multimatch Documentation

Release 0.1.0

Adina Wagner

Dec 26, 2020

Contents:

CHAPTER 1

Algorithm

The **MultiMatch** method is a vector-based, multi-dimensional approach to compute scan path similarity. It was originally proposed by Jarodzka, Holmqvist & Nyström (2010) and implemented as a Matlab toolbox by Dewhursts and colleagues (2012).

The method represents scan paths as geometrical vectors in a two-dimensional space: Any scan path is build up of a vector sequence in which the vectors represent saccades, and the start and end position of saccade vectors represent fixations. In the example image below, the scan path is build by connecting the fixations (red dots) with vectors (black lines), which constitute simplified saccades.

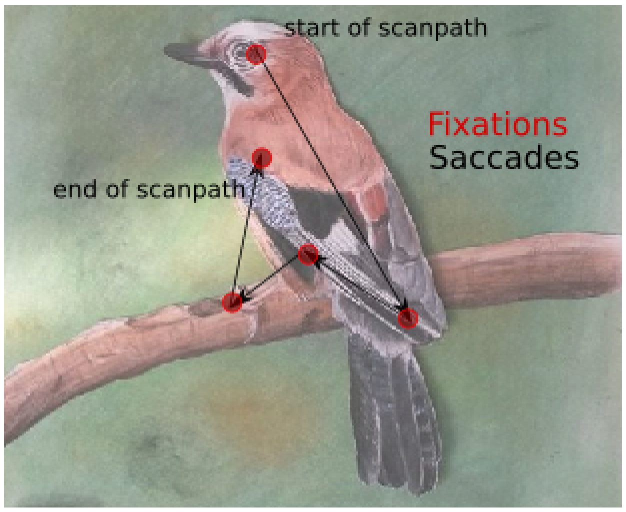
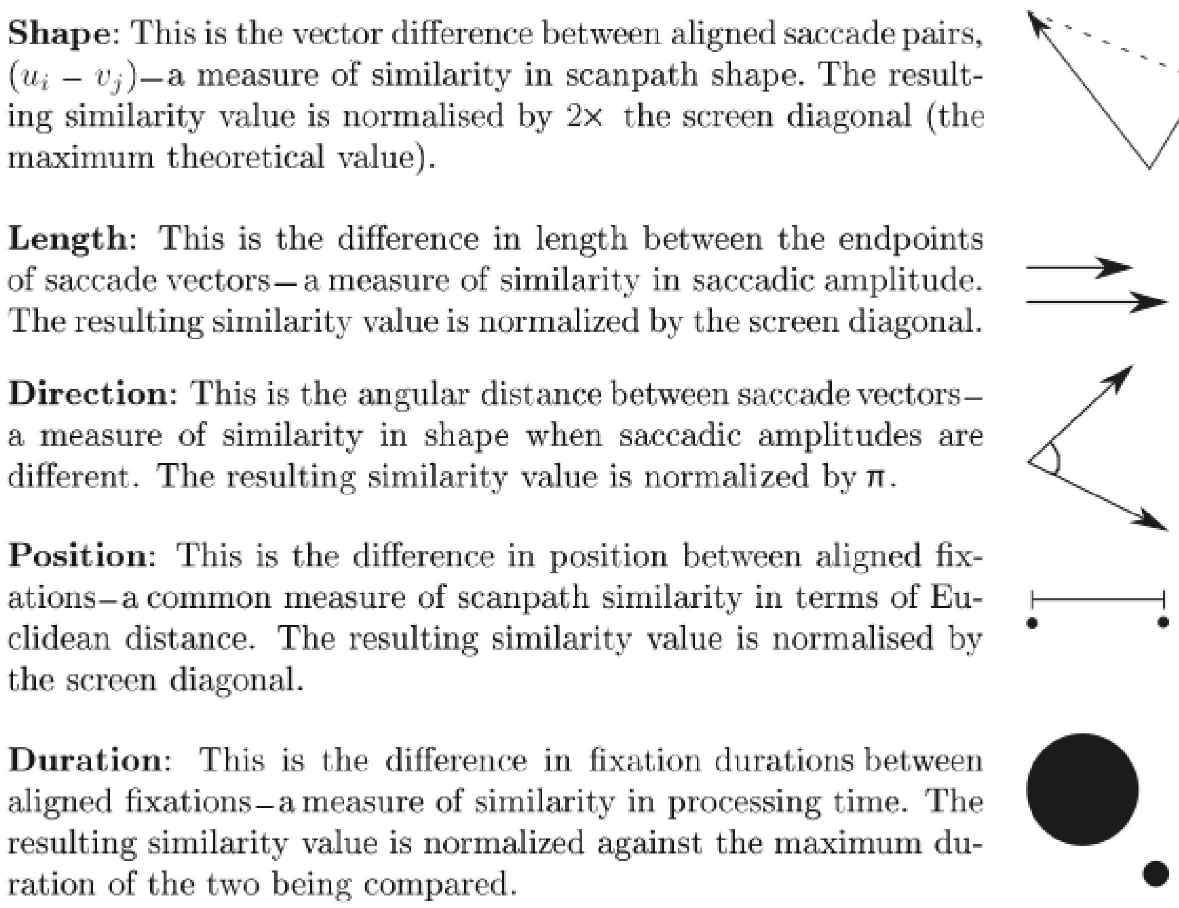


