AFRL-RH-WP-TR-2018-0036

Quantifying Eye Movement Trajectory Similarity for Use in Human Performance Experiments in Intelligence, Surveillance, and Reconnaissance (ISR) Research

Dr. Mary E. Frame
Wright State Research Institute
4035 Colonel Glenn Hwy,
Beavercreek, OH 45431

JUNE 2018

Interim Report

AIR FORCE RESEARCH LABORATORY
711TH HUMAN PERFORMANCE WING
AIRMAN SYSTEMS DIRECTORATE
HUMAN-CENTERED ISR DIVISION
HUMAN ANALYST AUGMENTATION BRANCH
WRIGHT-PATTERSON AFB OH 45433
AIR FORCE MATERIAL COMMAND
UNITES STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th Air Base Wing Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RH-WP-TR-2018-0036 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

OCAMPO.SABRINA.M.1185673517
Digitally signed by OCAMPO.SABRINA.M.1185673517
Date: 2018.07.31 13:43:01 -04'00'

SABRINA M. OCAMPO, Work Unit Manager
Human Analyst Augmentation Branch
Airman Systems Directorate
711th Human Performance Wing
Air Force Research Laboratory

DOVER.LANCE.N.1151444926
Digitally signed by DOVER.LANCE.N.1151444926
Date: 2018.07.27 15:11:16 -04'00'

LANCE N. DOVER, DR-III, DAF
Chief, Human-Centered ISR Division
Airman Systems Directorate
711th Human Performance Wing
Air Force Research Laboratory

This report is published in the interest of scientific and technical information exchange and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved OMB N 0704 0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
19–06–2018	Interim	17 Oct 2016 – 19 Jun 2018

4. TITLE AND SUBTITLE

Quantifying Eye Movement Trajectory Similarity for Use in Human Performance Experiments in Intelligence, Surveillance, and Reconnaissance (ISR) Research

5a. CONTRACT NUMBER

FA8650-15-D-6583

5b. GRANT NUMBER

18RHCOR087

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Mary E. Frame

5d. PROJECT NUMBER

5e. TASK NUMBER

0002

5f. WORK UNIT NUMBER

H0L5

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Wright State Research Institute
4035 Colonel Glenn Hwy,
Beavercreek, OH 45431

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Material Command
Air Force Research Laboratory
711th Human Performance Wing
Airman Systems Directorate
Human-Centered ISR Division
Wright-Patterson AFB, OH 45433

10. SPONSOR/MONITOR'S ACRONYM(S)

711 HPW/RHXM

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AFRL-RH-WP-TR-2018-0036

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION A: Approved for public release.

13. SUPPLEMENTARY NOTES

Report contains color. PA Clearance#: 88ABW-2018-3124, cleared 19 June 2018

14. ABSTRACT

The purpose of this technical report is to serve as an introduction to ScanMatch, a Matlab package used to determine the similarity of two strings of eye tracking data. This report provides a background and recommendations for potential applications in Intelligence, Surveillance, and Reconnaissance research. Additionally, this document serves as a tutorial for using this software, providing a detailed description of the software implementation procedures. Finally, this report includes a demonstration of the functionality and flexibility of this software using pseudo-data with predictable results, and concludes with descriptive and statistical analyses on real experimental surveillance task data.

15. SUBJECT TERMS

Statistical analyses. Intelligence, Surveillance, and Reconnaissance.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Sabrina Ocampo
U	U	U	SAR	60	19b. TELEPHONE NUMBER *(include area code)* N/A

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18

Abstract

The purpose of this technical report is to serve as an introduction to ScanMatch, a Matlab package used to determine the similarity of two strings of eye tracking data. This report provides a background and recommendations for potential applications in Intelligence, Surveillance, and Reconnaissance research. Additionally, this document serves as a tutorial for using this software, providing a detailed description of the software implementation procedures. Finally, this report includes a demonstration of the functionality and flexibility of this software using pseudo-data with predictable results, and concludes with descriptive and statistical analyses on real experimental surveillance task data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

The performance of persons who watch surveillance videos, either in real-time or recordings, can vary with their level of expertise. It is reasonable to suppose that some of the performance differences might be due, at least in part, to the way experts scan a visual scene versus the way novices might scan the same scene. For example, experts might be more systematic or efficient in the way they scan a scene compared to novices. Even within the same person, video surveillance performance can vary with factors such as fatigue. Again, differences, in the way their eyes scan a scene might account for some of the differences.

Full Motion Video (FMV) "Eyes-on" intelligences analysts, in particular, actively scan video scenes for items of interest for long periods of time.

To better understand the characteristics of scanning behavior of Intelligence, Surveillance, and Reconnaissance (ISR) analysts, it is important to track eye movement characteristics. It is relatively simple to model eye characteristics over time such as pupil dilation or eyelid opening over time. It is also common to characterize different types of eye movement over time. However, when it comes to making comparisons between two sequences of eye data, it is much more challenging and by default, most commercial eye tracking systems do not come equipped with analytic measures of comparing trajectory morphologies. However, there are a variety of applications where this comparison can be useful to ISR research:

- Comparing eye scanning movements before and after critical events, such as detecting a target
- Comparing scanpaths between two different analysts
- Comparing scanpaths of analysts over time (e.g. early in the day vs. later in an analyst's shift)
- Comparing scanpaths across different surveillance tasks
- Comparing scanpaths in simple versus complex surveillance scenarios

This technical report explores some common metrics for quantifying the similarity between two eye movement scanpaths, with an emphasis on the Matlab toolbox ScanMatch. ScanMatch is a Matlab package that computes a similarity score between two scanpaths. This technical report will include a detailed tutorial for using ScanMatch and will present data that was analyzed using ScanMatch. Both experiments involving pseudodata with known conclusions and experimental data from Piaseki (2016) were analyzed using ScanMatch.

2.0 DEVELOPING A SIMILARITY METRIC FOR EYE SCANPATH DATA

In research on human analyst performance in Intelligence, Surveillance, and Reconnaissance (ISR) tasks, it is useful to collect eye tracking metrics of analysts as they perform search operations. In particular, information about fixations and saccades (i.e., the jumps from one fixation to another), has been useful for yielding information regarding:

- Workload (e.g., Sirevaag, et al., 1999; Stasi, et al., 2010; Schleicher, et al., 2008, Backs & Walrath, 1992),
- Fatigue (e.g., Morris & Miller, 1996; Sirevaag, et al., 1999; De Gennaro, et al., 2000; LeDuc, et al., 2005),
- Attention (Duchowski, 2007; Tsai et al., 2012), and even
- Inattention blindness (Drew, Vo, Wolfe, 2013).

Fixation and saccade locations can serve as markers for attention since where an analyst is looking on the screen is often highly correlated with what he or she is attending to. However, highly correlated does not mean perfectly correlated with conscious attention, as the well-known inattention blindness study by Drew, Vo, and Wolfe (2013) demonstrates. In their study, radiologists examining an X-ray often fixated a gorilla figure embedded in the X-ray and made repeated backtracks or re-fixations to it but did not consciously notice the anomaly. Inattention blindness and associated re-fixation and backtrack patterns are important in understanding ISR analyst performance.

Scanpaths. Among the rich variables that are collected in eye tracking data, one which centers on the temporal aspects of fixation patterns but whose analysis is frequently neglected due to its complexity, is the scanpath. A scanpath is defined by the temporal sequence of point-by-point (x,y) screen coordinates of where a person is looking on the screen. Figure 1 shows three example notional scanpaths although the temporal direction is not shown. At a minimum, scanpaths encompass at least one full fixation-saccade-fixation sequence (Poole & Ball, 2006). Clearly, scanpaths capture the fixation, re-fixation, and backtrack patterns revealing of analysts attention, conscious or otherwise.

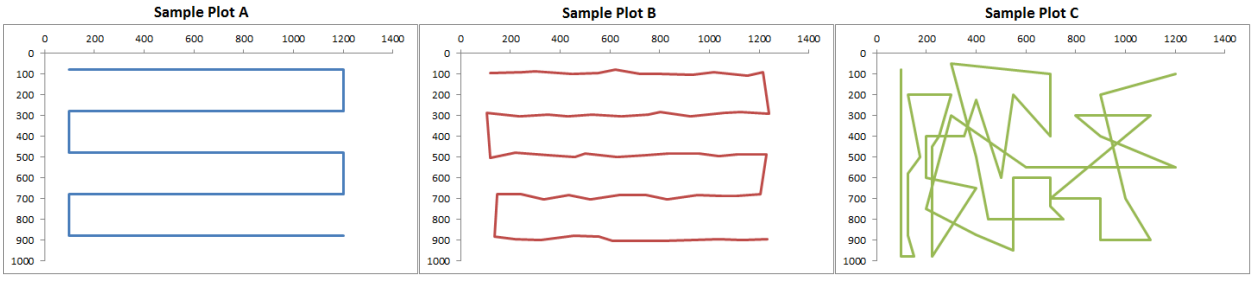

Figure 1: Three sample plots of scanpath data. The leftmost image (Plot A) is highly similar to the scanpath of Plot B, but is highly dissimilar from the scanpath of Plot C. All 3 plots are made with an identical number of raw gaze points.

2.1 Assessing the Similarity and Differences of Different Scanpaths

Experts search visual displays both differently and more efficiently than novices. For example, Plot B in Figure 1 is very systematic and might hypothetically be characteristic of an expert whereas Plot C has a helter-skelter quality and might, again hypothetically, be characteristic of a novice's search.

Thus, as revealing as the analysis of a single scanpath might be, we also want to be able to compare the similarities and differences of search scanpaths:

- Between two experts,
- Within one analyst but between search tasks of differing difficulty,
- Between a novice and an expert, and
- Between different sub-scanpaths within one search task.

In the Figure 1 optimized examples, Plot A and Plot B look extremely similar in overall morphology, while Plot C looks very different from both Plot A and Plot B. Although this is a highly contrived example, where differences are fairly easy to estimate, real eye data is often much noisier and it is often impossible to accurately quantify the degree of difference via visual inspection alone.

Since it would be useful for a wide variety of tasks to have a means of quantitatively comparing the similarity or difference of two strings of eye-tracking scanpath data, the goal of this technical report is threefold:

- To present sequence-alignment metrics broadly and explain the theory of how sequence alignment works to quantify the degree of difference in two strings of data
- To present a tutorial on ScanMatch, a Matlab toolbox specifically designed to perform sequence alignment on eye tracking scanpath data.
- To illustrate the use of ScanMatch on scanpath data collected during a complex ISR search task.

2.2 Terminological and Data Notes

Before proceeding to how sequence alignment can be used to assess the similarity of strings of data, it is important to be clear about the content of the data strings. Data factors include:

- Terminology: Fixations & saccades as data points
- Data file structure
- Data collected versus data analyzed
- Temporal resolution of the data
- Spatial resolution of the data
- Meaning-based functionally equivalent locations

2.2.1. Terminology: Fixations & Saccades as Data Points

"Fixation" and "saccade" have, so far, been used intuitively. Fixations, per se, and the apparatus and methods used to record them are complex since the underlying eye movements are very complex (e.g., Holmqvist & Nystrom, 2011). The reason is that the eyes are in constant micro-motion even during a presumed fixation. The constant motion (e.g., ocular tremor and microsaccades) can be of a very high frequency. Generally, eye tracker classification algorithms can vary greatly and many have user-set parameterizations for what is classified as a fixation versus saccade or if other types of movement are classified (i.e., glissades and smooth pursuit). Most typically classify saccades using a static velocity or distance between-samples threshold and classify movements that fall under this threshold as a fixation.

2.2.2. Data File (Minimal) Structure

The data file for a single scanpath consists of a minimum of three columns: one for the *x*-location, one for the *y*-location, and one for the timestamp of a fixation. Other columns might be included to encode trial conditions. Each row records the information for one fixation along the scanpath. Table 1 illustrates how these coordinates and timestamps would appear for a 1280x1024 monitor with a sampling rate of 60 Hz. Although this table shows four rows, on a real data set there would be 60 rows for each second's worth of data, making the data set for a real scanpath much longer and more variable. In fact, some eye trackers have a much higher sampling rate and thus would produce even longer data files. Depending on the interests of the researcher, the data may contain gaps if saccade movements are removed. However, for examining a scanpath, one can typically import the entire data set, regardless of fixation or saccade or other movement type classification.

Table 1: Sample Formatting Required By Most Sequence Alignment Algorithms

X Coordinate	Y Coordinate	Time (ms)
630	400	16.67
600	405	33.33
605	405	50
610	420	66.67

2.2.3. Data Collected Versus Data Analyzed

Although similar in structure, the data files that are output by an eye tracking system are not necessarily the data files that are used for a similarity assessment. The raw data output by an eye tracking system must generally be pre-processed or cleaned prior to loading into typical sequence alignment algorithms including ScanMatch. Typically *x*- and *y*- coordinates as collected by eye-tracking systems may need to be converted into pixel coordinates from decimal

coordinates (calculated as a percentage of screen coverage from the eye-tracker's origin corner) depending on the comparison algorithm. For example, the middle screen point designated (.5, .5) on a 1280x1024 screen would need to be converted to (640, 508) pixels for certain algorithms. If time is required by a comparison algorithm it will typically need to be formatted in milliseconds or seconds if that is not the default timestamp output by the eye tracker. More general temporal and spatial data differences are discussed below after considering the data-to-be-analyzed file structure.

2.2.4. Temporal Resolution of the Sequence Data

The length, i.e., the number of rows, of a "first level" eye tracker data file is determined by the duration of a scanning session and the sampling rate of the eye tracking equipment.

Further, since the multiple ocular tremor-produced micro-fixations can be considered as just noise around an, at least, "representative" fixation or fixation-of-interest, eye-movement recorders typically pre-process high sampling-rate input in order to output data files which filter out the micro-fixation noise. Thus, the recorded fixations or "fixations of record" can be many fewer than actually made by a person but which are still a reasonably accurate, albeit a spatial average, record of where a person was looking within a very short time interval. Of course, researchers can down-sample their files for a coarser temporal resolution depending on their needs or interests.

2.2.5. Spatial Resolution & Functionally Equivalent Image Locations

The number of "representative" fixations in a scanpath is not a problem for analysis programs. Nor is the spatial resolution of the display or image being searched a problem for computation. However, there can be a conceptual problem with determining similarity in an image with too great a pixel granularity or spatial resolution of the *x*- and *y*-coordinates.

