Behavior Research Methods
2010, 42 (3), 692-700
doi:10.3758/BRM.42.3.692

ScanMatch: A novel method for comparing fixation sequences

Filipe Cristino
University of Bristol, Bristol, England

Sebastiaan Mathôt and Jan Theeuwes
Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

and

Iain D. Gilchrist
University of Bristol, Bristol, England

We present a novel approach to comparing saccadic eye movement sequences based on the Needleman–Wunsch algorithm used in bioinformatics to compare DNA sequences. In the proposed method, the saccade sequence is spatially and temporally binned and then recoded to create a sequence of letters that retains fixation location, time, and order information. The comparison of two letter sequences is made by maximizing the similarity score computed from a substitution matrix that provides the score for all letter pair substitutions and a penalty gap. The substitution matrix provides a meaningful link between each location coded by the individual letters. This link could be distance but could also encode any useful dimension, including perceptual or semantic space. We show, by using synthetic and behavioral data, the benefits of this method over existing methods. The ScanMatch toolbox for MATLAB is freely available online (www.scanmatch.co.uk).

The last 30 years has seen an explosion of interest in eye movements as both a topic of study and a research tool for probing perceptual, cognitive, and neural processes (Findlay & Gilchrist, 2003; Rayner, 2009). With an increase in the availability of eyetrackers and a reduction in their cost, eye movement studies have flourished not only in vision research (e.g., Parkhurst, Law, & Niebur, 2002; Tatler, Baddeley, & Gilchrist, 2005) but also in language (see Rayner, 1998, 2009), advertising (e.g., Maughan, Gutnikov, & Stevens, 2007; Pieters, Rosbergen, & Wedel, 1999), clinical (e.g., Mosimann et al., 2005), and animal (e.g., Kirchner & Thorpe, 2006) research. Although these fields differ widely in their aims and goals, one common denominator among them is the techniques used to study and analyze the eye movements.

The majority of studies in this area are concerned with the saccadic eye movement response. Saccadic eye movements are the fast ballistic movements in which the eye moves rapidly from one location to another. Between saccades, the eye is stationary; these periods of time are referred to as *fixations*, and it is during these time intervals that useful visual information is gathered (see Carpenter, 1988, for more details).

Despite the growing interest in saccades and fixations, most methods of analysis rely on saccade latencies (time to initiate a saccade; see, e.g., Zingale & Kowler, 1987) or the fixation locations as primary measures of behavior. In the vast majority of the literature on eye movements, saccade amplitude or duration, number of fixations, fixation durations, or other close derivatives have been used as the main measures. However, despite saccades being fundamentally sequential (one saccade is followed by the next one), very few methods are available for studying their sequential properties. Markov models (Markov, 1971) have been used rather successfully in some domains, such as eye movement modeling in reading (Engbert & Kliegl, 2001), but implementing them is complex. Another, more widely applied method, which takes into account fixation order, is Levenshtein distance (Levenshtein, 1966), more commonly referred to as the *string edit method*.

Levenshtein distance measures the editing cost of transforming one string into another one, using, in its basic form, a set of three operations (insertion, deletion, and substitution), with a cost of one for each operation. The Levenshtein distance metric is given by the minimal score or minimum editing cost between two strings. More advanced versions can be found, either by using transposition as an extra operation (Wagner & Lowrance, 1975) or by assigning different weights to each operation (Okuda, Tanaka, & Kasai, 1976). Such methods have been successfully used by a number of authors to study saccade sequences (e.g., Brandt & Stark, 1997; Choi, Mosley,

I. D. Gilchrist, i.d.gilchrist@bristol.ac.uk

& Stark, 1995; Foulsham & Underwood, 2008; Hacisalihzade, Allen, & Stark, 1992; Noton & Stark, 1971). The method involves defining a number of spatial regions of interest (ROIs) in the scene being scanned and recoding the fixation sequence as a series of letters representing the fixated locations. Although the string edit method has proven to be a useful tool and is relatively fast to compute, one of its main drawbacks is the lack of relationship between ROIs. As a result, the algorithm lacks flexibility, since it cannot differentiate between close and distant regions or, more generally, between similar and dissimilar ROIs. A second drawback of this kind of method is that it does not take into account fixation duration; all fixations, however short or long, are treated equally. It is clear that fixation duration is an important indicator of processing during a fixation (Henderson & Pierce, 2008).

In this article, we describe a new method, used for nearly 4 decades in bioinformatics to analyze biological sequences (either DNA or protein sequences), for quantitatively scoring two eye movement sequences. The resemblance between both problems is striking; from the early days in gene sequence analysis, it was discovered that sequences from related genes could be classified as similar if a maximum of residues (string elements) matched along the sequence (Durbin, Eddy, Krogh, & Mitchison, 1998). This was a very important discovery, since close similarity between two genes or proteins is a strong argument for their homology (common ancestry). To solve the string-matching problem, collaboration between computer scientists, mathematicians, and biologists resulted in a large number of optimized methods called *sequence alignment algorithms*. These methods are optimized to the extreme because, in this field, it is often necessary to compare millions of sequences together (i.e., in the Human Genome project). The methods developed in this area range from robust and accurate but slow, such as the Needleman–Wunsch algorithm (Needleman & Wunsch, 1970), to very fast but without the guarantee of finding the optimal solution, such as the BLAST algorithm (Altschul et al., 1997). In this article, we show how the methodology of sequence alignment (Needleman–Wunsch algorithm) can be applied to eye movements and then present three experiments in which the method (which we have called *ScanMatch*) is used. The ScanMatch application was coded in MATLAB with the bioinformatics toolbox, and the code is freely available online for downloading or can be requested directly from the authors.