Fig. 1: Example scan path as used in the MultiMatch algorithm

Two such sequences (which can differ in length) are compared on the five dimensions **vector shape**, **vector length** (saccadic amplitude), **vector position**, **vector direction** and **fixation duration** for a multidimensional similarity evaluation.



Shape: This is the vector difference between aligned saccade pairs, $(u_i - v_j)$—a measure of similarity in scanpath shape. The resulting similarity value is normalised by 2× the screen diagonal (the maximum theoretical value).

Length: This is the difference in length between the endpoints of saccade vectors—a measure of similarity in saccadic amplitude. The resulting similarity value is normalized by the screen diagonal.

Direction: This is the angular distance between saccade vectors—a measure of similarity in shape when saccadic amplitudes are different. The resulting similarity value is normalized by π.

Position: This is the difference in position between aligned fixations—a common measure of scanpath similarity in terms of Euclidean distance. The resulting similarity value is normalised by the screen diagonal.

Duration: This is the difference in fixation durations between aligned fixations—a measure of similarity in processing time. The resulting similarity value is normalized against the maximum duration of the two being compared.

Fig. 2: Dimensions of scan path comparison, taken from Dewhurst et al., 2012

1.1 Overview

The method takes two n x 3 fixation vectors (x-coordinate in px, y-coordinate in px, duration in sec) of two scan paths as its input. Example files how input should look like can be found here.

- **Step 1: Representation of scan paths as vector sequences**

  An idealized saccade is represented as the shortest distance between two fixations. The Cartesian coordinates of the fixations are thus the starting and ending points of a saccade. The length of a saccade in x direction is computed as the difference in x coordinates of starting and ending point. The length of a saccade in y direction is computed accordingly. To represent a saccade as a vector in two-dimensional space, the lengths in x and y directions are transformed into polar coordinates (length from coordinate origin (Rho), polar angle in radians (Theta)) by means of trigonometry.

- **Step 2: Scanpath simplification**

  Scanpaths are simplified based on angle and amplitude (length) to reduce their complexity. Two or more saccades are grouped together if angles between two consecutive saccades are below an angular threshold *TAmp*, and intermediate fixations are shorter than a duration threshold *TDur*, or if the amplitude of successive saccades is below a length threshold *TAmp* and the surrounding fixation duration. As such, small, locally contained saccades, and saccades in the same general direction are summed to form larger, less complex saccades (Dewhurst et al., 2012). This process is repeated until no further simplifications are made. Thresholds can be set according to use case. The original simplification algorithm implements an angular threshold of 45° and an amplitude threshold of 10% of the screen diagonal (Jarodzka, Holmqvist & Nyström, 2010).