The functional question here is determining how close two fixations, or a cluster of fixations, must be in order to be "at the same location" or be at functionally equivalent locations. This issue is important in order to determine re-fixations since a return to a prior fixation "point" or image feature may be indicative of interest by an analyst even if there is no conscious or other overt indication of interest.

It would be possible to define two close points being "at the same location" using a distance threshold[1] or mathematical neighborhood criterion. Under such a definition, pixel granularity is not a problem and, in fact, the higher the granularity, the more precise the determination of closeness can be. However, the predominant approach to determining functionally equivalent locations is to use a relatively course grid or partitioning of an image. For example, a 1280 x

[1] In Section 1.2.1, a distance threshold criterion was discussed for differentiating fixations from saccades. Here a distance threshold is used for deciding when two fixation *locations* are close enough so that the two fixations are considered to be at the essentially same location.

1024 pixel image may be reduced to a 26 x 26 grid of functionally equivalent locations. This reduces the number of functionally equivalent locations from 1,310,720 to a mere 672 locations.

How coarse a grid can a grid be? It depends on the interest and purpose of the user of the data. Consider some extreme examples: It might be sufficient for some purpose to simply determine whether the left or right side of an image is viewed, i.e., fixated, more. Not just the quantity of fixations but of the saccades can be of interest if one is studying a person viewing a tennis match from the viewpoint of a line judge. Or, the interest might be in just the fixations and transitions from the top of a screen to the bottom as in the view of a tennis match that is usual in television.

This coarse-grid approach is discussed in more detail in the sections that follow and the ScanMatch software examples.

2.2.6. Region of Interest (ROI): Meaning-Based Functionally Equivalent Locations

As just discussed, the use of a course grid can vastly reduce the number of functionally equivalent locations. However, it does so with total disregard to any inherent meaning or predefined region(s) of interest (ROI) among the elements comprising the image. But a marketer might want to know which products are most viewed and which are ignored irrespective of where they appear in an image of a store shelf. Or a researcher might want to know if certain people are more attended to than others. Regions of interest can be specified in eye-tracking analysis software, and the ROI's need not conform to a simple rectangular array of image pixels. Within an ROI, all fixations are functionally equivalent despite their actual locations. This is further discussed in the following sections and specifically in the discussion of the ScanMatch software.

2.3 Other Techniques for Analyzing Eye-Tracking Data

Scanpath analysis does not exhaust all the information in eye-tracking data. Other ways to further analyze the data include:

- Heatmaps
- Time series techniques
- Scanpath distance metrics

2.3.1. Heatmaps: Gaining Information by Dropping Time Stamps

The inclusion of the timestamps in the data files is crucial to the comparison metrics which are the focus of this report. Oddly, the inclusion of the timestamps can obscure other aspects of the fixations patterns which can better be revealed using other techniques such as heatmaps (also known as intensity maps) which discard the temporal aspects when they are not pertinent for analysis or visualization. Figure 2 provides an example of a heat map from Djamasbi, Siegel, & Tullis (2011). Green coloring indicates that participants looked at a location infrequently while redder coloring indicates that participants looked at a particular location for a longer duration.

Such non-temporal patterns and information will be the subject of a subsequent report.

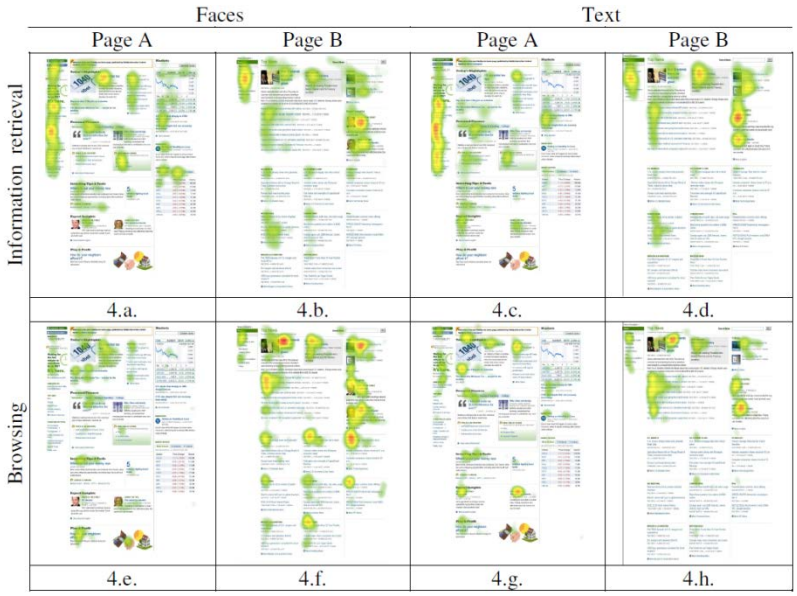

Figure 2: Sample of multiple heatmaps from Djamasbi, et al. (2011) of web viewing under various conditions.

2.3.2. Time Series Analysis Techniques

Due to the volume of data in two scanpath fixation time series, it is not intuitive to compare them by visual inspection alone. Further, two scanpath fixation sequences generally cannot be accurately compared by traditional statistical metrics such as by using t-tests or Analysis of Variance since, as time series, values are highly autocorrelated from one time point to another. Although this technical report will not contain instructional information for performing these analyses, it is useful to take note of the special nature of eye tracking data as a time series compared to data where values are independent. Time series analysis metrics for use with eye tracking data will be explored in a subsequent technical report.

2.3.3. Scanpath Distance & Saccade-Based Metrics

Last, possibly simplest, but not necessarily least, are metrics based on scanpath distances such as total length, mean length, and standard deviation length. For example, it would not be unreasonable to assume or hypothesize that experts are more efficient than novices and that therefore the total length of the scanpaths of experts would be shorter than those of novices. Possibly also, cluttered or visually dense scenes might promote the use of generally shorter saccades during searches.

The point here is that eye movement data is very rich and potentially revealing of ISR analyst search behavior. The approach taken here is just one way to mine the rich data for gems.

3.0 RATIONALE/THEORY BEHIND SEQUENCE ALIGNMENT

Sequence alignment is a technique developed in biology that has been applied in a relatively limited context. In particular, this technique was developed to determine the degree of difference between two different protein structures or DNA sequences. However, the basic principles and techniques can easily be translated to analysis of eye tracking scanpath data. By converting the visual field or visual display into a grid structure, or alternatively if the task itself involves search through information arranged in a grid structure, or if the task contains specific Areas of Interest (AOIs), it is possible to make comparisons between two sequences of eye fixation locations as well as their transitions.

3.1 Types of Elements Comprising a Sequence in Eye Fixation Applications

Earlier, a distinction was made between micro-fixations and the more aggregative fixations recorded in an eye movement file. There is yet another level of aggregation needed before performing a sequence alignment.

The following subsections outline two popular approaches for performing sequence alignment on eye data. The first approach described is an ROI-centered approach. The strength of this approach is that similarity of eye tracking path data is characterized based on the important features or regions of interest (ROIs) associated with the visual displays[2]. This allows for important comparisons between eye path data when the displays are varied. Another advantage is that this method is simple to implement and interpret, producing a single similarity metric. Many programs using this method are also designed to handle more than just pairs of string data, meaning that they are capable of comparing 3 or more strings in regard to their similarity.

The second approach is a vector-based approach to comparing scanpath information. One of the strengths of this approach is that it is capable of producing a more resolute output, providing multiple metrics of similarity based on length, distance, etc. This can also be a better comparison metric if there is no clear hypothesis or any areas of particular interest of the displays (such as in a surveillance task with no targets or viewing a single image). One disadvantage of this approach is that it is agnostic to display characteristics and therefore cannot be used to characterize what observers were viewing in one condition versus another.

It is recommended therefore that each tool be applied when appropriate and that in many circumstances (such as in surveillance tasks involving known targets or known AOIs) it may be useful to collect both metrics of similarity for scanpath comparison.

3.2 Sequence Alignment Broadly Defined

[2] Occasionally, the terminology areas of interest (AOIs) will be used instead of ROIs, but these refer to the same concept. Both terms are used in the literature.

Sequence alignment methods are fairly straightforward in how they are performed, but specific weighting and calculations are highly flexible and specifiable in most programs. These methods are a specific form of string-edit distance measures which measure the similarity between two strings of values based on how many changes must be made from one sequence to another. Consider the following example of two different DNA protein strings:

String 1	A	T	C	C	G	A	T	C	G	T	T	A	T	A
String 2	A	T	G	C	A	T	C	G	A	T	T	A		

There are multiple ways of changing String 2 so that it matches String 1. One could simply change each element and add elements at the end, e.g., change the first G to a C, change the next different letter, A, to a G, etc. However, this is not necessarily optimal as it requires more than the minimum number of possible changes to make these strings identical. If one wanted to make fewer moves, one could simply add a space between the fourth and fifth letter of String 2 to make the A and T align with string one, like the following:

String 1	A	T	C	C	G	A	T	C	G	T	T	A	T	A
String 2	A	T	G	C	--	A	T	C	G	A	T	T	A	

Now with only a single move, four changes were made down the line so that the A, T, C, and G following the dash all align. Although a dash is optimal in this position, it would not be optimal in the position between the first T and G in String 2 since the G would still not align with String 1, making a substitution (denoted below in **bolded red**) more optimal as follows:

String 1	A	T	C	C	G	A	T	C	G	T	T	A	T	A
String 2	A	T	**C**	C	--	A	T	C	G	A	T	T	A	

One could continue making these changes, or more optimally have a computer try each permutation at a much faster pace, to eventually find the optimal pattern of replacements and dashes in both strings to make them match one another. Once this optimal pattern has been found, a metric can be computed to determine how similar the two strings are based on the number of changes. In some circumstances a researcher may prefer to vary the dissimilarity weight of dashes and substitutions so that they are not equal, but these circumstances will be discussed in greater detail later in this technical report.

3.3 Sequence Alignment in Eye Tracking Using ROI Sequences

One of the easiest ways to generate a string of eye tracking data that would be usable in a sequence alignment algorithm is to use coded regions of interest (ROIs). ROIs are generally defined as areas that are experimentally relevant on a screen or within an array of features. For example, in the following image of shampoos arranged on multiple shelves (Figure 3), an experimenter might be interested in which shelf observers focus on the most and the probability of transitioning to a certain shelf given that they already looked at another shelf. The

experimenter would therefore most likely define each shelf as an ROI. These ROIs can then each be given an arbitrary label. When examining the sequence of scanpath data, at each time point, the ROI the observer is looking in is recorded.

Figure 3: The image has 6 coded ROIs which are defined by each visible shelf ranging from the top shelf arbitrarily coded as A to the bottom shelf arbitrarily coded as F. At each time step of eye tracking data, the ROI the observers are examining is recorded.

In this market research example, once the experimenter collects eye tracking data from multiple observers, they may wish to compare how similar the scanpaths are between observers. To do this a similar operation would be performed as in the DNA string example.

Observer 1	A	B	B	B	C	D	D	C	F	B	B	B	A
Observer 2	B	B	B	D	D	A	A	B	B	B	A		

As with the earlier example, an algorithm would be used to calculate the fewest number of changes that would need to be made to the strings of data to make them identical to one another. In this case the optimal solution is:

Observer 1	A	B	B	B	C	D	D	C	F	B	B	B	A
Observer 2	--	B	B	B	**C**	D	**D**	**C**	--	B	B	B	A

Although in this shelving example all of the ROIs are arranged in an orderly fashion and are equal in size, when constructing ROIs they do not need to be equal in size or arranged in an orderly manner. We can extrapolate this principle to an ISR context by imagining a surveillance task where an analyst must keep track of 8 different buildings in different locations on the screen

and watch for possible terrorist activity, pressing a button to indicate at which building a suspicious activity is taking place (Figure 4). In this hypothetical task, the ROIs would likely be defined as the areas of the screen where the buildings are located and may not be of equal size or arranged in an orderly manner, but the manner for calculating differences between scanpaths would be identical to the method above. In this case, the strings of letters that are compared would be written similarly and the comparisons could be conducted identically. However, in an instance where the ROIs cover drastically different areas of the screen or are configured in a non-matrix format, a more sophisticated weighting metric might need to be used for determining distance. For example, in Figure 4 certain portions of building C are closer to building A than building B and vice versa. Additionally, as the image in Figure 4 illustrates, there may be a large portion of the screen that is not classified as any particular area of interest. In performing similarity characteristics, this might need to be specified as another AOI; otherwise this will be classified in both strings as missing data.

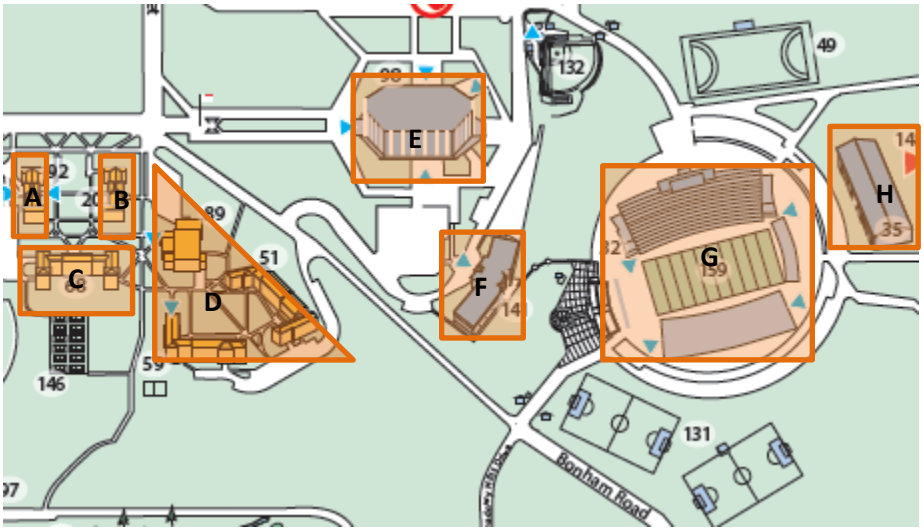

Figure 4: The image has 8 coded ROIs which are defined by visible buildings of interest (images from Miami University of Ohio campus map). These ROIs are coded with letters A-H. Note that ROIs can be of different sizes and shapes, but must not overlap in area coverage. Some areas are not classified in this example, but may be given an explicit code if desired.