GENERAL METHOD

Creating a Sequence

Eye movement data collected through any eyetracker can be used to create a sequence of eye movements to which the ScanMatch method can be applied. First, the data need to be filtered down to saccades and fixations. Then, to be able to code string sequences, images have to be divided into ROIs, which either can be designated feature regions within an image (e.g., eyes, mouth, and nose in a face image; doors, windows, and people in an interior scene) or can be created by simply binning the image into a discrete number of regular bins. A letter is then assigned to each region, and every eye fixation within that region is tagged with its name. One problem with previous string-based methods was the lack of coding of fixation duration. Here, to take fixation time into account, we introduce temporal binning into the string by repeating the letter corresponding to the ROI in a way that is proportional to the fixation duration. In this way, the string created by the saccade sequence incorporates spatial location, sequential information, and temporal durations (see Figure 1). In Experiments 2 and 3, we set this temporal sampling to 50 msec, since this value allows an accurate sampling of the large variability of fixation times (typically, 100–1,000 msec).

Another practical factor that can become an issue when strings are used to represent eye movement sequences is that, in some circumstances, one may need more than 26 ROIs (the length of the English alphabet). To avoid this, a

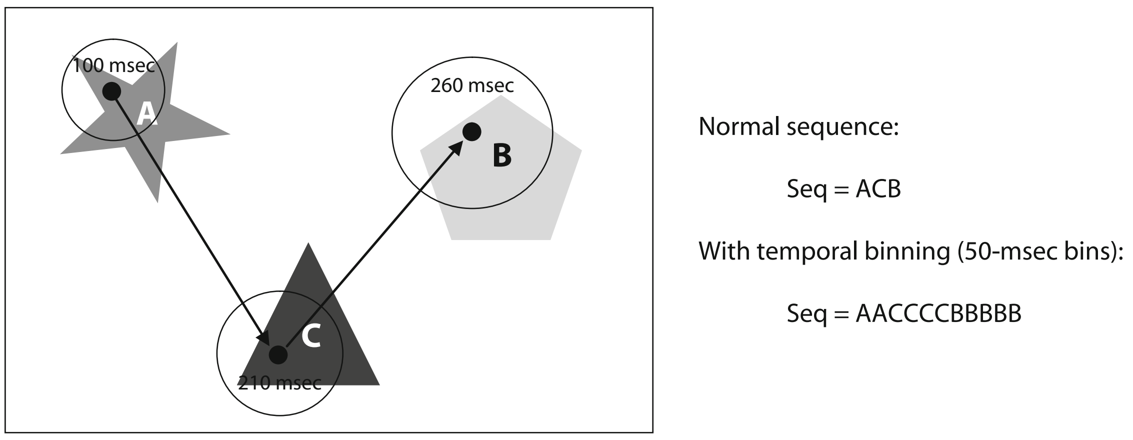


Figure 1. From fixations to eye movement sequences. Each fixation is given the letter of the region of interest (ROI) where it landed (in this example, ACB). To take into account fixation durations, dwell times can be sampled (in this example, with 50-msec bins), with each sample given the ROI letter of the fixation, resulting in the following sequence: AACCCCBBBBB.

double-string procedure was implemented, in which each region is coded by two letters; for ease of reading, the first letter is in lowercase, and the second in uppercase. The second letter does not need to go up Z. If, for example, an image is divided into 12 × 2 bins, one may choose to name the first-row bins aA to aL, and the second row bA to bL (hence using a modulus of 12). As part of the ScanMatch software, a MATLAB procedure is provided to create double strings from ROI numbers that takes into account the modulus.

Comparing Strings

The aim of the method presented in this article is to quantitatively describe how similar two eye movement sequences are (i.e., "aAaBaCaD" with "aAaCaCaD" or "aEaAaTaS"). For this, we used a family of algorithms widely used in bioinformatics called *sequence alignment* algorithms.

Sequence Alignment

Global sequence alignment algorithms seek the optimal alignment over the entire length of two sequences by maximizing its score. The sequences are aligned according to a substitution matrix that gives scores for every alignment (i.e., aligning aA with aB may give a score of 5, whereas aligning aA with aK gives a score of 2; and aligning an aA with an aA may give the maximum score of 10). To be able to perform the sequence alignment, gaps (spaces between two residues) can be introduced in either sequence at a certain cost (i.e., −1). Gap-to-gap alignments are not allowed, since they are redundant. The implementation of such an algorithm has being extensively researched in the computational biology field, with one of the most popular implementations being BLAST (Altschul et al., 1997), the method used for the Human Genome project. The implementation of this method has been very successful, but it prioritizes speed over accuracy because it is intended to be used for the analysis of enormous data sets (millions of entries). For the purpose of this article, we chose the Needleman–Wunsch algorithm, a slower but far simpler method, which is guaranteed to find the optimal solution.