- **Step 3: Temporal alignment**

  Two simplified scan paths are temporally aligned in order to find pairings of saccade vectors to compare. The aim is not necessarily to align two saccade vectors that constitute the same component in their respective vector sequence, but those two vectors that are the most similar while preserving temporal order. In this way, a stray saccade in one of the two scan paths does not lead to an overall low similarity rating, and it is further possible to compare scan paths of unequal length. To do so, all possible pairings of saccades are evaluated in similarity by their shape (i.e. vector differences). More formally, the vector difference between each element i in scan path S1 = (u1, u2, . . . , um) and each element j in scan path S2 = (v1, v2, . . . , vn) is computed and stored in Matrix M as a weight. Low weights correspond to high similarity. An adjacency matrix of size M is build, defining rules on which connection between matrix elements are allowed: In order to take temporal sequence of saccades into account, connections can only be made to the right, below or below-right. Together, matrices M and the adjacency matrix constitute a matrix representation of a directed, weighted graph. The elements of the matrix are the nodes, the connection rules constitute edges and the weights define the cost associated with each connection.

- **Step 4: Scanpath selection**

  A Dijkstra algorithm (Dijksta, 1959) is used to find the shortest path from the the first two saccade vectors to the last two saccade vectors. "Shortest" path is defined as the connection between nodes with the lowest possible sum of weights.

- **Step 5: Similarity calculation**

  Five measures of scan path similarity are computed on the aligned scan paths. This is done by performing simple vector arithmetic on all aligned saccade pairs, taking the median of the results and normalizing it. As a result, all five measures are in range [0, 1] with higher values indicating higher similarity between scan paths on the given dimension.

For a more detailed overview of the algorithm, take a look at the original publication by Dewhurst et al. (2012) and Jarodzka et al. (2010).

1.1.1 References

Dewhurst, R., Nyström, M., Jarodzka, H., Foulsham, T., Johansson, R., & Holmqvist, K. (2012). It depends on how you look at it: Scanpath comparison in multiple dimensions with MultiMatch, a vector-based approach. Behavior research methods, 44(4), 1079-1100. https://doi.org/10.3758/s13428-012-0212-2

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269 - 271. https://doi.org/10.1007/BF01386390

Jarodzka, H., Holmqvist, K. & Nyström, M. (2010). A vector-based, multidimensional scanpath similarity measure. In ETRA '02: Proceedings of the 2010 symposium on eye tracking research and applications, ACM, New York. https://doi.org/10.1145/1743666.1743718

CHAPTER 2

multimatch-gaze

multimatch-gaze is a Python-based reimplementation of the MultiMatch algorithm (Jarodzka, Holmqvist & Nyström, 2010). The original Matlab toolbox was kindly provided via email by Dr. Richard Dewhurst and the method was ported into Python with the intent of providing an open source alternative to the Matlab toolbox.

The module provides the possibility to compute the similarity of two scan paths with a terminal command or within a Python instance (see section API).

2.1 Getting started

2.1.1 Installation

It is recommended to use a dedicated virtualenv.

```
# create and enter a new virtual environment (optional)
virtualenv --python=python3 ~/env/multimatch
. ~/env/multimatch/bin/activate
```

Via pip install

multimatch-gaze can be installed via pip (**P**ip **I**nstalls **P**ython). To automatically install multimatch-gaze with all dependencies (pandas, numpy, argparse) type:

pip install multimatch-gaze

Via Github

The source code for multimatch-gaze can be found on Github.

A short tutorial. . .

. . . session to get a first hang on how to use multimatch-gaze can be conducted by cloning the Github repository and executing the examples provided in the API section. The data used in these examples corresponds to the data found in the repository.