3.4 Sequence Alignment in Eye Tracking Without Specific ROI Markers

Although it is useful to parse scanpaths based on meaningful interest markers such as ROIs, at times it may be the goal of the researcher to model where on the screen observers are looking, regardless of the screen content. For example, when comparing two trials or two observers, the research question of interest may simply be to determine how similar a search pattern is. Imagine a search task in which the goal is to find a hidden object in an array of similar objects. A

researcher may wish to know if observers search in a particular pattern or just if they search for the object in a manner similar to other observers.

To apply sequence alignment in tracking eye scanpaths with no concrete area of interest locations, first grid boundaries need to be established. Resolution of developed grids can vary as necessary and should be built to be flexible into the program creating the grid. However, since this is aligning transitions to different grid areas, it would be best to keep a fairly low resolution to the grid (as large of an area in each cell as possible) to account for measurement error in the eye tracker, which is virtually never pixel precise. Additionally, at too fine grain of a resolution, there may be more transitions than necessary (i.e., it may register as transitioning to another "area" but the observer is actually looking at the same pertinent feature of the scene, such as a building which covers multiple pixels of the screen). These grid areas need to be numbered or given a letter indicator for each grid cell, with each cell essentially serving as an experimenter-constructed ROI. For example, imagine the grid in Figure 5 below is a divided-up screen with a superimposed grid; the cell names are indicated with a letter.

A	**B**	**C**
D	**E**	**F**
G	**H**	**I**

Figure 5: Arbitrarily defined 3x3 grid partitioning a screen into 9 equal-sized ROIs.

3.5 Comparing Fixation Sequences Within a Trial

Although the previous examples have shown string comparisons either between trials or between observers, it is possible to make comparisons between two strings in the same trial based on particular behavioral or critical events, provided that there is no temporal overlap and shared time points in both strings. Take for example, an experiment where a critical event occurs such as an array of dots changing direction on every trial. An experimenter may want to identify how consistent an observer's scanpath is throughout the trial by comparing how they visually search the space before vs. after the image changes. Just as in the previous cases, two strings can be generated: one from before a critical event occurs and one from after the critical event has occurred and we want to compare if there the degree of difference in the scan pattern of the analyst before and after the critical event. Since there are no specified areas of interest in this hypothetical task, one could still make comparisons between the scanpaths by dividing the screen into the matrix in Figure 6.

Before Event	A	B	C	A	C	B	D	E	D	F	G	H	I	H	B	C	D	E
After Event	A	C	A	B	D	E	B	A	F	G	H	I	D	E				

Even though there are more samples collected before the event than after the event, we can still apply the algorithm to make the correction as well as make a visualization of the scanpath for both. As seen in this example, to make the fewest substitutions (which are usually penalized more than blank spaces) blanks can be added in the longer string as well. Figure 4 provides an illustration of both scanpaths. If a researcher is interested in overall similarity of the morphology alone without any penalty for differences in length (as one might be in data sets with multiple redundant samples like eye tracking data), there will likely be little to no penalty for the two sequences. However, if a researcher is interested in comparing difference with timing as a consideration, or realizes that by having more points the longer string may have multiple degrees of freedom to vary, then the two strings would likely be computed as being dissimilar due to the differences in string length. For example, in the sequences below, the 3-blank run could have any permutation of possible 3 letter combinations.

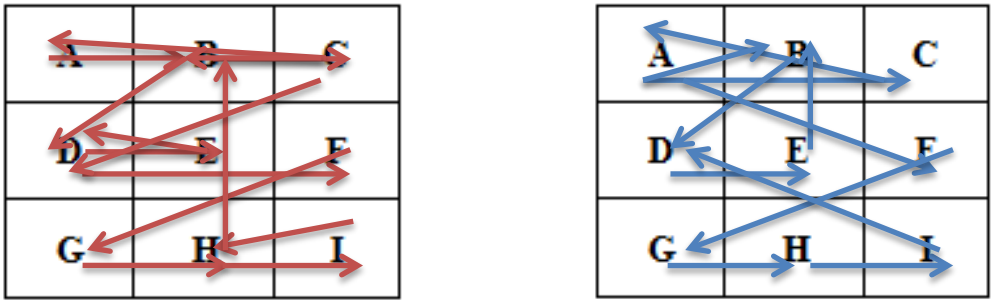

Figure 6: The left figure illustrates the first sequence scanpath (red arrows) and the right figure illustrates the scanpath after the event (blue arrows). Note that the morphology looks highly similar despite the first sequence having many more data points.

Before Event	A	B	C	A	C	B	D	E	D	--	F	G	H	I	H	B	C	D	E
After Event	A	--	C	A	--	B	D	E	**D**	A	F	G	H	I	--	--	--	D	E

3.6 Scoring & Replacement Types: Substitutions & Blank Space Insertions

Regardless of whether the sequences elements arise from ROIs or just fixations, the goal of the output is to determine similarity of gaze patterns when scanning or surveying a scene. Scoring algorithms do this with two types of replacements as described earlier: substitutions and inserting a blank space. Additionally though, many algorithms determine similarity for substitutions based on spatial distance (i.e., a substitution from A to B is considered a smaller deduction than the substitution from A to G since these are spatially further away on the grid. Generally, there are differential penalties to these replacements usually with no penalty for dashes/gaps, but with a -1 penalty for substitutions (or larger to account for spatial distance) – these are specified in many algorithms including the ScanMatch algorithm in Matlab that will be described in detail later. The goal is to minimize (need an optimization procedure) the amount of penalty, in other words, make as few changes as possible to make the two strings identical. The penalty number

(known as the edit distance or Levenshtein distance; Levenshtein, 1966) serves as a metric of the difference between the sequences. A comparison matrix can be constructed before analyses are conducted which determines the penalty associated with substitutions. Although these are usually symmetrical with a penalty of 0 associated with the diagonal (replacing a value with itself carries no penalty), they can be constructed to be asymmetrical if it makes sense to do so with task constraints.

3.6.1. Example of a Substitution Matrix for above Sequences

Figure 7 shows a substitution matrix for the above 3x3 grid sample sequences.

	A	B	C	D	E	F	G	H	I
A	0	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}
B	C_{21}	0	C_{23}	C_{24}	C_{25}	C_{26}	C_{27}	C_{28}	C_{29}
C	C_{31}	C_{32}	0	C_{34}	C_{35}	C_{36}	C_{37}	C_{38}	C_{39}
D	C_{41}	C_{42}	C_{43}	0	C_{45}	C_{46}	C_{47}	C_{48}	C_{49}
E	C_{51}	C_{52}	C_{53}	C_{54}	0	C_{56}	C_{57}	C_{58}	C_{59}
F	C_{61}	C_{53}	C_{63}	C_{64}	C_{65}	0	C_{67}	C_{68}	C_{69}
G	C_{71}	C_{72}	C_{73}	C_{74}	C_{75}	C_{76}	0	C_{78}	C_{79}
H	C_{81}	C_{82}	C_{83}	C_{84}	C_{85}	C_{86}	C_{87}	0	C_{89}
I	C_{91}	C_{92}	C_{93}	C_{94}	C_{95}	C_{96}	C_{97}	C_{98}	0

Figure 7: A substitution matrix for the example above with AOIs indicated as A-H. The cost of aligning C in one string with E in another is C35. The Levenshtein matrix has a penalty value of $C_{ij} = 1$ when $i \neq j$ (any value not on the diagonal). However, if you wish to inflict a larger gap penalty, the non-diagonal values can vary based on distance from one another.

3.6.2. Algorithms for Automatic Alignment of Eye Tracking Data

Various algorithms such as the Needleman-Wunsch algorithm are most useful for aligning eye tracking data (Needleman & Wunsch, 1970). This algorithm uses a global alignment and flexible scoring scheme to align two sequences. It is optimal when the two sequences are of already similar length (similar number of samples).

Another algorithm is Dijkstra's algorithm which presupposes that a comparison matrix is first transformed into a graph representation and optimizes by finding the shortest path from one node to another on a graph (Dijkstra, 1959). As stated before, the substitution matrix is optimal when there are a finite number of possible transitions and are poorer at higher resolutions. However, at high resolutions, one can represent fine grain differences using a dotplot, such as the image (Figure 8) below.

Figure 8: Sample dotplot from Jarodzka, Nystrom, & Holmqvist (2010).

3.6.3. A Simplified Example by Hand

To understand how most comparison algorithms are calculated and how to interpret a difference score it is helpful to go through a simplified example by hand and discuss what makes that sequence optimal. The example below has two simple sequences and the optimal solution.

Sequence 1	A	B	C	D	C	D	B	A
Sequence 2	A	B	D	C	B	C		
Optimal	A	B	--	D	C	--	B	**A**

3.6.4. Calculating the Difference Value by Hand

Typically, to get the overall score for a particular alignment's similarity, merely sum the penalty for gaps with the penalty for substitutions. Two sequences are considered optimally aligned at whatever combination of gaps and substitutions produces the lowest summed value. For the example above, let's presume that each type of penalty carried a score of one, which would mean that the above example with one substitution and two gaps would have an overall dissimilarity score of 3. However, due to differences in the number of substitutions and gaps that may need to be added based on overall length of the string (longer strings will likely require more substitutions), this value can be easily normed. In the above example the strings have been modified so that both are 8 units long. The normed value of the dissimilarity between the two strings is then simply 3/8 (dissimilarity score over length of string) or .375. Since two identical strings would receive a similarity score of 1, subtract the dissimilarity value from 1 to get a

normed similarity score, in this case, .625. To compare between multiple combinations of trials or observers, use the normed score.

Although it is common to weigh substitutions and gaps with an equal penalty when there are no a priori assumed relationships between the two strings, if a researcher does have justification, it can be entirely reasonable to weigh these penalties differently. For example, in eye tracking data which involve long vectors and each sample may only represent a small fraction of a second's worth of data, when making a comparison between two strings, substitutions are likely to be more heavily penalized than gaps. In 2.3's example, let's say substitutions are still penalized at a value of 1 but gaps are only penalized as .25. This makes the overall dissimilarity score 1.5/8 which equals .1875 or a similarity score of .8125. Additionally, one can take into account a more complex substitution matrix like the one in Figure 7. A replacement of C to B might only carry a penalty score of 1 as these are directly adjacent to one another, but a larger transition as seen in the example from C to A might merit a slightly higher penalty, e.g., 1.25. When this is taken into consideration the dissimilarity becomes 1.75/8 = .21875 or a similarity score of .78125.

3.6.5. Trial vs Observer Level Comparisons

It may be desirable to compare multiple trials for a single observer but it might also be of interest to compare observers against one another for particular trial types. One approach is to calculate average scanpaths for each group one is interested in comparing. However, this just leads to a nominal comparison, not a statistical comparison.

One solution proposed by Feusner & Lukoff (2008) is to calculate the average pairwise similarity between ($d_{between}$) and within (d_{within}) two groups of scanpaths. The metric of interest is the difference between these two values, then significance tests of group similarity are done with permutation tests.

$$d^* = d_{between} - d_{within}$$

4.0 ScanMatch: AN EASY TO IMPLEMENT SEQUENCE ALIGMNENT TOOL

Although there are many potential algorithms to choose from when making comparisons between time series data strings, one particular program that is relatively easy to use and is designed specifically for eye tracking data that is commonly produced in ISR research is ScanMatch. ScanMatch (Cristino, Mathôt, Theeuwes, & Gilchrist, 2010) is a freely available to download Matlab toolbox (works with Matlab version 6.x or newer) which can be used to create grid based AOIs and compare two or more sequences of eyetracking scanpath or fixation data. It can be used with eyetracking data from any eyetracker at any sampling resolution. Although not designed for any specific application, it can be useful for comparing how individuals survey a scene in two different conditions or before and after an event. It does not require the MATLAB bioinformatics toolbox to run but recommends that users carry the license for this toolbox as it is based off algorithms for sequence alignment in the bioinformatics toolbox.

4.1 Option to Include Temporal Resolution and Fixation Duration

In addition to what standard string edit distance packages report, this program includes the option to incorporate temporal resolution based on fixation duration in addition to standard AOI based approaches to sequence alignment. One way it can do this is through temporal binning. By default with a temporal binning value = 0, the sequence alignment algorithm takes into consideration the fixation duration (multiple instances in a row of being within the same AOI) and incorporates this into the alignment. However, if it is important to merely count transitions to other squares or if the user wants to specify a temporal bin duration (amount of time based on an optionally imported timestamp) to consider a single value (e.g., samples taken at 20 ms intervals, temporal bin of 100 ms, would mean that 100 ms of being in a particular AOI would be counted as a single string rather than 5 strings).

ScanMatch is also capable of specifying more than 26 AOIs (most AOI approaches are limited to the number of letters in the English alphabet) and is capable of producing up to a 26x26 square grid (676 total AOIs). In this toolbox, each AOI is specified by two letters, the first in lowercase and the second in uppercase (e.g., aA, aB, bA, etc.). This toolbox utilizes the Needleman-Wunsch global alignment algorithm and allows for a substitution matrix and gap penalty to be specified. It calculates a normalized score for similarity based on the following formula:

$$\textbf{Normalized Score} = \frac{score}{Max(substitution\ matrix) * length\ of\ longest\ possible\ sequence}$$

4.2 Specific Advantages of Using ScanMatch

- The advantage of this toolbox over standard Levenshtein distance algorithms is that this metric is more robust and accurate when dealing with noisy samples with both simulated and human data.

- Additionally, the temporal resolution of ScanMatch is better in that it is able to account for transitions as unidirectional (i.e., the transition from A to C may not be the same as the transition from C to A, but traditional Levenshtein distance metrics are insensitive to direction and therefore classify them as the same. The ScanMatch program is able to differentiate these two distances if specified to do so, thus preserving the temporal order of fixations. It also can account for fixation duration, giving it better temporal precision than most AOI based alignment toolboxes.

4.3 Importing Data Into ScanMatch

To import data into the program, it must be configured in a particular manner that is not the default organization in most raw data files of commercial eye trackers. Data needs to be arranged in the following column structure:

- X coordinate
- Y coordinate
- (optional) Time.
- All other values removed

Each trial needs to be either imported as an individual file (reorganized in Excel or other text/data editing program) or individual variables must be created in Matlab for each trial. If importing multiple trials from a dataset, creating individual trial variables can be easily done in a loop or nested loop (if there is a subject-level and trial-level indicator variable) using the

- PrepareDataforScanMatch.m - Matlab file.

The code for this script is included in Appendix A. This script will segment by subject, condition, trial, etc. and remove any additional variables so that the data can be imported directly into ScanMatch in a readable format.