Needleman–Wunsch Algorithm

The Needleman–Wunsch algorithm uses dynamic programming to solve the alignment problem. The basic idea is that it uses local optimal alignment of sub-sequences to create the best overall alignment. By doing this, it reduces the number of possibilities to be considered but still guarantees the optimal solution. The algorithm is divided into two parts; the first creates a matrix with all scoring possibilities based on a substitution matrix and a gap penalty, and the second seeks the optimal alignment, tracing back the matrix from the top left corner to the outmost column or row by selecting the optimal route (route adding up to the maximum score). The score is given by the optimal route throughout the matrix; the higher the score, the more likely it is that the two sequences are similar. In this algorithm, only two parameters need to be set: the substitution matrix and the gap penalty.

The substitution matrix. The substitution matrix provides the algorithm with a score (the higher the better) for aligning two letters. As was previously discussed, one of the main drawbacks of the string edit method is the lack of relationship between the ROIs. In the present method, this problem is addressed with the substitution matrix, since it can encode information about the relation between each ROI. In bioinformatics, several matrices are available, with the most popular being PAMxx and BLOSUMxx (Mount, 2008). PAMxx provides scores based on the observed frequencies of alignments in related proteins (xx meaning up to xx% of divergence between two genes—i.e., xx = 50), where identities are given the highest scores (frequently observed substitutions are given a positive score, and rarely observed substitutions a negative score). Using similar logic, for sequences of fixations, within the substitution matrix, the relationships between ROIs in an image can be coded in a number of ways. A basic relationship is the one based on distance, with scores in the substitution matrix inversely related to the Euclidean distance between bins and with the highest score given to the identities and the lowest to the bins the furthest apart. However, the relationship does not have to be spatial; for example, a relationship can be based on color (e.g., in a visual search task) or a more semantic segmentation of an image (e.g., clothes or toys in some interior scene). One important parameter when constructing the substitution matrix is the cutoff point of what should be positive (highly related items) and what should be negative (loosely related items). In the case of eye movements, where the substitution matrix is based on distance between ROIs, one could use the variability in saccade landing to decide the cutoff point at which values in the substitution matrix should become negative (threshold). With this method, for the distance-based substitution matrix, this means that the aim of the alignment algorithm will be to align only regions that are within the variability of saccade landing. We therefore define this threshold as 2 standard deviations of all the saccade amplitudes in a particular experiment from which the sequences are compared. A graphical user interface for constructing and visualizing a substitution matrix based on distance is provided with the ScanMatch toolbox.

The gap penalty. The gap penalty is the second and last free parameter for the string alignment algorithm and is directly related to the substitution matrix. Its value is the score for aligning any element in a sequence with a gap (a free space). A large gap penalty will favor the insertion of many gaps and will, therefore, discourage the alignment of less highly related residues, whereas a small or negative gap penalty will favor the alignment of residues (even if only loosely related) and discourage gaps. One could see the gap penalty as the limit for which two residues should not be aligned but, instead, a gap created. Figure 2 describes a simple example using two different values for the gap penalty.

As it can be seen in Figure 2, the gap penalty acts as a similarity threshold, below which the insertion of a gap is favored over alignment. In longer sequences, gaps may

```
Gap penalty = 0      aA__aB
                     |  | |      Score = 10 + 0 + 0 = 10
                     aAaC__

Gap penalty = –2     aAaB
                     |  |        Score = 10 + (–1) = 9
                     aAaC
```

Substitution matrix

	aA	aB	aC
aA	10	–1	–5
aB	–1	10	–1
aC	–5	–1	10

Figure 2. Simple case of comparing two short sequences, with, on the right, an example of a substitution matrix to demonstrate the effect of the gap penalty.

be inserted if this benefits the global alignment of the sequences, even though this may result in suboptimal local alignments. If the threshold in the substitution matrix is well chosen, as was previously explained, the gap value can be set to zero. In the three experiments described in this article, the gap value is always set to zero.

The scoring. The alignment score given by the Needleman–Wunsch algorithm is highly dependent on the substitution matrix and on the length of the sequences compared. Indeed, comparing two identical strings with 20 residues will naturally give a higher score than will two identical sequences with 2 residues. To overcome these issues, a normalization of the score is performed:

$$\text{Normalized score} = \frac{\text{score}}{\text{Max(substitution matrix)} * \text{length of the longest sequence}}.$$

Using this normalization, the best possible match between two sequences will give a score of 1. Examples of sequence comparisons can be found online on the ScanMatch Web page as part of a ScanMatch toolbox tutorial.

MATLAB Code

The ScanMatch software is available for download from www.scanmatch.co.uk or can be requested directly from the authors. It provides a set of MATLAB script files that allow the comparison of the sequences. The Needleman–Wunsch algorithm implementation under MATLAB was taken from the bioinformatics toolbox. Although the ScanMatch program does not need the bioinformatics toolbox to run, a licence for this toolbox should be held. A user interface (Figure 3) is provided to facilitate visualization of the algorithm. The MATLAB files are annotated, and a tutorial on how to use the toolbox can be found on the ScanMatch Web page.