2.2 Support

All bugs, concerns and enhancement requests for this software can be submitted here. All contributions, be it in the form of an issue or a pull-request, are always welcome. If you wish to test the functionality of multimatch-gaze locally, for example prior to a pull-request, install pytest and run the following command from the root of the repository:

```
python -m pytest
```

```
python -m pytest -s -v
```

2.3 Acknowledgements

We thank Dr. Richard Dewhurst for kindly and swiftly providing the original Matlab code for the MultiMatch toolbox via e-mail and being supportive of an open source implementation.

CHAPTER 3

API

The multimatch-gaze command is the standalone equivalent of the MultiMatch toolbox and is easiest executed directly from the command line.

3.1 Command line

The computation of the similarity between two scan paths doesn't involve anything beyond the command line keyword multimatch-gaze followed by two input files, corresponding to tab-separated files with a fixation vector, and the screensize in pixel, supplied as two consecutive integers corresponding to the x and y dimension of the screen:

```
multimatch-gaze path/to/scanpath_one path/to/scanpath_two x_dim y_dim
```

The input files will be read with numpys recfromcsv() and should contain one fixation per line. The first three columns of the input file are relevant. One column should contain x-coordinates of the fixation in px (start_x), one column should contain y-coordinates in px (start_y), and one column should contain contain the fixation duration (duration) in seconds. Example files with this structure can be found here. Note that the input data needs to have a header with the correct column names (x_start, y_start, duration).

An examplary command line call that you could execute if you cloned the repository looks like this:

```
multimatch-gaze data/fixvectors/segment_0_sub-01.tsv data/fixvectors/segment_0_sub-19.
↪tsv 1280 720
```

Scanpath simplification

Optionally, scan paths can be simplified to reduce their complexity. To simplify scan paths, specify the following arguments:

- --direction-threshold: If two consecutive saccades have a small angle, they will be combined. Should be in degrees, such as 45.0 for 45°
- --amplitude-threshold: If two consecutive saccades are short, they will be combined. Should be in pixel, such as 100.0 for 100px.

- `--duration-threshold`: Only if the intermediate fixation's durations are shorter than this threshold the above simplification will be performed. Should be in seconds, such as `0.1` for 100ms.

Note: If either direction- or amplitude threshold is specified as 0, no grouping will be performed!

A commandline call of the module **with** simplification would hence look like this:

```
multimatch-gaze data/fixvectors/segment_0_sub-01.tsv data/fixvectors/segment_0_sub-19.
↪tsv 1280 720
--direction-threshold 45.0 --amplitude-threshold 100.0 --duration-threshold 0.1
```

There are no guidelines whether and if so, how much, simplification is appropriate, and it is strongly dependent on individual use case. The original Matlab toolbox implements a default amplitude threshold of 10% of the screen diagonal as amplitude, 45° as angle, and 300ms as duration thresholds. `multimatch-gaze` has defaults of 0 for simplification parameters (i.e. simplification is not performed by default).

Output configuration

The way results are displayed in the command line can be configured with the `-o/--output-type` parameter. Three different formats are possible:

- `hr` (**default**): Results are returned row-wise, with dimension name. This is the most human readable format, and good for a quick glance at results:

```
Vector similarity = <value>
Direction similarity = <value>
...
```

- `single-row`: Results are returned in a single row, delimited with tabs, and without dimension name. Makes it easy to collate results in a table:

```
<vectorsim>\t<directionsim>\t<lengthsim>\t<positionsim>\t<durationsim>
```

- `single-del`: Results are returned row-wise, with tabs seperating dimension name and value. This makes it easy to pick out a selection of scores:

```
vector\t<value>
direction\t<value>
length\t<value>
position\t<value>
duration\t<value>
```

REMoDNaV helper

REMoDNaV is a velocity-based event detection algorithm for eye movement classification. It detects and labels saccades, fixations, post-saccadic oscillations, and smooth pursuit movements, and it was specifically developed to work with dynamic stimulation. `REMoDNaV` is an open-source Python package, and its outputs, BIDS-compliant TSV files, can be read natively by `multimatch-gaze`. The conversion of data to a fixation vector is then handled internally.