The data preparation file is designed to import .xlsx files (this can be substituted for a different call such as readcsv to import a .csv file if data is saved in that file format) with multiple trials and multiple subjects into Matlab. If the imported data only has one subject or one trial this code will still function properly. Using nested loops, the data preparation file saves multiple .mat (Matlab's variable structure which can be called directly by ScanMatch) files based on subject number and trial (e.g., Sub5Trial10). However, both the loops as well as the saved file names could be varied on other indicator variables instead, such as correct or incorrect responses, or another variable could be created. For example, if the researcher was interested in comparing scanpaths for correct vs incorrect responses across 5 observers, they could replace the trial loop with an indicator variable (for corr = 1:2) and reference that column from the data instead of the trial column as well as change the save file to create the .mat files (e.g., Sub5Correct).

5.0 STEP-BY-STEP PROCEDURE FOR RUNNING SCANMATCH

There are three major steps to running a similarity comparison between two vectors of eye tracking data in ScanMatch comprised of smaller parameterization steps. The first step is to prepare the data for importation, which may involve trimming extraneous variables and segmentation depending on the needs of the experimenter. The second step is to set parameters for ScanMatch and the third step is to make the similarity comparison itself. A more specific tutorial for these three steps is listed below.

1) Before running ScanMatch or any functions, preprocessing of the data must be completed. The Matlab script PrepareDataforScanMatch.m should be modified and run. The variables that will need to be modified are as follows:
    a. Change working directory to folder where the data file is located
    b. Modify the maximum number of trials
    c. If there are multiple subjects, update the maximum number of subjects
    d. Depending on the nature of the task, it may be necessary to add an additional loop nested as (Subject – Condition – Trial) if there are multiple trials of a certain condition (e.g., Correct vs Incorrect responses, multiple pre-planned conditions, etc.) – will need to write the code similarly as the other two loops with indicator variables
    e. The columns referenced for the loop are based on the above data structure. If you have more columns, you may need to change the column numbers referenced. In the subject loop, column number 1 is referenced and in the trial loop column number 2 is referenced. The output data are specified as columns 3 through 5 as these columns correspond to X coordinate, Y coordinate and Timestep. However, if the imported data varies this needs to be changed.

   Once the preprocessing file has been run there should be multiple .mat files produced, one for every trial of every condition of every subject.

2) Type ScanMatch_Struct() into the command window. This should pop up a GUI that you can interact with to specify certain parameters. See Figures 9 through 11 below for an illustration of the three resolutions used in this pseudo-data analysis. In this GUI window, the grid resolution can be specified. For this test, three different resolutions were used (5x5), (10x10), and (20x20). However, in real analyses grid resolution does not need to be symmetrical and in fact may be suboptimal as this will segment a rectangular screen into smaller rectangles rather than square shaped AOIs. The X and Y resolution are based on the screen resolution of the computer the data was simulated for. For this test, the parameterization in the RoI mask was as follows:

    Xres: 1920
    Yres: 1200
    Xbin: 5 (or 10 or 20 in different resolution conditions)
    Ybin: 5 (or 10 or 20 in different resolution conditions)

RoiModulus: 5 (or 10 or 20 in different resolution conditions)
Threshold: 3.5
GapValue: 0
TempBin: 0

Running this code creates a new Matlab variable, ScanMatchInfo which is a structure that contains all of these parameters.

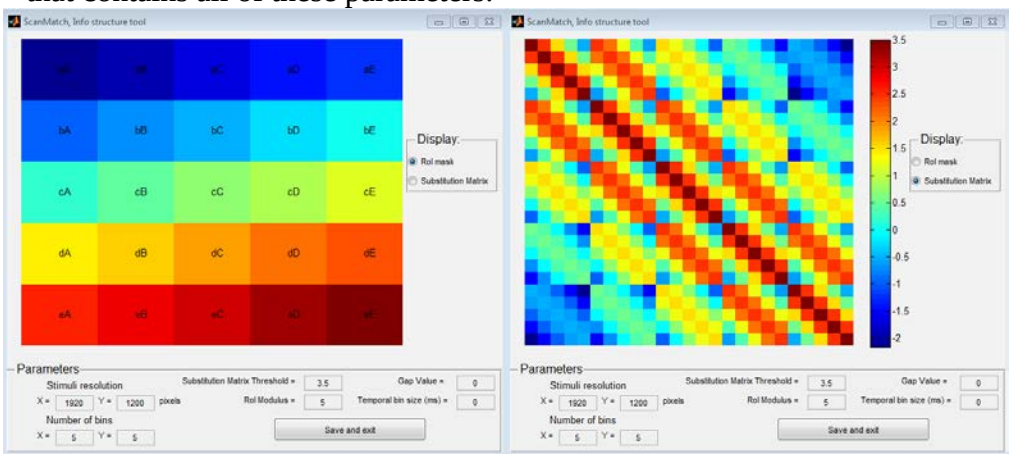

Figure 9: Region of Interest structure and Substitution Matrix for the low resolution (5x5) grid with proper parameterization.

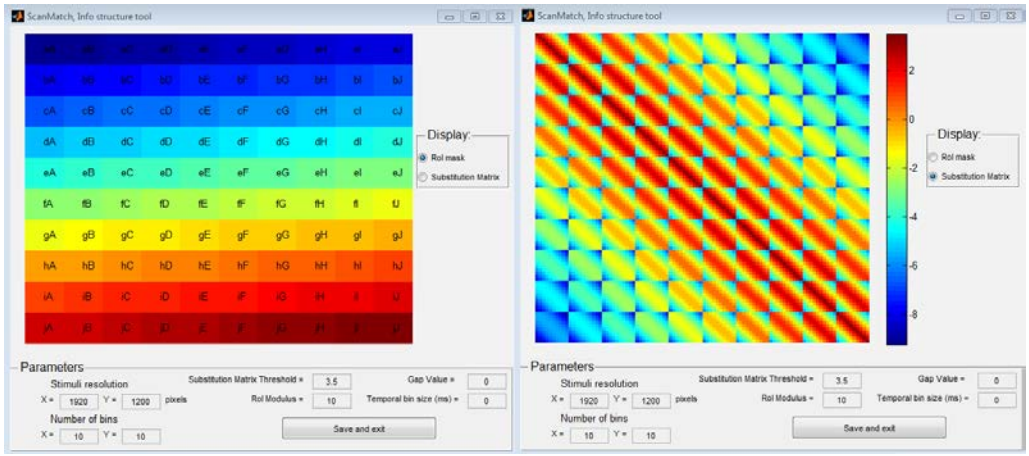

Figure 10: Region of Interest structure and Substitution Matrix for the medium resolution (10x10) grid with proper parameterization.

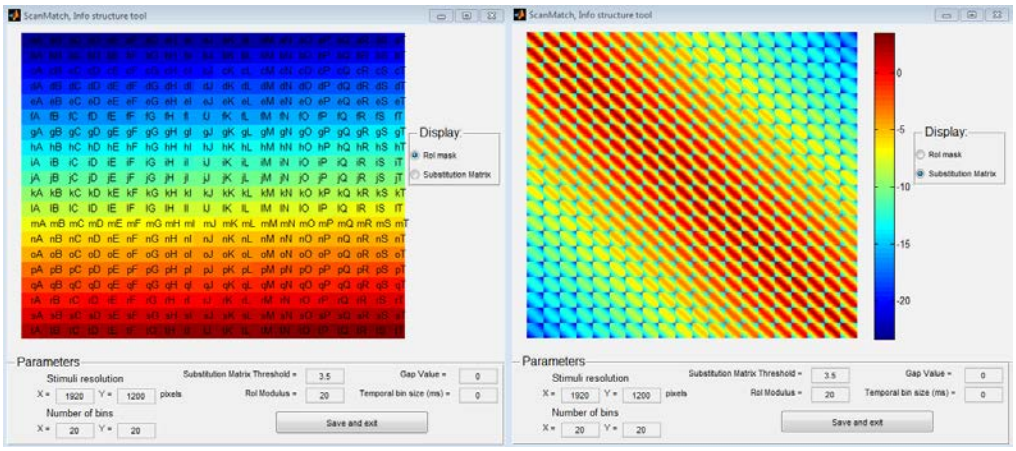

Figure 11: Region of Interest structure and Substitution Matrix for high resolution (20x20) grid with proper parameterization

3) All (x,y) coordinate data was mapped on to the grid structure specified in ScanMatch_Struct using the ScanMatch_FixationToSequence code. To encode a particular trial using the set ScanMatch parameters, code must be entered either into a script or into the Matlab Command Window. The syntax for the first trial of the first subject is as follows:

```
Testseq1= ScanMatch_FixationToSequence(TestTrial1,
ScanMatchInfo)
```

This was then repeated for all trials of all subjects.

The ScanMatch function produced comparisons between the trials and summaries of each alignment and normalized score. The syntax is as follows:

```
[Score align] = ScanMatch(Testseq1, Testseq2,
ScanMatchInfo, 'ShowViewer', 1)
```

Figures 12 and 13 are screenshots and illustrate the output produced by ScanMatch.

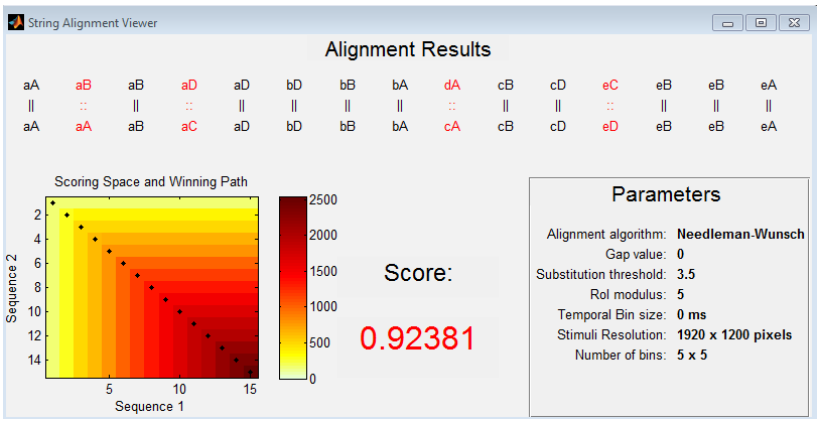

Figure 12: Sequence Alignment between the "Standard" and the equal length minor difference trial at a low (5x5) resolution. As illustrated, few substitutions were made and no gaps were added, making the comparison score fairly high, indicating high similarity.

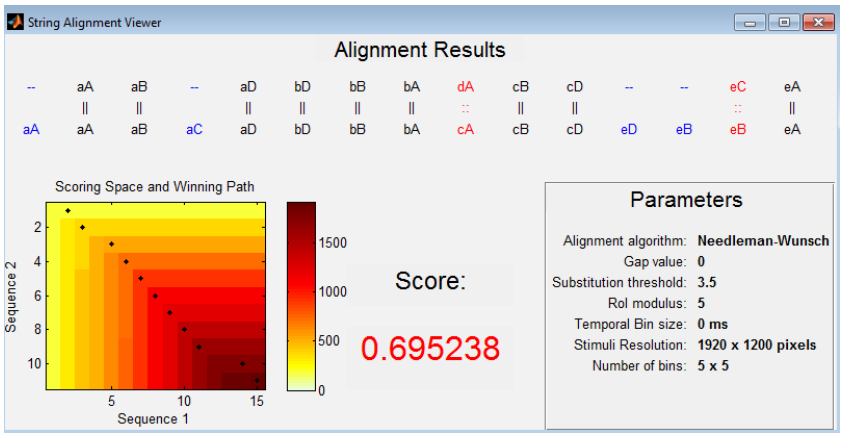

Figure 13: Sequence Alignment between the "Standard" and the different length minor difference trial at a low (5x5) resolution. As illustrated, fewer substitutions were made but four gaps were added, making the comparison score lower than in Figure 7 where no gaps were added, but still indicating relatively high similarity between the two sequences.

6.0 TESTING SCANMATCH ON PSEUDODATA

6.1 Goal

To test the functionality of the ScanMatch software, it was first tested using pseudo-data with predictable results. The goal of this test was to determine if ScanMatch would quantify similarity scores accurately with default parameterization for the substitution matrix and gap penalty. The default gap penalty imposes a larger similarity penalty for transitions of greater spatial distance, characterizing for example a transition from the top right corner of the screen to the bottom left corner of the screen as a more dissimilar difference than a transition from the top right corner of the screen to the direct center of the screen. This would be a generally valid assumption for visual scanpath data in most ISR contexts.

6.2 Blocking and Design

The goal was to test how the program handles gaze patterns of identical morphology, relatively similar morphology and completely different morphology when the strings are of identical length (15 data points each). Following these initial pairings, it was important to compare if the program differentiates when the strings are of different lengths when there are major or minor differences in morphology between scanpaths. Truncated strings in the "Different Length" conditions were identical to the Major/Minor Difference trial strings but with 4 omitted data points in each trial. Based on the specifications of the toolbox, a researcher can choose to have a penalty for substitutions but not for inserting blanks to align the strings. The structure of the test data was designed in testable pairs with predictable similarity ratings. All pseudo-data conditions are plotted in section 6.3, Figures 14, 15 and 16.

The goal of the design in Table 2 is to illustrate how ScanMatch computes similarity scores between trials with known differences in regard to morphology (requiring more substitutions) and length (requiring more gaps). Trial 1 vs Trial 2 should have a perfectly identical score of 1 and serves merely as a sanity test of the algorithm.

Table 2: Blocking for Pseudodata conditions.

Trial Pairs	Condition
Trial 1 vs Trial 2	Identical Values – Same Length
Trial 3 vs Trial 4	Minor Difference – Same Length
Trial 5 vs Trial 6	Major Difference – Same Length
Trial 7 vs Trial 8	Minor Difference – Diff Length
Trial 9 vs Trial 10	Major Difference – Diff Length

6.3 Plots of PseudoData Conditions

Figures 14-16 plot the various conditions used in this pseudodata comparison. Figure 14 is the "standard" path that will be used as a baseline for comparison. Figure 15 consists of two trajectories that are morphologically similar to the standard scanpath, with one comprised of the same number of points and the other consisting of fewer data points to quantify the effect of string length differences. The morphology of the paths in Figure 16 is intended to be quite dissimilar from the standard path, with one path consisting of the same number of trials and the other consisting of fewer trials, again to determine the effect of length differences on similarity score.