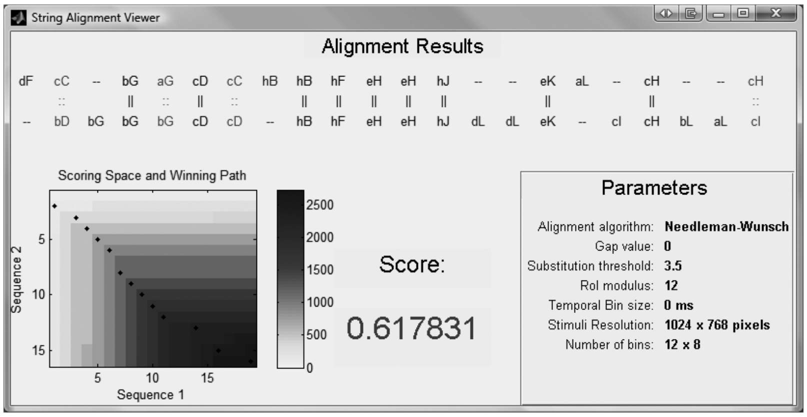
To test our new ScanMatch method, we used three data sets: a synthetic one, which, by varying a set of parameters, enabled us to observe how well the algorithm behaved, and two data sets taken from simple eyetracking experiments.

EXPERIMENT 1

This first experiment was based on synthetic data to demonstrate the power of the proposed approach in describing eye movement sequences and to compare our results against the well-established Levenshtein measure (string edit distance).

Method and Results

We simulated a simple experiment in which saccades were generated between 26 linearly arranged tiles. The starting point for scanning was either from the left (from tile aG to tile aT) or from the right (tile aT to tile aG), as described in Figure 4. We predicted that this would result in two different types of eye movement sequences, vary-



Alignment Results

```
dF  cC  --  bG  aG  cD  cC  hB  hB  hF  eH  eH  hJ  --  --  eK  aL  --  cH  --  --  cH
    ::      ||  ::  ||  ::      ||  ||  ||  ||  ||          ||          ||          ::
--  bD  bG  bG  bG  cD  cD  --  hB  hF  eH  eH  hJ  dL  dL  eK  --  cI  cH  bL  aL  cI
```

Score: 0.617831

Parameters

Alignment algorithm:	**Needleman-Wunsch**
Gap value:	**0**
Substitution threshold:	**3.5**
RoI modulus:	**12**
Temporal Bin size:	**0 ms**
Stimuli Resolution:	**1024 x 768 pixels**
Number of bins:	**12 x 8**

Figure 3. The ScanMatch graphical user interface (V1.00) under MATLAB. It gives a visual representation of the string alignment method.

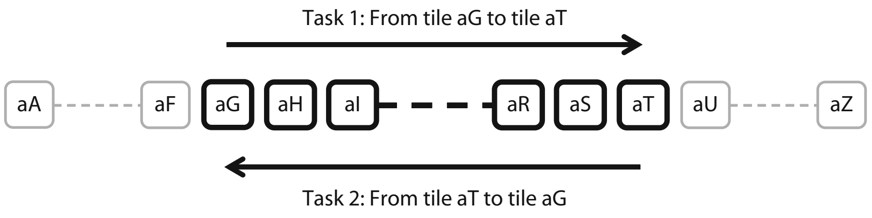

Task 1: From tile aG to tile aT
aA
aF
aG
aH
aI
aR
aS
aT
aU
aZ
Task 2: From tile aT to tile aG

Figure 4. Experiment 1: Stimuli and task. Twenty-six tiles are arranged linearly (from aA to aZ), and the two simulated tasks were to go from tile aG to aT (for Task 1) or to go from tile aT to tile aG (for Task 2), fixating each tile in between.

ing only temporally (order of the visited tiles), but with the same spatial statistics (with each tile being visited the same number of times in whichever direction the task was performed).

String sequences were created as described in Figure 5, where, by varying a single parameter (σ), we were able to create sets of sequences that varied in similarity. For this simple simulation, we assumed a constant fixation time; hence, each simulated fixation is represented by a single tile.

Strings are created by sampling a normal distribution centered on each tile to be visited (tile aG to aT) and with a standard deviation σ. If a sample is outside the range of tiles (lower than aA or greater than aZ), it will be deleted and not replaced. In such a case, the sequence will have fewer letters. If the standard deviation of the distribution (σ) is equal to zero, each string will be 12 letters long in a perfect order. As σ increases, the strings will diverge more and more, and there will be more variability in the length of the sequences (Figure 6A). In this experiment, a total of 24 sets of strings for each σ level were created, each set containing 100 sequences for the first task (going from tile aG to aT) and 100 for the second (going from tile aT to aG). We chose these numbers to match the numbers of trials in a typical experiment. The parameter σ was varied from 0 to 14 by steps of 0.5. Figure 6A shows the mean string length of all the generated strings across the different σ levels (error bars represent 95% confidence intervals), and Figure 6B shows the mean distance between two consecutive string residues (equivalent to saccade amplitude in a real experiment) in generated strings. The bigger the spread of the distribution is, the greater the variation of the strings in content and in size.

The substitution matrix used here was based on distance as outlined in the previous session and created with a standard deviation of 3. This value was the mean standard deviation of the distance between two consecutive residues across all noise conditions. The same substitution matrix was used to compare all strings across all the levels. In this experiment, the sequences were compared using our new ScanMatch method, as well as using the classic Levenshtein distance method (a single-character string was used, since the number of ROIs was under 26).