Should you have data produced by `REMoDNaV`, you can to supply the `--remodnav` parameter:

```
multimatch-gaze data/remodnav_samples/sub-01_task-movie_run-1_events.tsv
data/remodnav_samples/sub-01_task-movie_run-2_events.tsv 1280 720 --remodnav
```

As `REMoDNaV` classifies pursuits, which can be seen as a "visual intake" category such as fixations, you can decide whether to **include** or **discard any pursuit events**. Using pursuits would be useful for example in the case of moving stimuli: Visual intake of a moving target would appear as a pursuit in eye tracking data. Setting this function is handled with the `--pursuit` parameter. Chose between options `"discard"` and `"keep"`.

- `discard` (**default**) will disregard pursuit events.
- `keep` will turn a pursuit movement into two fixations - the start and ending point of the pursuit movement.

Specify to keep pursuit movements (i.e. inclusion into the scan path) like this:

```
multimatch-gaze data/remodnav_samples/sub-01_task-movie_run-1_events.tsv
data/remodnav_samples/sub-01_task-movie_run-2_events.tsv 1280 720 --remodnav --
↪pursuit 'keep'
```

3.2 Python

If you wish to use the functionality of multimatch-gaze within a running Python instance such as IPython, you can import the module and use the function `docomparison`. Here is an example:

```
import multimatch_gaze as m
import numpy as np

# read in data
fix_vector1 = np.recfromcsv('data/fixvectors/segment_0_sub-01.tsv',
delimiter='\t', dtype={'names': ('start_x', 'start_y', 'duration'),
'formats': ('f8', 'f8', 'f8')})
fix_vector2 = np.recfromcsv('data/fixvectors/segment_0_sub-19.tsv',
delimiter='\t', dtype={'names': ('start_x', 'start_y', 'duration'),
'formats': ('f8', 'f8', 'f8')})

# Optional - if the input data are produced by REMoDNaV
# pursuits = True is the equivalent of --pursuits 'keep', else specify False
fix_vector1 = m.remodnav_reader('data/remodnav_samples/sub-01_task-movie_run-1_events.
↪tsv',
screensize = [1280, 720], pursuits = True)

# execution with multimatch-gaze's docomparison() function without grouping
m.docomparison(fix_vector1, fix_vector2, screensize=[1280, 720])

# execution with multimatch-gaze's docomparison() function with grouping
m.docomparison(fix_vector1, fix_vector2, screensize=[1280, 720], grouping=True,␣
↪TDir=30.0,
TDur=0.1, TAmp=100.1)
```

The results will be returned as an array, such as `[0.98, 0.87, 0.45, 0.78, 0.80]`.

CHAPTER 4

An example computation

The following section shows a multimatch-gaze use case to compute the scan path similarities of participants that watched the Hollywood movie Forrest Gump during simultaneous fMRI acquisition.

4.1 Data and sample

Data for all analyses stems from the 2016 released extension of the studyforrest dataset (Hanke et al., 2016; Sengupta et al., 2016). In this extension, N = 15 right-handed participants (age range 21 - 39 years, mean age 29.4 years, six female, normal or corrected-to-normal vision), who had previously participated in the studyforrest project, watched the audio-visual movie 'Forrest Gump' (R. Zemeckis, Paramount Pictures, 1994) during simultaneous fMRI and eye-tracking recording. The video track for the movie stimulus was re-encoded from Blu-ray into H.264 video (1280 x 720px at 25 frames per second (fps)). In accordance to the procedure in an earlier phase of the studyforrest project, the movie was shortened by removing a few scenes less relevant for the major plot to keep the fMRI recording session under two hours. The shortened movie was then split into eight segments of roughly 15 minutes of length (for an overview on segment duration, final stimulus content and detailed procedures see Hanke et al. (2014)). Visual stimuli were projected on to a screen inside the bore of the magnet using an LCD projector, and presented to the subjects through a front-reflective mirror on top of the head coil at a viewing distance of 63cm. The screen dimensions were 26.5cm x 21.2cm (corresponding to 1280 x 1024px) at a resolution of 720p at full width, with a 60Hz video refresh rate (Sengupta et al., 2016). Eye-tracking was performed with an Eyelink 1000 (software version 4.594) using monocular corneal reflection and pupil tracking with a temporal resolution of eye gaze recordings of 1000Hz. The camera was mounted at an approximate distance of 100cm to the left eye of subjects, which was illuminated by an infrared light source (Hanke et al., 2016). Eye-tracking data were normalized such that all gaze coordinates are in native movie frame pixels, with the top-left corner of the movie frame located at (0, 0) and the lower-right corner located at (1280, 546) (ibid.). The amount of unusable data, primarily due to signal loss during eye blinks, ranged from less than 1 to 15% for 13 of the 15 in-scanner subjects (the other two subjects' data contained 85 and 36% of data loss, respectively). In-scanner acquisition had an approximate spatial uncertainty of 40px according to the calibration procedure (ibid.).