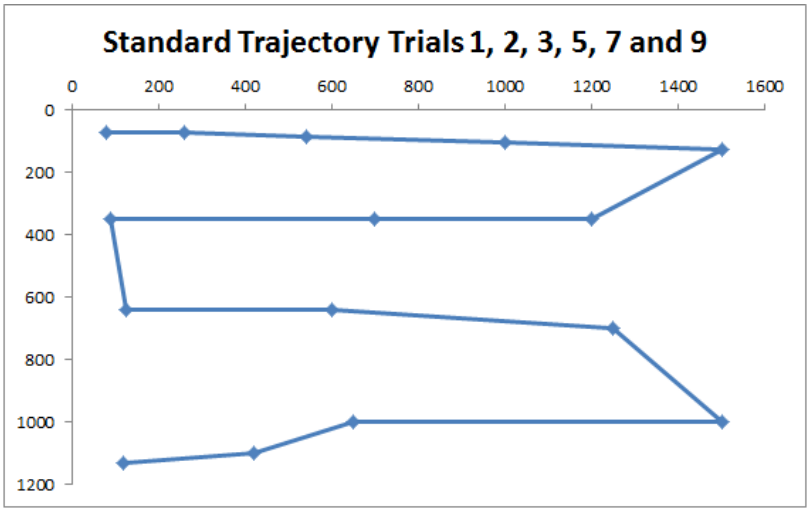

Figure 14: "Standard" trajectory data. This is constructed as a standard Boustrophedon pattern of scanning a scene (scanning back and forth horizontally) with some random noise. This trajectory contains 15 data points.

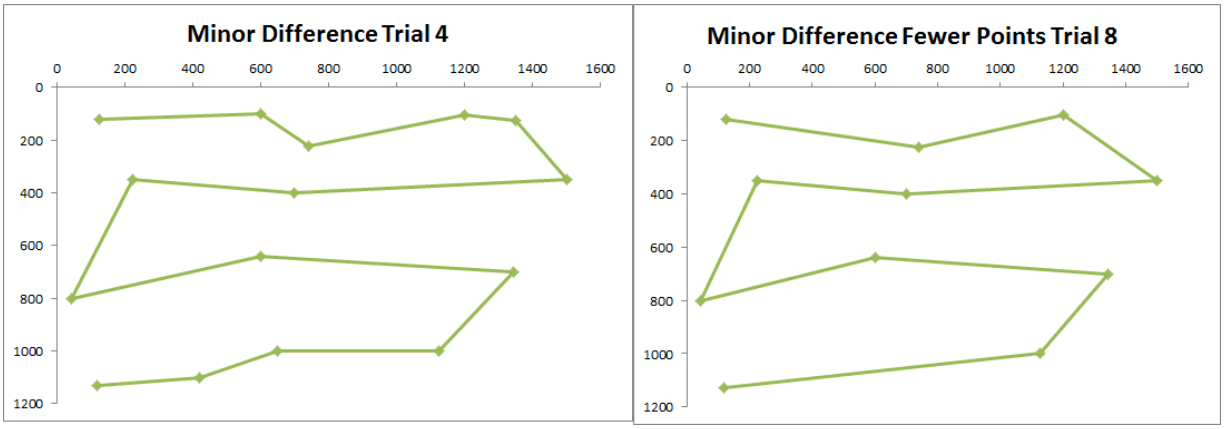

Figure 15: The standard trajectory with minor modifications. Scanmatch should detect strong similarity between the "Standard" scanpath in Figure 1 and Trial 4's scanpath and

strong but slightly poorer similarity between the "Standard" and Trial 8. Trial 4 contains 15 data points and Trial 8 contains 11 data points.

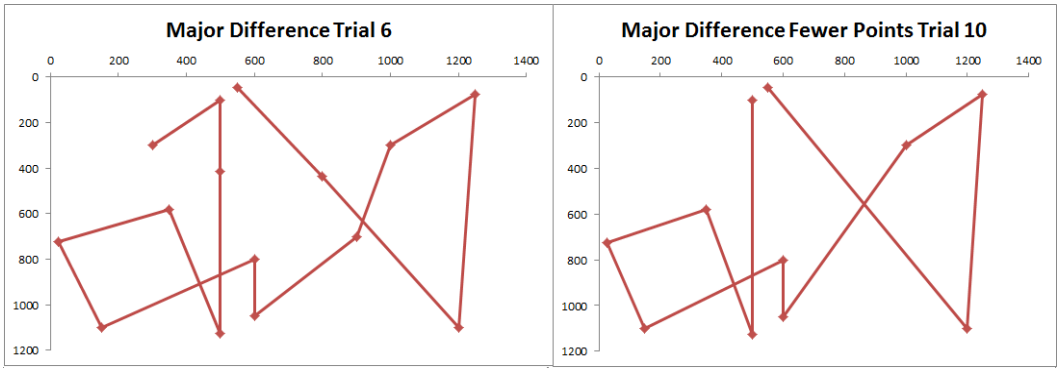

Figure 16: Scanpath designed to differ greatly from the initial scanpath. Scanmatch should detect strong dissimilarity between the "Standard" scanpath in Figure 1 and Trial 6's scanpath and an even poorer match between the "Standard" and Trial 10. Trial 6 contains 15 data points and Trial 10 contains 11 data points.

6.4 Effect of Grid Resolution

Another aspect that is of interest is how the resolution of the superimposed AOI grid affects the similarity output. With a higher resolution grid, the algorithm is more sensitive to minute changes in the fixation location, which may be optimal when examining scanning data with small targets, but may be excessively sensitive for large AOIs. Data was run on three resolution conditions: Low Resolution (5x5 AOI grid), Medium Resolution (10x10 AOI grid), and High Resolution (20 x 20 AOI grid). The real-world task constraints should dictate the precise grid used (either one of these three resolutions or a custom square grid, or an asymmetrical grid), but these results are a means of displaying how similarity ratings between strings vary as a result of manipulations of the grid resolution assuming otherwise default parameters.

6.5 Raw Data Structure

Before data preprocessing to be imported into ScanMatch, the raw data was formatted in a manner similar to a standard dataset for a single observer (see variables below), although simplified in regard to number of variables. Most eyetracking software programs will output additional information or variables, which can either be removed before preprocessing or aspects of the preprocessing code can be modified. The Matlab script PrepareDataforScanMatch.m is heavily commented and provides advice for if there are multiple subjects or multiple trials in a single dataset and how to subset columns for eventual preprocessing output.

Subject	Trial	X Coordinate	Y Coordinate	TimeStep

As was discussed in 4.3, before this data was imported to ScanMatch, all extraneous variables were removed before importing into ScanMatch, in this case: Subject and Trial.

7.0 RESULTS OF PSEUDODATA TEST

Below in Table 3 are the results from a straightforward comparison between the "standard" trial and other comparison trials. Three different resolution conditions were tested:

- High Resolution (HR) – 20x20 grid
- Medium Resolution (MR) – 10x10 grid
- Low Resolution (LR) - 5x5 grid

Table 3: Similarity scores for all 5 trial comparisons at High Resolution (20x20), Medium Resolution (10x10), and Low Resolution (5x5) grid conditions.

Trial Pairs	Outcome HR	Outcome MR	Outcome LR
Trial 1 vs Trial 2	1	1	1
Trial 3vs Trial 4	.6831	.7826	.9238
Trial 5 vs Trial 6	.3781	. 3493	.4750
Trial 7 vs Trial 8	.4223	.5540	.6952
Trial 9 vs Trial 10	.3624	.3017	.3876

The results in Table 3 follow a predictable pattern given that with the default parameters in the program there is a penalty for alignments due to:

- differences in the morphology of the scanpath
- differences in length between the two strings

Similarity scores seem highly inflated at the lowest resolution condition**.** Additionally, similarity scores are lower when the grid resolution is raised due to increased sensitivity to change. However, in certain comparisons, such as those between the standard trajectory and the two trials which are of a different length of time, the similarity score is higher under a higher resolution, indicating that increased spatial sensitivity has a similarity tradeoff with temporal similarity.

7.1 Further Analyses to Test Flexibility of ScanMatch

In addition to testing the primary function of ScanMatch it is also desirable to know the effects of changing some of the parameters as these may need to vary with real data depending on the structure and desired analyses, as well as based on the dimensions of the monitor or screen data is recorded on. For example, all of the previous similarity scores were generated using a grid cut in equal values horizontally and vertically (e.g. if the screen was segmented into 5 vertical segments, it was also segmented into 5 horizontal segments). However, in an actual study it might be better to segment the screen into an unequal number of horizontal and vertical segments if the monitor resolution is not square. For example, instead of generating a 5x5 grid containing rectangles, it might be better on a 1920x1200 pixel monitor to have an 8x5 grid so each square within the grid is 240x240 pixels. This way each cell is not distorted horizontally or vertically. Naturally this should provide slightly different numbers than the results of the initial study using

an equal number of horizontal and vertical segments, but should nevertheless produce similar values as scaling up or down the resolution will produce similar changes.

There are two parameters that can be specified in the ScanMatch_Struct() GUI:

- Gap Penalty
- Threshold Value

7.1.1. Manipulating Gap Penalty

The first of the two parameters that can be manipulated is the Gap Penalty. Increasing this value encourages the program to add gaps with a smaller similarity penalty. The gap penalty cannot be a negative value or the program will display an error. A Gap Penalty of 0 is the default parameterization. This parameterization gives a fairly large penalty for adding gaps. This penalty decreases as a function of increasing the Gap Penalty parameterization, so higher values will encourage adding more gaps with a lower dissimilarity penalty. As seen in the top panel of Table 4 below, where trials were compared using a 8x5 grid and the default threshold of 3.5 with three different gap penalty values as well as in the bottom panel of Table 4 (16x10 grid, threshold = 3.5, and varying gap penalties), regardless of condition, all similarity values are rated as being of a higher similarity when the gap penalty is higher. In addition, when comparing within a single condition, there are substantially more gaps inserted when the gap penalty is .5 or 1 compared to 0 and gaps are inserted into both strings rather than just the shorter string as is done when the gap penalty is 0 (more punitive of gaps).

Table 4: Effects of gap penalty manipulation and grid granularity.

Trial Pairs (8x5 grid)	GP = 0	GP = .5	GP = 1
Trial 1 vs Trial 2	1	1	1
Trial 3vs Trial 4	.8476	.8476	.8762
Trial 5 vs Trial 6	.4480	.5746	.7080
Trial 7 vs Trial 8	.6381	.6762	.7238
Trial 9 vs Trial 10	.3810	.4873	.6210

Trial Pairs (16x10 grid)	GP = 0	GP = .5	GP = 1
Trial 1 vs Trial 2	1	1	1
Trial 3vs Trial 4	.8144	.8335	.8525
Trial 5 vs Trial 6	..3714	.5333	.6890
Trial 7 vs Trial 8	.5875	.6318	.7048
Trial 9 vs Trial 10	.3619	.4952	.6285

7.2.1. Manipulating Threshold Value

The second parameter that can be manipulated is the threshold value. This changes the threshold of the substitution matrix, which is essentially just the cutoff value of what is defined as positive (highly related items) or negative (loosely related items) in the substitution matrix. The following tables illustrate the variation in similarity that occurs when threshold and gap penalty are varied conjunctively in each of the trial pair comparisons (with the exception of Trial 1 vs Trial 2 since these are identical and therefore all outcome values are 1 regardless of threshold or gap penalty parameter fluctuations.

Table 5: Trial Similarity Comparisons Based on Manipulating Gap Penalty and Threshold

Trial 3 vs Trial 4	GP = 0	GP = .5
Threshold = 2	.7333	.8333
Threshold = 3.5	.8476	.8476
Threshold = 5	.8933	.8933

Trial 5 vs Trial 6	GP = 0	GP = .5
Threshold = 2	.3723	.6333
Threshold = 3.5	.4480	.5746
Threshold = 5	.5336	.5869

Trial 7 vs Trial 8	GP = 0	GP = .5
Threshold = 2	.5667	.6667
Threshold = 3.5	.6381	.6762
Threshold = 5	.6667	.6933

Trial 9 vs Trial 10	GP = 0	GP = .5
Threshold = 2	.3195	.5667
Threshold = 3.5	.3810	.4873
Threshold = 5	.4517	.4933

As can be seen from the panels in Table 5 above, the Gap Penalty increases are more influential on changing the similarity value when the threshold is decreased but have almost no influence on the similarity score when the threshold is higher than the default value of 3.5. However, this is because even with a GP = 0, a higher threshold value leads to higher ratings of similarity.

These analyses demonstrate the importance of consistent parameters when making comparisons. Variations in parameters should be motivated by theory. For eye movements, the distance between ROIs determines the substitution matrix. Therefore the zero-point in the substitution matrix should be defined by the variability in saccade length, which will vary by experiment (Cristino, et al., 2010). Generally speaking, the threshold should be defined as 2 standard deviations of all saccadic amplitudes. If the substitution matrix threshold is set most appropriately, the Gap Value can (and is recommended to) be set to zero.

8.0 TESTING SCANMATCH ON EXPERIMENT DATA

8.1 Goal

The goal of testing ScanMatch on real data is to:

- Determine if ScanMatch is capable of parsing data from a commercial eye tracker, namely the Tobii-T60 Eye tracker.
- Determine if the cleaning script created to prepare the data to be readable by ScanMatch useful on real data, which is significantly more complex than the pseudodata generated in Section 5.0.
- Determine a range of similarity values for non-contrived, noisy data.
- Demonstrate the value of ScanMatch for making comparisons between multiple trials, multiple conditions and multiple observers and conducting statistical analyses for comparison rather than merely providing tables of descriptives.

The following data involves significant events. Before/after event comparisons will need to be made. However, it will also be useful to test for an "optimal" resolution based on the characteristics of targets and eye trajectory data in this task. Once a relatively optimal grid resolution is determined for blocking whole screen ROIs, comparisons will be conducted based on the specific blocking of the task.

8.2 Blocking and Design

The real experiment data sampled to test the ScanMatch algorithm was taken from a search/surveillance task similar to tests conducted in the Analyst Test Bed and other ISR applications (Piasecki, 2016). The sample eye path data was collected using a Tobii-T60 stationary eye tracker. In the task, observers would look at a scene and attempt to identify where 4 card suit symbols were hidden in each image. Stimuli images were visually rich and the card suits were difficult to detect (the color of each card suit was only 10% different than the color of the area it covered, making it difficult to distinguish from the surrounding backdrop). In addition, during the task there would sometimes be a flashing dot present. After each trial, observers were asked if they detected the flashing dot and if so which quadrant of the screen it was located in.