Sequences of each set (Task 1 or 2) were compared with each other; that is, strings from the first set were compared

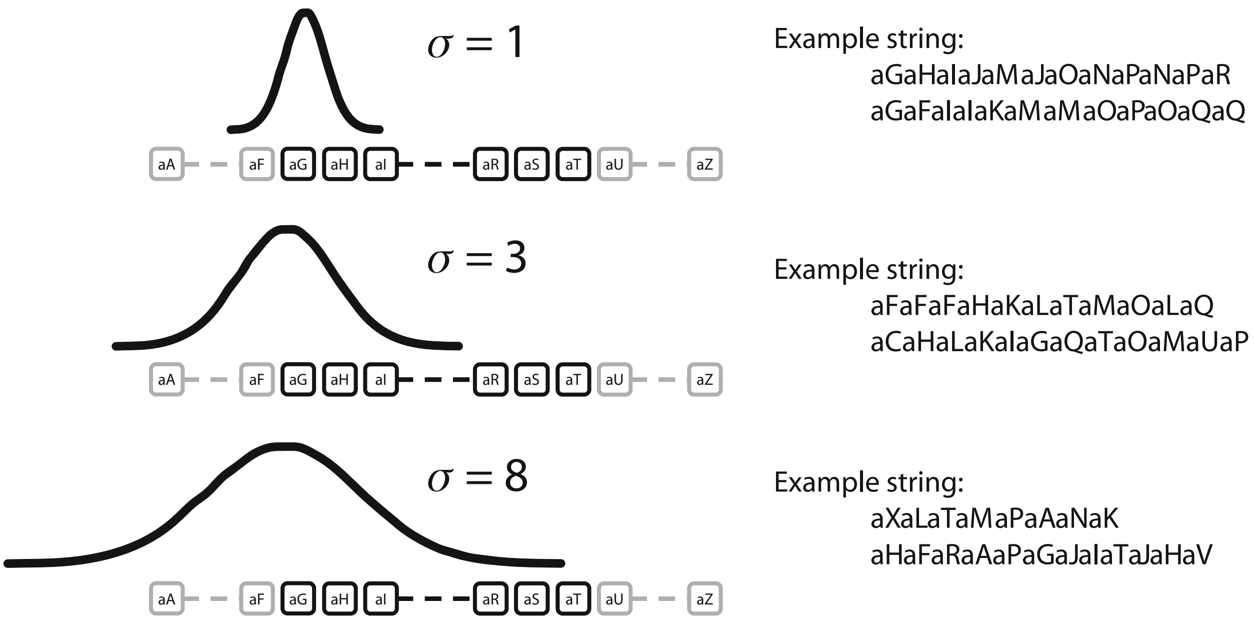


Figure 5. Experiment 1: Sequence creation. As the standard deviation of the polling distribution increases, the variability in landing increases, making the generated sequences more and more different. If a saccade lands outside the 26 tiles (smaller than aA or greater than aZ), this sample is deleted and not replaced (hence the likelihood that the sequences will get shorter as the noise increases).

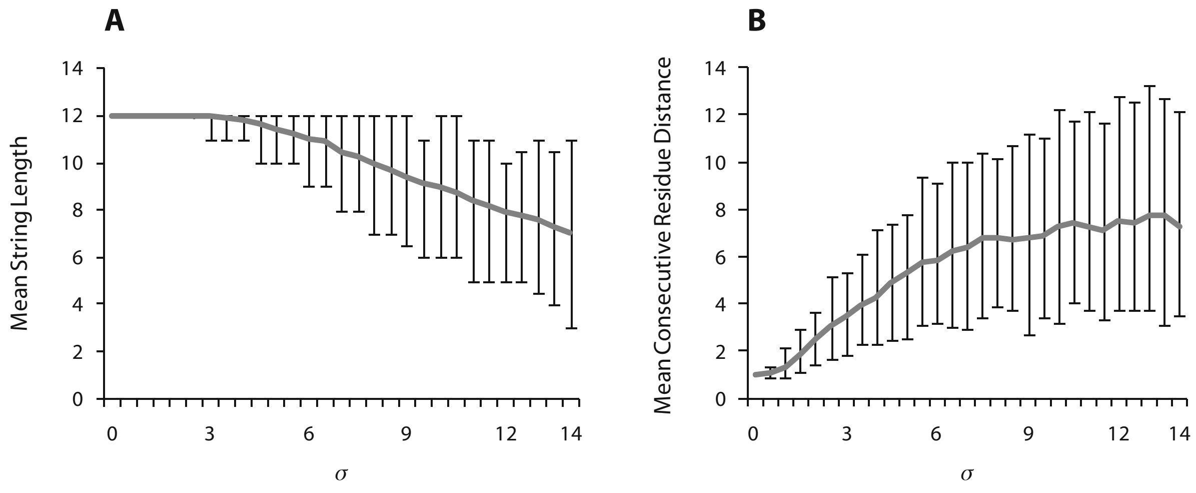


Figure 6. Effect of σ on the generated sequences. (A) As the standard deviation of the distribution increased, the mean string length of a sequence decreased, allowing different-length sequence comparisons (error bars represent 95% confidence intervals). (B) The distance between two consecutive residues increased as σ got higher (i.e., aA to aB had a distance of 1, whereas aA to aD had a distance of 4).

with all the other strings in that set and also with strings in the second set. It follows, therefore, that comparisons within sets should lead to higher scores, whereas comparisons across sets should lead to lower scores. A *k*-mean clustering algorithm was then used to classify each comparison set to either the first set (a within-sets comparison) or the second set (a between-sets comparison). The data were analyzed by measuring the percentage of incorrect classifications (with chance giving a misclassification of 50%). Figure 7 shows the mean percentages of misclassifications for the 24 sets, where the error bars represent the 95% confidence intervals.

