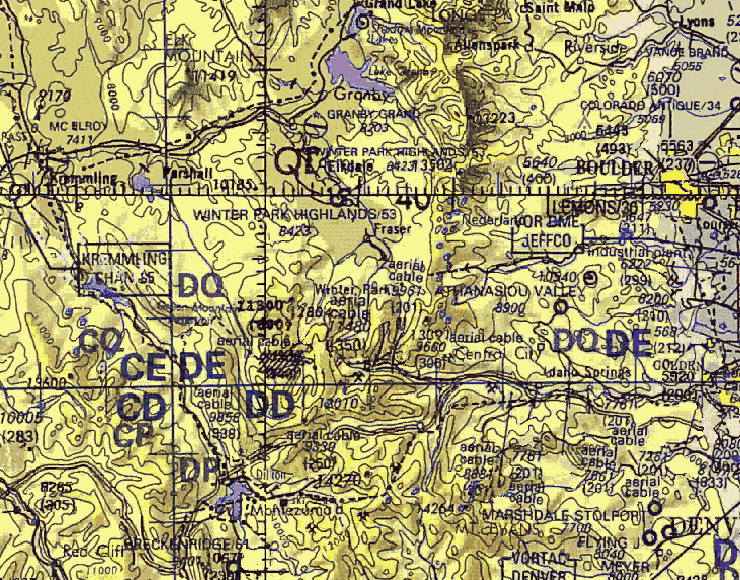
 

Figure 6: Low global clutter vs. high global clutter map

Once character strings were established for the scan paths, we were able to compare the different paths as previously described. Our work comprised mostly of devising the algorithm for the string comparison. We were not able to compare the distance between two or more scan paths due to time constraints, but this will be the case of study in the future to explore.

Future work on this project will involve comparing scan path strings between more participants and trying to calculate more advanced statistics on saccade length and fixation duration. By writing a categorization algorithm, we may be able to adjust the filter and find the relationship between differing trials. Much like the sequencing of DNA chains, which compare strings of DNA code between individuals looking for matches, character strings of visual scan paths can be run through matching algorithms to try to find the similarities of paths. This is just one way to compare the paths in the future work on this topic.