One of the hypotheses of interest in this task is if the search patterns/eye path trajectories varied as a result of detecting one of the card suits or one of the flashing dot distractors. Identifying the card suits represents a conscious detection in that observers would actively search for these targets whereas the flashing dot distractor serves as a surprise, sometimes undetected, stimulus that was not actively searched for. In this case the researcher is interested in characterizing the similarity of path data before and after each event of interest (detecting a card suit or detecting the flashing dot stimulus). Although the researcher could plot the path data for all trials, there are numerous samples and it is necessary to quantify degree of similarity, not simply examine

morphology and rely upon subjective judgment, which tends to be inaccurate and potentially also biased. For this reason numerous comparisons are of interest.

- Before any event within observer, i.e., first few seconds of trial 1 vs trial 4 vs trial 8 for Observer 1
- Between observer baseline comparison – identical to first test in timing (before any events/detection) but between same trials of each observer
- Within observer before/after each suit detected within a single trial
- Within observer before/after flashing dot detected within a single trial
    - Each of the two above may be cleaved into multiple segments depending on number of suits detected or number of distractors present on a given trial
- (Possibly) Construct a path of same length n (number of data points = n), with a particular morphology – can make direct comparison of a single path to one of these pre-constructed paths, for example:
    - Scanning back and forth horizontally
    - Scanning back and forth vertically
    - Feature examination (buildings)
    - Feature examination (by color)
    - Circular

8.2.1. Plots of Raw Gaze Paths

A few sample trials from two participants were used in these analyses. Figures 17-20 are plots of raw data for each trial compared in ScanMatch. Since comparisons were made within trial before and after the critical event, before and after each event of interest are plotted in panels.

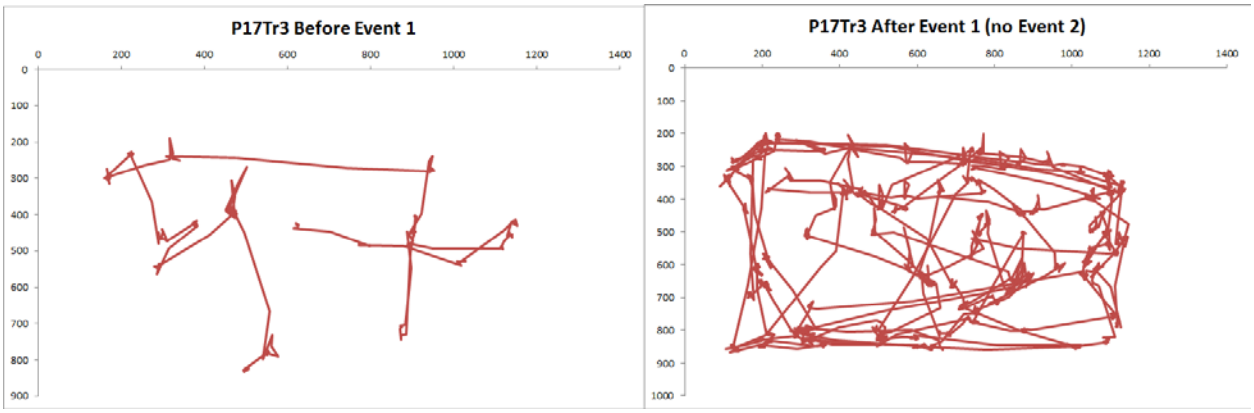

Figure 17: Piasecki (2016) Observer 17 Trial 3 data. The left panel illustrates the eye scanpath before a suit is detected while the right panel illustrates the eye scanpath after the first suit is detected. In this trial, the observer only found one card suit.

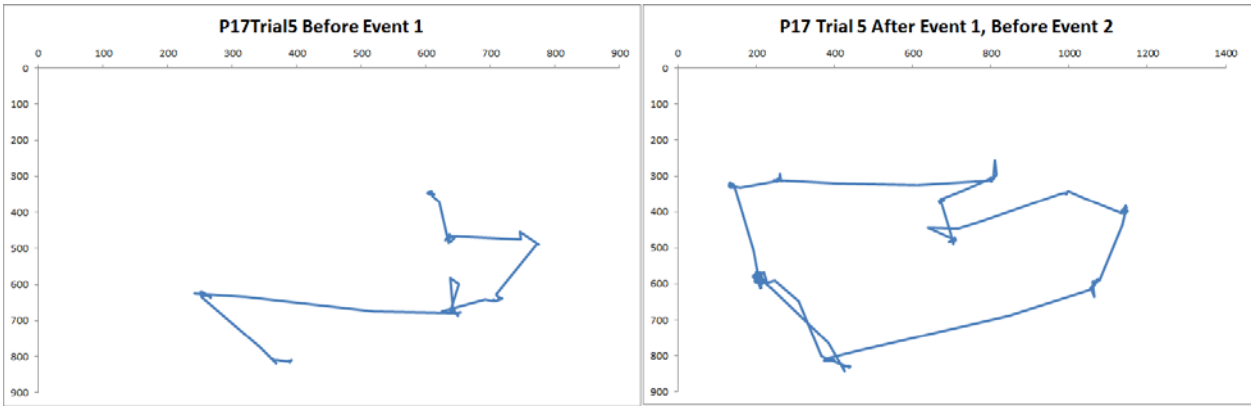

Figure 18: Piasecki (2016) Observer 17 Trial 5 data. The left panel illustrates the eye scanpath before a suit is detected while the right panel illustrates the eye scanpath after the first suit is detected but before the second card suit is detected.

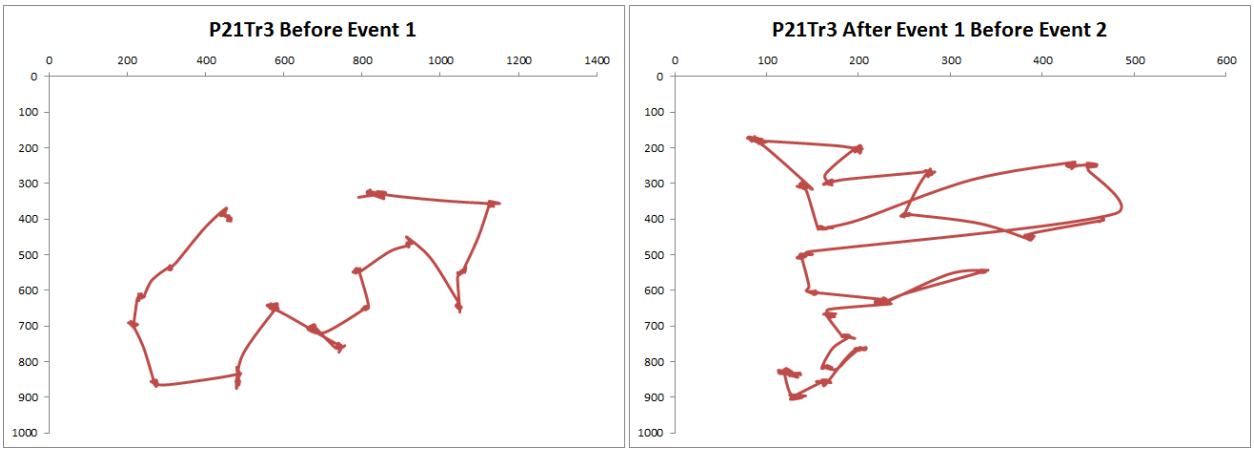

Figure 19: Piasecki (2016) Observer 21 Trial 3 data. The left panel illustrates the eye scanpath before a suit is detected while the right panel illustrates the eye scanpath after the first suit is detected. In this trial the observer only found one card suit.

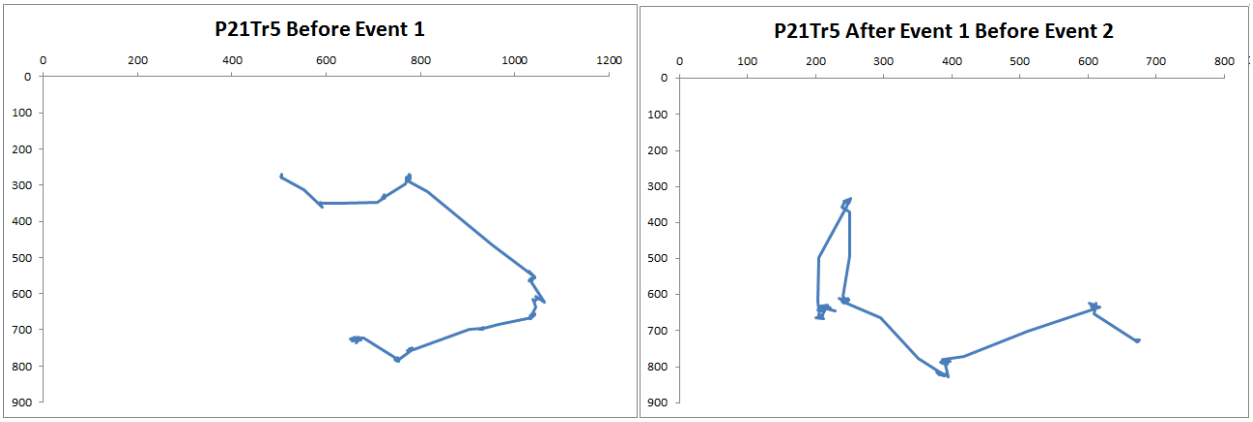

Figure 20: Piasecki (2016) Observer 21 Trial 5 data. The left panel illustrates the eye scanpath before a suit is detected while the right panel illustrates the eye scanpath after the first suit is detected. In this trial the observer only found one card suit.

8.3 Raw Data Structure

The raw data needed to be sorted and cleaned before being run through ScanMatch. The raw Tobii output consisted of numerous output variables, but of interest for the purposes of sequence alignment were the following: MediaName, KeyPressEventIndex, RecordingTime, GazePointX (MCSpx), GazePointY (MCSpx).

MediaName serves as an indicator variable of which trial image is being used and in the PrepareDataforScanMatch_Piasecki.m file is used as a trial indicator variable (Script included in Appendix B). KeyPressEvent indicates when a particular event occurs. Recording time is measured as the number of milliseconds following the start of the recording. GazePointX (MCSpx) and GazePointY (MCSpx) are the coordinates on screen of gaze position based on the image resolution rather than the screen resolution. In this case since images were full screen, whether basing coordinates on image or on screen resolution is not different.

8.4 Preprocessing and Data Cleaning

Before being imported into the Matlab cleaning script unnecessary or empty columns were omitted. Only the columns of data listed above were used after initially loaded. After running through the Matlab preprocessing script, individual data files were created based on segmentations for Observer, Trial, and events. In this case, events are discrete button presses observers made when they detected one of the four card suits. The data contained no indicator for when a distractor appeared so it was not possible to segment by these events. However, if an indicator variable is added to denote when a distractor appeared in a trial the PrepareDataforScanMatch_Piasecki.m file could easily be modified to segment based on these temporal events. The code does not contain a subject loop at this time due to the low number of test subjects but this could easily be added for speed when batch processing large groups of observer data.

8.5 Procedure for Running ScanMatch

The procedure for running ScanMatch is identical to the process outlined in detail for the test data set. The procedure is also outlined in the Matlab script PiaseckiDataProcedure.m (script included in Appendix C).

9.0 CRITICAL COMPARISONS AND INTERPRETATION

1) Single subject, within trial, before and after first event

This analysis establishes a comparison of a baseline for a trial versus the same observer's behavior after they detected one of the card suits. Low similarity scores would indicate that the observer dramatically changed his or her behavior after detecting a target while high similarity scores would indicate that observer searching behavior was roughly similar throughout the trial.

2) Single subject, between trials, before first event comparison and after first event comparison

This analysis establishes a comparison of the same subject across trials for consistency at various stages of the search process. Comparing the before event (baseline) of two different trials allows for examination of how consistent the observer's overall search pattern was. Comparing the trajectories after finding the first card suit also determines how consistently observer behavior is after finding a target. High similarity scores indicate that observers use roughly the same search strategy in each trial. Low similarity scores indicate inconsistency of observers between trials.

3) Comparisons between subjects, before first event comparison and after first event comparison

This analysis establishes the level of dissimilarity between observers at different points of visual search (during the baseline or after a target is detected). High similarity scores indicate that the two observers used roughly the same search strategy when looking for card suits. Low similarity scores indicate that observers were using different search strategies.

9.1 Grid Resolution ROIs and Similarity

Multiple grid resolutions were tested. Default parameterization of 0 was used for the gap penalty, meaning that gaps were heavily weighted. For this reason, resolution did not matter (values did not differ from 5x5 to 10x10 grid) since similarity was most heavily penalized for gaps.

Since different observers respond at different times on each trial, the degree of dissimilarity of length of strings is not controlled and are therefore of highly variable length. For this type of data it is not recommended to use a gap penalty of 0 as this artificially deflates the similarity of the scores.

In this analysis the comparison of interest is the morphology of the scanpaths under various conditions. In the following tables these codes will be used for labelling events: Baseline indicates before a card suit is found, Event 1-Event 2 indicates the time between when one card suit is found and a subsequent card suit is found, and Final indicates from when the final card

suit is found on a given trial until the end of the trial. However, similarity scores for critical trials with a gap penalty equal to 0 are reported on the following page:

Table 6: Similarity scores at low ROI grid resolution for across observers, across trials, and across events.

Grid Resolution: 5x5, GP = 0

Subject(s)	Trial(s)	Event(s)	Similarity
17	5	Baseline vs Event 1-Event 2	**.3449**
17	3	Baseline vs Event 1-Event 2	**.1418**
17	5 vs 3	Baseline	**.2573**
17	5 vs 3	Event 1-Event 2	**.1204**
21	5	Baseline vs Event 1-Event 2	**.6375**
21	3	Baseline vs Event 1-Event 2	**.6248**
21	5 vs 3	Baseline	**.4157**
21	5 vs 3	Event 1-Event 2	**.4918**
17 vs 21	5	Baseline	**.1024**
17 vs 21	3	Baseline	**.4536**
17 vs 21	5	Event 1-Event 2	**.3466**
17 vs 21	3	Event 1-Event 2	**.2622**

Data was re-run using a gap penalty of 1 to better assess sensitivity to grid resolution. The results did show greater variability and were more strongly influenced by grid resolution.

Table 7: Across subject, across trial, and across event similarity comparisons at varying ROI grid resolutions.