Our new ScanMatch method performs very well even with noisy data, performing at chance only when σ is greater than 10. It successfully classifies sequences without a single error up to a noise level of 2, whereas the Levenshtein distance method starts to misclassify as soon as noise is added ($\sigma > 0.5$). At this level of noise, the string edit method already misclassified 18% of the trials, since the sequences could be very different with that amount of noise.

ScanMatch performs significantly better than the Levenshtein distance method up to an added noise level with a standard deviation of 8.5 (nonoverlapping error bars in Figure 7), at which point the string edit method is already performing at chance. This experiment shows how well our new method behaves even with a higher level of divergence in the input sequences.

On a modern personal computer system (Intel Quad core clocked at 3.0 GHz with 4 GB of RAM running under Vista x64 with MATLAB 2009a on a single core), the ScanMatch method took less than 2 min to perform the

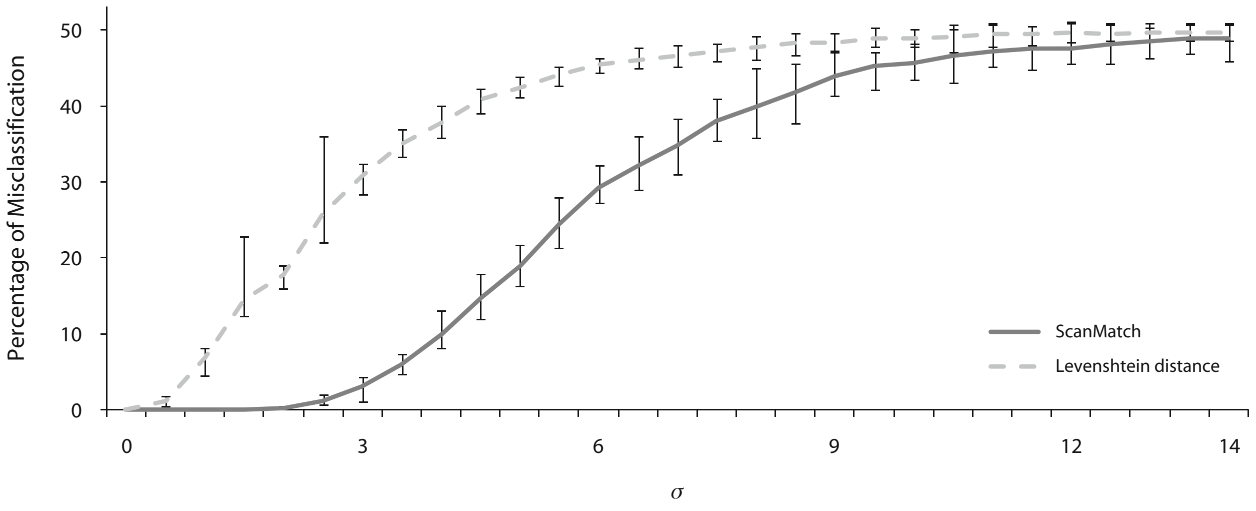


Figure 7. Experiment 1: Results. A *k*-mean clustering algorithm was used to classify the sequences per task. ScanMatch outperforms the Levenshtein distance up to a very high level of noise (8), with no misclassifications up to a σ level of 2.5.

100,000-sequence comparison needed to compare a block of 100 sequences.

EXPERIMENT 2

In this experiment, we assessed the performance of the new method, using a simple sequential-looking task with human eye movement data.

Method and Results

The stimuli, displayed on a computer screen, were composed of red and green numbers ranging from 1 to 9 (1.3 × 1.9 visual degrees) randomly placed on the screen. A single participant (one of the authors, S.M.) was asked on a given trial to saccade from number to number in ascending (from 1 to 9) or descending (from 9 to 1) order and to follow either the green or red numbers while their eye movements were recorded. Instructions were given to the participant before each trial, explaining which color and which order to follow. In a single block, the participant performed the 20 trials in random order (4 conditions performed 5 times). Eye movements were recorded using an Eyelink II, produced by SR Research. The EyeLink II is a head-mounted binocular eyetracking device with a sampling rate of 500 Hz. The stimuli were displayed on a CRT screen with a screen resolution set to 1,024 × 768 pixels. A 9-point calibration was performed before the start of the experiment, and a drift correction before each trial. Saccade extraction was done using an SR Research normal saccadic filter with values of 30º/sec velocity threshold and 8,000º/sec^2 acceleration threshold. The participant used a chinrest situated 66 cm from the display screen.

In this experiment, since the stimuli were constant across task instructions, the eye movements were binned spatially with a grid size of 12 × 8, with a label starting from aA to aL on the first line to hA to hL on the last. Each eye movement in a bin was given its label, creating a total of 20 sequences. The substitution matrix was based on the distance between each bin (as described above, with a cutoff value of 5.5) and with a gap value of 0.

Each sequence was compared with every other sequence using our ScanMatch method, creating a 20 × 20 score matrix. The eye movement sequences of a participant performing in the same condition should be more similar than when he or she performs in a different condition. To check this hypothesis, a *k*-means algorithm was used to cluster the scoring matrix for each comparison set (a single trial against all of the others). The self-comparison (leading to a perfect score of 1) was removed from the clustering process. The mean number of saccades per trial was 18.55 (*SD* = 2.18), with a mean amplitude of 6.5º (*SD* = 11.5). Note that this *SD* value is the one used to construct the substitution matrix.

Comparisons of same-condition trials scored a mean normalized score of .74 (*SD* = .034), whereas comparisons of different-condition trials scored a mean normalized score of .15 (*SD* = .01), a significantly lower score [$F(1,32) = 7{,}592$, $p < .001$]. Trials were perfectly clustered with not a single error. Even though the spatial statistics between color conditions were the same (the participant had to fixate each number the same number of times in order to execute the task), using the temporal aspect of the sequences, ScanMatch clustered all the trials successfully. This experiment shows the effectiveness of our new method when used with ROIs created by binning the image and a substitution matrix based purely on distance.

EXPERIMENT 3

In this experiment, we collected human eye movement data from a number of participants on a visual search task and investigated whether the new method would be able to differentiate between conditions.

Method and Results

In this visual search task, participants had to report the orientation of a letter T, tilted 90º, while their eye movements were recorded. As soon as they located the target, the participants were asked to press either the left or the right arrow key to report the target's orientation.

The randomly generated stimuli consisted of a target and 19 distractors. These were a combination of Ls in various orientations, 10 of which were a different color from the target (red or green) and 9 of which were the same color. The letters had a size of 0.375 visual degrees and were separated by a gap of at least 2.8º. The monitor was calibrated to display the red and green letters with the same luminance. The participants knew the color of the target before each trial, and the same stimuli were presented across participants but in a different order. Eye movement recording methods and display characteristics were the same as those in Experiment 2.

Eight participants took part in this experiment, performing 200 trials each (100 searching for a red target and 100 for a green target). All had normal or corrected-to-normal vision. Since the spatial locations of the target and distractors were different from trial to trial, the ROIs were based on the color and orientation of the Ls. The ROIs created were circular, centered on the middle of the letter, with a diameter of 80 pixels (2.8º). In this experiment, to demonstrate the flexibility of our algorithm, we used a substitution matrix in which we coded color as our main measure. We therefore coded a substitution of a green object with a green object as a high value (in the case of searching for a green target), and a substitution with a red object as a low value (any substitution with the background or the starting point was coded to 1).

In this experiment, the gap value was left at zero. The analysis was carried out on a per participant basis, where every eye movement sequence within the participant was compared with every other sequence within and across tasks (the hypothesis being that eye movement sequences in which the same-color target was searched for would be more similar than sequences in which a different-color target was searched for). We therefore ran the ScanMatch method twice: once with the green substitution matrix, where compared trials in which the green target was searched for should score highly, and once with the red substitution matrix, where compared trials in which the red target was searched for should score highly.

Using our ScanMatch method, trials compared within each task had a significantly higher normalized score (red trials, $M = .68$, $SD = .02$; green trials, $M = .66$, $SD = .013$) than did trials compared across tasks (red trials, $M = .26$, $SD = .013$; green trials, $M = .25$, $SD = .014$), and this difference was highly reliable [$F(1,28) = 5{,}793$, $p < .001$]. The clustering was very successful, with 95% ($M = .95$, $SD = .03$) of red and 94% ($M = .94$, $SD = .05$) of green trials correctly clustered. This experiment demonstrates the success and potential of a substitution matrix in which a nonspatial (in this case, perceptual) dimension can be coded.

DISCUSSION

We have presented a new method—ScanMatch—for comparing sequences of fixation generated by eye movements. The method addresses two significant limitations of the previously used sting edit methods. First, the method allows the durations of the fixations to be taken into account and included in any assessment of the similarity between two fixation sequences. Second, the method allows identified ROIs to be classified. At its most simple, this means that the spatial distance between ROIs can be taken into account such that ROIs that are close to each other can be considered "less different" than ROIs that are farther apart. Importantly, this method allows ROIs to be classified as more similar, or different, on any dimension, be it cognitive, perceptual, or task based.

We have presented three experiments that tested the performance of these methods as applied to fixation sequences. In the first experiment, in which we used synthetic data, we made a direct comparison between string edit methods and our proposed new method. This was possible here because we assumed that all fixation durations were the same, and it was only under these circumstances that we could compare the two methods directly. The ScanMatch method reliably outperformed the string edit method at identifying the original simulated groups, and this was particularly true as more noise was introduced into the groups to make the task of differentiating between the groups harder.

In the subsequent experiments, we showed the ability of the method to deal with real eye movement data and differentiate between conditions in a robust manner. The method was stable and reliable, both for a small data set from a single participant (Experiment 2) and for data sets from multiple participants of the size that occurs in a typical experiment (Experiment 3). We demonstrated that the methods could code similarity between regions in spatial coordinates (Experiment 2), as well as in perceptual coordinates—in this case, color (Experiment 3).

At the heart of this new method is the Needleman–Wunsch algorithm, which has been developed to be computationally efficient without compromising on the quality of the fit between strings. As a result, the comparison of large numbers of strings is possible on a modern desktop computer.