Subject(s)	Trial(s)	Event(s)	Sim 5x5	Sim 10x10	Sim 20x20
17	5	Baseline vs Event 1-Event 2	**.5302**	**.4903**	**.5142**
17	3	Baseline vs Event 1-Event 2	**.3870**	**.3850**	**.3819**
17	5 vs 3	Baseline	**.4581**	**.4373**	**.4882**
17	5 vs 3	Event 1-Event 2	**.3717**	**.3703**	**.3670**
21	5	Baseline vs Event 1-Event 2	**.7311**	**.6582**	**.6571**
21	3	Baseline vs Event 1-Event 2	**.7397**	**.6764**	**.7061**
21	5 vs 3	Baseline	**.5875**	**.5526**	**.5513**
21	5 vs 3	Event 1-Event 2	**.6800**	**.6249**	**.6779**
17 vs 21	5	Baseline	**.3590**	**.3584**	**.3584**
17 vs 21	3	Baseline	**.6458**	**.5926**	**.6481**
17 vs 21	5	Event 1-Event 2	**.5215**	**.4703**	**.4846**
17 vs 21	3	Event 1-Event 2	**.4725**	**.4509**	**.4539**

These results illustrate that with a larger gap penalty overall values are reported as larger since aligning based on variable length strings is not heavily penalized, so similarity reflects a comparison of the morphology of scanpaths. As seen with the test data, similarity scores are overall lower when using a higher resolution grid. However, the results above indicate that at a very high resolution (20x20), scores become more similar in some conditions. It may not be consequential for comparisons, but for a full experiment it would be important to report the resolution used. As noted with the test data, the higher similarity values under higher spatial resolution appear to occur when the two strings are highly dissimilar in length.

9.2 Statistical Results

Naturally, in a real experiment, many more of these comparisons would be made and statistical analyses would be conducted. To show one very small scale illustration of this, some basic statistics were computed. The first set of analyses is to determine differences over the time course of trials between observers. In a full experiment more of these comparisons would be made with a greater number of subjects. In line with previous research comparing similarity of 613479

when making correct and incorrect answers on a physics test (Madsen, Larson, Loschky, & Rebello, 2012), these analyses compare segments paired based on event (e.g., comparing observers' gaze patterns before any targets are detected) versus incongruous pairs (e.g., comparing different gaze patterns for different event blocks between observers). The latter should be significantly smaller, with no difference indicating that potential patterns based on time could be inseparable from noise.

9.2.1. Between Observer Simple Analyses and Descriptive Statistics

Observer 17 and Observer 21 were compared across trials at various time segments within each trial, determined relevant to an event, in this case reporting detection of a card suit. Examining initial trial trajectory across all trials between observers allows for an understanding of the degree of regularity between observers at the start of a given trial. To perform this analysis, pairwise similarity values were made between observers on each trial before any targets were detected. Parameterization of ScanMatch was set at a 10x10 resolution grid and gap penalty was set to 1 to account for variability in time to first detection between trials and between observers. This way the similarity comparisons were based more on the scan pattern itself rather than segment duration.

When comparing observers' gaze patterns before a card suit was detected, ScanMatch similarity scores ranged from .3668 to .7616, (Mean = .5556, SD = .1225). A histogram of pre-event similarity scores between observers is displayed in Figure 13 in the top left panel. Given the variability of eye movements within observer and the sensitivity of the grid resolution used, these represent fairly high degrees of similarity, indicating that at the start of each trial, observers display a similar search strategy.

Similarity of trajectories between observers declined after the first target was found, indicating that perhaps after finding one target, observers made different changes to their search strategies. The middle panel on the left of Figure 21 provides a histogram of the range of similarity values after a single target was found. Values ranged from .3603 to .5818, (Mean = .4645, SD = .0795).

Finally, observer trajectories after the final target was found were compared. The bottom left panel of Figure 21 displays the histogram of similarity scores from when the final target was detected until the end of the trial. Similarity scores during this time frame are higher than after detecting the first target. This could be due to having a greater number of samples. In most trials both observers' longest segment was after the final target was detected, indicating that after early success in the trial, observers would unfruitfully search for most of the trial time.

Figure 21: Left images: Histograms of similarity scores between Piasecki (2016) Observer 17 and Observer 21 on each trial. A score of 1 indicates perfect similarity and a 0 indicates perfect dissimilarity. Right images: graphs depicting histograms for non-congruous segments over the trial. Blue indicates OBS17 as the first condition listed in title and OBS21 as the second condition listed in title and red indicates OBS21 as first condition and OBS17 as second condition.

The between subjects results seem to imply that people behave fairly similarly until they are disrupted during a trial by finding a target, at which point they diverge in scanning similarity, and then become more similar again as they approach the end of the trial. In line with previous

research, it is useful to compare non-congruous segments of the trials across observers. If similarity values are high, then the variability from beginning to the end of the trial may be simply an artifact or random noise. However, if these values are consistently lower, this indicates that there are robust similarities across observers over the course of the trials.

The right panel shows histograms of similarity values for incongruous trials (i.e., unpaired temporally across observers). An example of an "incongruous trial" would be the sequence before a target is detected for OBS17 paired with the scanpath after the first target is detected for OBS21. A paired t-test was conducted to determine if there was a significant difference between the similarity values on the temporally paired trials versus not temporally paired trials. The incongruous conditions had a significantly lower similarity value (M = .46, SD = .10) than the temporally paired similarity scores (M = .52, SD = .11), $t(66) = , p = .01$. This indicates that across time, observers demonstrate fluctuations in search strategy over the time course of each trial that is above and beyond noise. Observers demonstrate similar search strategies at the start and end of trials but employ divergent search strategies between detecting various targets.

9.2.2. Within Observer Simple Analyses and Descriptive Statistics

In addition to the value of examining single subject consistency within trial, examining within subject metrics might allow for a better understanding of why similarity scores decrease after finding an initial target, but then increase again near the end of a trial. An important question is how scanpath morphologies vary throughout a trial and if this is a fairly consistent between observers. For this reason, one individual observer was rated on similarity between multiple trials on the same event (e.g., before a target detected). As with the between subject analyses, these paired timing events were contrasted with incongruous events to determine if differences in similarity ratings were systematic or due to noise. The similarity scores were significantly larger for the paired events by trial (M = .557, SD = .151) than event mismatched trials (M = .433, SD = .110), $t(3.913) = , p < .001$, again indicating that similarity scores are not simply reflective of random noise.

Multiple similarity comparisons were made between trials for the following three conditions: Before any targets detected (M = .523), between target 1 and target 2 detected (M = .457), and after final target detected (M = .690). Not all permutations of trials were compared, but rather were compared as Trial 1 vs Trial 2, Trial 2 vs Trial 3, etc. This was to limit the number of comparisons made to a reasonable test number. In a full analysis comparisons of all permutations could be computed if desired.

An ANOVA demonstrates that similarity varied across events for Observer 17, $F(2,27) = 10.426$, $p < .001$. There was no significant difference between similarity scores from before any targets were detected to after the first target was detected, $t(9) = 1.257$, $p > .05$, but there was a significant difference between after the first target detected and after the final target was detected, $t(9) = -4.329$, $p < .001$, as well as between before targets were detected and after the

final target was detected, $t(9) = -5.64$, $p = .001$. The following table summarizes this statistical information. Similarity scores were overall significantly higher after the final event, indicating that eye scanpaths for Observer 17 may have become more consistent or normalized after detecting multiple targets across trials that was more random earlier in the trial, before targets were detected. This may not be a robust pattern across observers. However, this illustrates a potential starting point for within observer comparisons.

Table 9: Differences in Similarity scores based on target detection across trials for Piasecki (2016) Observer 17.

Comparison	Mean Difference	t-statistic	p-value
Before events vs. between E1 and E2	.066	1.257	.12
Between E1 and E2 vs. After final event	-.232	-4.329	.001*
Before events vs. After final event	-.166	-5.646	.0001*

For these within observer analyses it may also be interesting to test similarity based on the order the trials appeared rather than based on coding number. Behavior for a certain observer may be consistent over the course of viewing multiple trials or perhaps that observer's behavior changes over the course of trials as observers test out various search strategies. However, no significant difference was detected however for Observer 17 in regard to similarity based on pairs organized by viewing order or by coding order, $t(29) = 1.699$, $p > .05$. It may be irrelevant within subject to order pairs based on viewing order rather than on the easier to test coding order. However due to low N this is not conclusive and testing on more observers would need to be done.

The two charts below provide an illustration of the distribution of scores based on across trial similarity scores. The first image illustrates that the overall similarity values are higher when comparing trials on the same temporal event across trials compared to similarity values when comparing different events across trials (left image, Figure 22). As expected and demonstrated by the statistical analyses, similarity ratings of scanpaths are significantly higher during the same temporal events compared to similarity scores between different temporal events. The right image of Figure 22 illustrates the range of similarity scores for each event to illustrate how consistent the observer was in their gaze pattern during the three temporal events of interest: before detecting any targets, after a target was detected, and after the final target was detected.

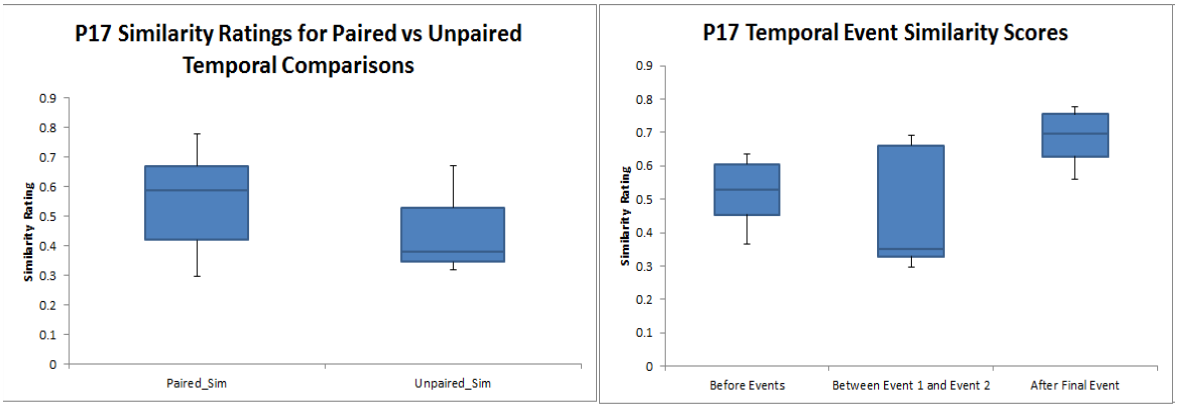

Figure 22: The left image illustrates the overall larger similarity scores detected across trials during the same temporal events compared to pairings between non-congruous events. The right image illustrates the overall larger similarity scores for gaze data after the last target was detected.

10.0 CONCLUSIONS FOR USING SCANMATCH

Overall, the ScanMatch toolbox is a highly flexible set of programs that can be useful in comparing multiple sequences of eye scanpath (or movement trajectory path) data between multiple observers, multiple trials, or between multiple segments of a trial where specific events can be used to segment critical trial components. The main purpose of this tool is to relate the similarity of scanpaths. Proper data cleaning and formatting must be emphasized for using this tool and input data should be formatted as three columns: X-Coordinate data, Y-Coordinate data, and Time, with the lattermost of these three as optional to include. Each row should be a single sample over time with the first row denoting the first time point and the final row indicating the final time point in that segment. Data must be input into ScanMatch as individual trials (or segments of trials if within trial comparisons are being made).

The tremendous customizability of ScanMatch is harnessed in the GUI interface for the ScanMatchStruct() function which is run before making any comparisons. Users can easily adjust the parameterization of comparisons by modifying the number of bins as well as the Gap Penalty and Substitution Matrix threshold. Although default values may be sufficient for many studies, in surveillance tasks, a spatial binning resolution of approximately 10x10 for a standard sized screen (1280x1024 pixels) has sufficient sensitivity to dissimilarity between trajectories without generating artificially high comparison values when length of segment is more dissimilar. Additionally, gap penalty should vary based on task. A gap penalty of 0 is most appropriate if conducting a study on a standard experimental task with short trials that are of relatively the same length or when looking at gaze data from similarly sized temporal bins within a trial (e.g., comparing the first minute of the trial to the second minute of the trial or comparing two trials of 1000ms each to one another). However, if the purpose of using ScanMatch is to make comparisons of the morphology or scanning strategy, regardless of overall trial length or segment time differences, a higher gap penalty is better. At very high values (GP approx. 5), there will be virtually no penalty for differences in string length and adding gaps. However, in surveillance tasks it may be useful to still bestow some penalty for differences based on gaps added for the comparison, meaning that gap penalty values should not be greater than 1 on most tasks such as those explored in this report.

Numerous statistical comparisons can be setup for within observer and across observer similarity comparisons. The comparisons included in this technical report are merely a small number of possible analyses and are not exhaustive. However, in most ISR experiments, comparisons will need to be made on the subject level looking at consistency of gaze pattern across time during one or more trials of a surveillance task. Because ScanMatch only provides alignment and similarity scores, the output of ScanMatch should be augmented with plots of raw or averaged trajectories based on relevant comparisons. This will allow for qualitative descriptions and characterizations of the morphology (e.g., is the observer scanning back and forth from top to bottom or focusing on various landmarks, etc.). Between observers, it is useful to compare trials of like kind (as would be done with any other metric) as well as trials that should not be correlated to determine if variability across conditions is merely due to noise as was done both in this report as well as previous studies which have implemented ScanMatch (Madsen, et al., 2012).

11.0 REFERENCES

Backs, R. W., & Walrath, L. C. (1992). Eye movement and pupillary response indices of mental workload during visual search of symbolic displays. *Applied Ergonomics*, 23(4), 243-254.

Cristino, F., Mathôt, S., Theeuwes, J., & Gilchrist, I. D. (2010). ScanMatch: A novel method for comparing fixation sequences. *Behavior Research Methods*, *42*(3), 692-700.

De Gennaro, L., Ferrara, M., Urbani, L., & Bertini, M. (2000). Oculomotor impairment after 1 night of total sleep deprivation: a dissociation between measures of speed and accuracy. *Clinical Neurophysiology*, 111(10), 1771-1778.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269-271.

Djamasbi, S., Siegel, M., & Tullis, T. (2011, July). Visual hierarchy and viewing behavior: An eye tracking study. In International Conference on Human-Computer Interaction (pp. 331-340). Springer Berlin Heidelberg.

Drew, T.,Vo, M. L., & Wolfe, J. M. (2013). The invisible gorilla strikes again: Sustained inattentional blindness in expert observers. *Psychological Science*, 24(9), 1848-1853.