The sequence of fixations is a complex data structure, and care needs to be taken, with the application of this and other methods, to be explicit about the assumptions that underlie the similarity calculation to be computed. One of the advantages of the ScanMatch method is that these assumptions are explicitly held within a single matrix (the substitution matrix). However, this matrix is simple in that it encodes a single substitution value between two ROIs. The assumption here is that this value is stable across the experiment and, therefore, will not, for example, change over time. This may be a particularly important consideration when these methods are used for experiments that involve learning.

These methods are particularly strong when the ordering of the fixations may be an important determinant of a change in the behavior of the participant. However, if the sequence in which the information is encoded is not important and, so, different fixation orders will not have an effect on cognition or perception, these methods may not be as useful.

In conclusion, our new method, ScanMatch, can differentiate between conditions and is robust to the substantial noise inherent in fixation sequences. As a method, it opens up the possibility of studying in far more detail and with more precision how eye movements influence performance in complex tasks in which a number of eye movements are generated in a sequence.

AUTHOR NOTE

This work was supported by the NWO and ESRC Bilateral Grant Scheme. Correspondence concerning this article should be addressed to I. D. Gilchrist, Department of Experimental Psychology, University of Bristol, 12A Priory Road, Bristol BS8 1TU, England (e-mail: i.d.gilchrist@bristol.ac.uk).

REFERENCES

Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, **25**, 3389-3402.

Brandt, S. A., & Stark, L. W. (1997). Spontaneous eye movements during visual imagery reflect the content of the visual scene. *Journal of Cognitive Neuroscience*, **9**, 27-38.

Carpenter, R. H. S. (1988). *Movements of the eyes* (2nd ed.). London: Pion.

Choi, Y. S., Mosley, A. D., & Stark, L. W. (1995). String editing analysis of human visual search. *Optometry & Vision Science*, **72**, 439-451.

Durbin, R., Eddy, S. R., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge: Cambridge University Press.

Engbert, R., & Kliegl, R. (2001). Mathematical models of eye movements in reading: A possible role for autonomous saccades. *Biological Cybernetics*, **85**, 77-87.

Findlay, J. M., & Gilchrist, I. D. (2003). *Active vision: The psychology of looking and seeing.* Oxford: Oxford University Press.

Foulsham, T., & Underwood, G. (2008). What can saliency models predict about eye movements? Spatial and sequential aspects of fixations during encoding and recognition. *Journal of Vision*, **8**(2, Art. 6), 1-17.

Hacisalihzade, S. S., Allen, J. S., & Stark, L. (1992). Visual perception and sequences of eye movement fixations: A stochastic modeling approach. *IEEE Transactions on Systems, Man, & Cybernetics*, **22**, 474-481.

Henderson, J. M., & Pierce, G. L. (2008). Eye movements during scene viewing: Evidence for mixed control of fixation durations. *Psychonomic Bulletin & Review*, **15**, 566-573.

Kirchner, H., & Thorpe, S. J. (2006). Ultrarapid object detection with saccadic eye movements: Visual processing speed revisited. *Vision Research*, **46**, 1762-1776.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics–Doklady*, **10**, 707-710.

Markov, A. (1971). Extension of the limit theorems of probability theory to a sum of variables connected in a chain. In *Dynamic probabilistic systems: Vol. I. Markov models* (pp. 552-577). New York: Wiley.

Maughan, L., Gutnikov, S., & Stevens, R. (2007). Like more, look more. Look more, like more: The evidence from eye-tracking. *Journal of Brand Management*, **14**, 335-342.

Mosimann, U. P., Muri, R. M., Burn, D. J., Felblinger, J., O'Brien, J. T., & McKeith, I. G. (2005). Saccadic eye movement changes in Parkinson's disease dementia and dementia with Lewy bodies. *Brain*, **128**, 1267-1276.

Mount, D. W. (2008). Comparison of the PAM and BLOSUM amino acid substitution matrices. *Cold Spring Harbor Protocols*, 2008:7. doi:10.1101/pdb.ip59

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**, 443-453.

Noton, D., & Stark, L. (1971). Scanpaths in eye movements during pattern perception. *Science*, **171**, 308-311.

Okuda, T., Tanaka, E., & Kasai, T. (1976). A method for the correction of garbled words based on the Levenshtein metric. *IEEE Transactions on Computers*, **C-25**, 172-178.

Parkhurst, D., Law, K., & Niebur, E. (2002). Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, **42**, 107-123.

Pieters, R., Rosbergen, E., & Wedel, M. (1999). Visual attention to repeated print advertising: A test of scanpath theory. *Journal of Marketing Research*, **36**, 424-438.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, **124**, 372-422.

Rayner, K. (2009). Eye movements and attention in reading, scene perception and visual search. *Quarterly Journal of Experimental Psychology*, **62**, 1457-1506.

Tatler, B. W., Baddeley, R. J., & Gilchrist, I. D. (2005). Visual correlates of fixation selection: Effects of scale and time. *Vision Research*, **45**, 643-659.

Wagner, R. A., & Lowrance, R. (1975). An extension of the string-to-string correction problem. *Journal of the ACM*, **22**, 177-183.

Zingale, C. M., & Kowler, E. (1987). Planning sequences of saccades. *Vision Research*, **27**, 1327-1341.

(Manuscript received September 22, 2009;
revision accepted for publication January 22, 2010.)