Duchowski, A. (2007). Eye tracking methodology: Theory and practice (Vol. 373). Springer Science & Business Media.

Feusner, M., & Lukoff, B. (2008, March). Testing for statistically significant differences between groups of scan patterns. In Proceedings of the 2008 Symposium on Eye tracking research & applications (pp. 43-46). ACM.

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Van de Weijer, J. (2011). Eye tracking: A comprehensive guide to methods and measures. OUP Oxford.

Jarodzka, H., Holmqvist, K., & Nyström, M. (2010, March). A vector-based, multidimensional scanpath similarity measure. In *Proceedings of the 2010 symposium on eye-tracking research & applications* (pp. 211-218). ACM.

LeDuc, P. A., Greig, J. L., & Dumond, S. L. (2005). Involuntary eye responses as measures of fatigue in US Army Apache aviators. *Aviation, Space, & Environmental Medicine*, 76(Supplement 1), C86-C91.

Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions, and reversals. In Soviet physics doklady (Vol. 10, No. 8, pp. 707-710).

Madsen, A., Larson, A., Loschky, L., & Rebello, N. S. (2012, March). Using ScanMatch scores to understand differences in eye movements between correct and incorrect solvers on physics problems. In *Proceedings of the Symposium on Eye Tracking Research and Applications* (pp. 193-196). ACM.

Morris, T. L., & Miller, J. C. (1996). Electrooculographic and performance indices of fatigue during simulated flight. Biological Psychology, 42(3), 343-360.

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3), 443-453.

Piasecki, A. M. (2016). Improving Anomaly Detection through Identification of Physiological Signatures of Unconscious Awareness (Master's thesis, Dayton Ohio, Wright State University). Obtained from OhioLink.

Poole, A., & Ball, L. J. (2006). Eye tracking in HCI and usability research. Encyclopedia of Human Computer Interaction, 1, 211-219.

Schleicher, R., Galley, N., Briest, S., & Galley, L. (2008). Blinks and saccades as indicators of fatigue in sleepiness warnings: looking tired? *Ergonomics*, 51(7), 982-1010.

Sirevaag, E. J., Rohrbaugh, J. W., Stern, J. A., Vedeniapin, A. B., Packingham, K. D., & LaJonchere, C. M. (1999). Multi-dimensional characterizations of operator state: A validation of oculomotor metrics. Office of Aviation Medicine Final Report. Washington, D.C.

Stasi, L., Renner, R., Staehr, P., Helmert, J., Velichkovsky, B., Cañas, J., . . . Pannasch, S. (2010). Saccadic Peak Velocity Sensitivity to Variations in Mental Workload. *Aviation, Space, & Environmental Medicine*, 81(4), 413-417.

Tsai, M. J., Hou, H. T., Lai, M. L., Liu, W. Y., & Yang, F. Y. (2012). Visual attention for solving multiple-choice science problem: An eye-tracking analysis. *Computers & Education*, 58(1), 375-385.

12.0 LIST OF ACRONYMS

ISR	Intelligence, Surveillance, and Reconnaissance
ROI	Region of Interest
FMV	Full Motion Video

APPENDIX A: Preparing Data for ScanMatch Script

```
% REFORMATTING DATA FOR ScanMatch Toolbox
% Code Author: Mary Frame
% The following code is used to extract X coordinate, Y coordinate, and
% Time information from a larger dataset so each .mat dataset only contains
% a single trial of information.  ScanMatch reads data one trial at a time.

% Change to correct working directory
cd('C:\Users\Mary.Frame\Documents\Sequence Alignment\ScanMatch Matlab
Code\ScanMatch');

% Read in full dataset if not already in individual files by trial
filename = 'TestData.xlsx';
data = xlsread(filename);
% X Y and Time, the values needed for output - columns will vary based on
% raw data structure
data2 = data(:,3:5);

% To generate small .m datasets or variables to be used by ScanMatch
% algorithms, use a for loop to extract trial data

% Specify maximum subject number, if only one subject can be commented out
maxsub = 1;

% Specify maximum trial number
maxtrial = 10;

% Subject loop and code contained within can be commented out if only one
subject
for sub = 1:maxsub
    sub_str = num2str(sub);
    subindx = data(:,1)==sub;
    i1 = find(subindx, 1, 'first');
    i2 = find(subindx, 1, 'last');
    datasub = data2(i1:i2,:);
    for tr = 1:maxtrial
        tr_str = num2str(tr);
        trindx = data(:,2)==tr;
        i3 = find(trindx, 1, 'first');
        i4 = find(trindx, 1, 'last');
        datatr = datasub(i3:i4,:);
        % This is where you specify the name of your output files by trial
        % Add subject number if that loop is used
        savfile = ['TestTrial','Sub',sub_str,'Tr',tr_str,'.mat'];
        % This saves the data set by each trial
        save(savfile,'datatr');
end
end
```

APPENDIX B: Prepare Data for ScanMatch Piasecki Data

```
% REFORMATTING DATA FOR ScanMatch Toolbox
% Script Author: Mary Frame
% The following code is used to extract X coordinate, Y coordinate, and
% Time information from a larger dataset so each .mat dataset only contains
% a single trial of information.  ScanMatch reads data one trial at a time.

% Change to correct working directory
cd('C:\Users\Mary.Frame\Documents\Sequence Alignment\ScanMatch Matlab
Code\ScanMatch');

% NOTE: Dimensions of the screen are 1280x1024

clear

% Read in full dataset if not already in individual files by trial
filename = 'AlyssaDataP17.xlsx';
data = xlsread(filename);
%Column 1 is time, column 5 is an index of event (for pre/post), Gaze data
%are columns 19-22 done separately
% X Y and Time, the values needed for output
data2 = data(2:end,[1 5 4 21:22]);

% This is an index of event numbers
eventindx = data(2:end, 5);
% To generate small .m datasets or variables to be used by ScanMatch
% algorithms, use a for loop to extract trial data

% Specify maximum trial number
maxtrial = 12;
%These are not used elsewhere in the code, these are merely checks for i5
%and i6  If either i5 or i6 is greater than or less than these boundary
%values (i1 and i2) then there is an error - mainly used for debugging
i1 = min(eventindx);
i2 = max(eventindx);

% No subject loop since only two subjecst - However, event loop very
% important, index of subsegmentation of trial
for tr = 1:maxtrial
    tr_str = num2str(tr);
    trindx = data2(:,1)==tr;
    i3 = find(trindx, 1, 'first');
    i4 = find(trindx, 1, 'last');
    datatr = data2(i3:i4,3:5);
    data3 = data2(i3:i4,:);
    i5 = min(data3(:,2));
    i6 = max(data3(:,2));
    count = 1;
    for event = i5:i6
        eventstr = num2str(event);
        % This indexes where any event occurs, iterated for each event
        evtindx = data3(:,2)==event;
        % This determines where the last event is, on final iteration this
        % will match evtindx but due to different number of events per
```

```
        % loop, this should be kept constant - cannot hard code event
        % numbers
        lastevt = data3(:,2)==i6;
        % The following two lines simply determine what row events occur on
        % so that they can be used for indexing later to subsegment the
        % data
        rowindx = find(evtindx);
        finalevtrow = find(lastevt);
        eventdata = datatr(count:rowindx,:);
        % This is where you specify the name of your output files by trial
        % Add subject number if that loop is used
        % This particular code is repeated due to differential naming
        % conventions
        savfile = ['Piasecki','S17Tr',tr_str,'preevent',eventstr,'.mat'];
        % This saves the data set by each trial
        save(savfile,'eventdata');
        % For each iteration of this loop will have correct start row
        count = rowindx+1;
        finalcount = count;
        finaldatatr = datatr;
    end
        postevents = finaldatatr(finalcount:end,:);
        % This particular code is repeated due to differential naming
        % conventions
        savfile2 = ['Piasecki','S17Tr',tr_str,'postevents.mat'];
        % This saves the data set by each trial
        save(savfile2,'postevents');
end

% Could do this for both subjects in a loop but want different naming
% conventions so will do as repeated code

% Read in full dataset if not already in individual files by trial
filename = 'AlyssaDataP21.xlsx';
data = xlsread(filename);
%Column 1 is time, column 5 is an index of event (for pre/post), Gaze data
%are columns 19-22 done separately
% X Y and Time, the values needed for output
data2 = data(2:end,[1 7 6 24:25]);

% This is an index of event numbers
eventindx = data(2:end, 7);
% To generate small .m datasets or variables to be used by ScanMatch
% algorithms, use a for loop to extract trial data

% Specify maximum trial number
maxtrial = 12;
%These are not used elsewhere in the code, these are merely checks for i5
%and i6  If either i5 or i6 is greater than or less than these boundary
%values (i1 and i2) then there is an error
i1 = min(eventindx);
i2 = max(eventindx);

% No subject loop since only two subjecst - However, event loop very
% important, index of subsegmentation of trial
for tr = 1:maxtrial
```

```
    tr_str = num2str(tr);
    trindx = data2(:,1)==tr;
    i3 = find(trindx, 1, 'first');
    i4 = find(trindx, 1, 'last');
    datatr = data2(i3:i4,3:5);
    data3 = data2(i3:i4,:);
    i5 = min(data3(:,2));
    i6 = max(data3(:,2));
    count = 1;
    for event = i5:i6
        eventstr = num2str(event);
        % This indexes where any event occurs, iterated for each event
        evtindx = data3(:,2)==event;
        % This determines where the last event is, on final iteration this
        % will match evtindx but due to different number of events per
        % loop, this should be kept constant - cannot hard code event
        % numbers
        lastevt = data3(:,2)==i6;
        % The following two lines simply determine what row events occur on
        % so that they can be used for indexing later to subsegment the
        % data
        rowindx = find(evtindx);
        finalevtrow = find(lastevt);
        eventdata = datatr(count:rowindx,:);
        % For each iteration of this loop will have correct start row
        count = rowindx+1;
        % This is where you specify the name of your output files by trial
        % Add subject number if that loop is used
        % This particular code is repeated due to differential naming
        % conventions
        savfile = ['Piasecki','S21Tr',tr_str,'preevent',eventstr,'.mat'];
        % This saves the data set by each trial
        save(savfile,'eventdata');
        postevents = datatr(count:end,:);
        % This particular code is repeated due to differential naming
        % conventions
        savfile = ['Piasecki','S21Tr',tr_str,'postevents.mat'];
        % This saves the data set by each trial
        save(savfile,'postevents');
    end
end
```

APPENDIX C: Piasecki Data Procedure Script

```
% Script Author: Mary Frame
% This code generates the comparisons reported in the Tech Report

cd('C:\Users\Mary.Frame\Documents\Sequence Alignment\ScanMatch Matlab
Code\ScanMatch\');

% Each file needs to be read into a variable - not all possible
% permutations tested for the purposes of the tech report

% The following lines load files and give each a variable name to be used
% by ScanMatch later during the comparison phase
% Single subject, Sub17, Trial 5, before first event
load('PiaseckiS17Tr5preevent29.mat'); S17T5Event29 = eventdata;

% Single subject, Sub17, Trial 5, after first event until 2nd event
load('PiaseckiS17Tr5preevent30.mat'); S17T5Event30 = eventdata;

% Single subject, Sub17, Trial 3, before first event
load('PiaseckiS17Tr3preevent22.mat'); S17T3Event22 = eventdata;

% Single subject, Sub17, Trial 3, after first event until 2nd event
load('PiaseckiS17Tr3postevents.mat'); S17T3PostEvents = postevents;

% Single subject, Sub21, Trial 5, before first event
load('PiaseckiS21Tr5preevent22.mat'); S21T5Event22 = eventdata;

% Single subject, Sub21, Trial 5, after first event until 2nd event
load('PiaseckiS21Tr5preevent23.mat'); S21T5Event23 = eventdata;

% Single subject, Sub21, Trial 3, before first event
load('PiaseckiS21Tr3preevent43.mat'); S21T5Event43 = eventdata;

% Single subject, Sub21, Trial 3, after first event until 2nd event
load('PiaseckiS21Tr3preevent44.mat'); S21T5Event44 = eventdata;


% This portion of the code may need to be repeated for testing a variety of
% parameterizations (i.e. higher vs lower thresholds or different
% resolutions)
ScanMatchInfo = ScanMatch_Struct()

% Tested 3 resolutions: 5x5, 10x10, 20x20
% Screen resolution was 1280x1024

% Comparisons of interest for ScanMatch

seq1 = ScanMatch_FixationToSequence(S17T5Event29, ScanMatchInfo); seq2 =
ScanMatch_FixationToSequence(S17T5Event30, ScanMatchInfo); seq3 =
ScanMatch_FixationToSequence(S17T3Event22, ScanMatchInfo); seq4 =
ScanMatch_FixationToSequence(S17T3PostEvents, ScanMatchInfo); seq5 =
ScanMatch_FixationToSequence(S21T5Event22, ScanMatchInfo);
```

```
seq6 = ScanMatch_FixationToSequence(S21T5Event23, ScanMatchInfo);
seq7 = ScanMatch_FixationToSequence(S21T5Event43, ScanMatchInfo);
seq8 = ScanMatch_FixationToSequence(S21T5Event44, ScanMatchInfo);

[score1 align1] = ScanMatch(seq1, seq2, ScanMatchInfo, 'ShowViewer', 1)
[score2 align2] = ScanMatch(seq3, seq4, ScanMatchInfo, 'ShowViewer', 1)
[score3 align3] = ScanMatch(seq1, seq3, ScanMatchInfo, 'ShowViewer', 1)
[score4 align4] = ScanMatch(seq2, seq4, ScanMatchInfo, 'ShowViewer', 1)

[score5 align5] = ScanMatch(seq5, seq6, ScanMatchInfo, 'ShowViewer', 1)
[score6 align6] = ScanMatch(seq7, seq8, ScanMatchInfo, 'ShowViewer', 1)
[score7 align7] = ScanMatch(seq5, seq7, ScanMatchInfo, 'ShowViewer', 1)
[score8 align8] = ScanMatch(seq6, seq8, ScanMatchInfo, 'ShowViewer', 1)

[score9 align9] = ScanMatch(seq1, seq5, ScanMatchInfo, 'ShowViewer', 1)
[score10 align10] = ScanMatch(seq3, seq7, ScanMatchInfo, 'ShowViewer', 1)
[score11 align11] = ScanMatch(seq2, seq6, ScanMatchInfo, 'ShowViewer', 1)
[score12 align12] = ScanMatch(seq4, seq8, ScanMatchInfo, 'ShowViewer', 1)
```