4.2 Event detection and scan path derivation

Raw gaze data was classified into different categories of eye movements with an adaptive, data-driven algorithm for robust eye movement detection for natural viewing (REMoDNaV) in Python. The algorithm categorizes the raw data into saccades, fixations, smooth pursuits, and post-saccadic oscillations (glissades), and disregards any unclassifiable data (such as blinks). It was specifically developed to compute robust results even under high noise conditions. For an overview of the algorithmic details and evaluation of REMoDNaV compared to contemporary algorithms and human annotations, please see the respective publication (Dar et al., in preparation) or take a look at the REMoDNaV module. The eye events are reported together with their start and end coordinates, their onsets and durations in seconds, their velocity, and the average pupil size. Fixation vectors as input for multimatch-gaze were derived from the REMoDNaV output. As the stimulus was dynamic with moving targets that evoke smooth pursuit movements, such pursuit events are categorized to be an eye movement category of 'visual intake', just as fixation. Therefore, in a first step, the start and end of pursuit movements were included in scan paths to compare as well. In a second step, the continuous eye movement data (~15 min per run) was split into shots corresponding to segments that did not contain scene changes between depicted locales using the published location annotation for the movie (Häusler & Hanke, 2016). This was done to accommodate the fact that subjects gazes have a bias towards the center in Hollywood movies (Tseng et al. 2009). This bias can at least in part be traced back to a strong center bias directly after cuts in dynamic scenes. Lastly, within each segment, scan paths of the median shot length of ~4.92 seconds. To further evade any problems associated with the center bias, scan paths were extracted from the end of the segment: The last oculomotor event within the range of the segment marked the end of a scan path. As such, scan paths began maximally distant to the snippet onset.

4.3 multimatch-gaze application

Overall scan path similarities were computed in a two-step procedure. First, scan path comparisons of all scan paths from the same shot of two subjects were calculated for all possible pairs of subject. This resulted in 105 combinations for N = 15 subjects. These comparisons were done without any further simplification (i.e. no use of the direction, length, and duration thresholds), as even minor differences in scan paths obtained from a movie can correspond to major differences in attended visual stimuli. In a second step, the resulting similarities for each of the five similarity dimensions were averaged. Thus, for each snippet longer than 4.92s five similarity measures were computed that represented the average similarity of scan paths of all subjects on the given dimension. The results of this computation can be found on Github.

4.4 Results

In total, 533 scan paths were extracted from the movie. The median duration of extracted scan path duration was 4.39 seconds (mean = 4.36s). The following figures give an overview of the similarity computations. Figures 1 and 2 display a frame within the segments in the first run of the movie with the lowest and highest group-level similarity (averaged across the five dimensions). The overlayed eye gaze was created with a custom script that is part of the studyforrest phase-2 data release (Hanke et al., 2016) and publicly available in the corresponding Github repository.

The overall similarity of gaze was high, however, there were consistent differences between dimensions. The Shape, Length and Position dimension displayed very high similarities, and the average Duration similarity was the lowest of all dimensions. Medians and means correspond closely, and standard deviations are very small. This is also highlighted by Figure 3.

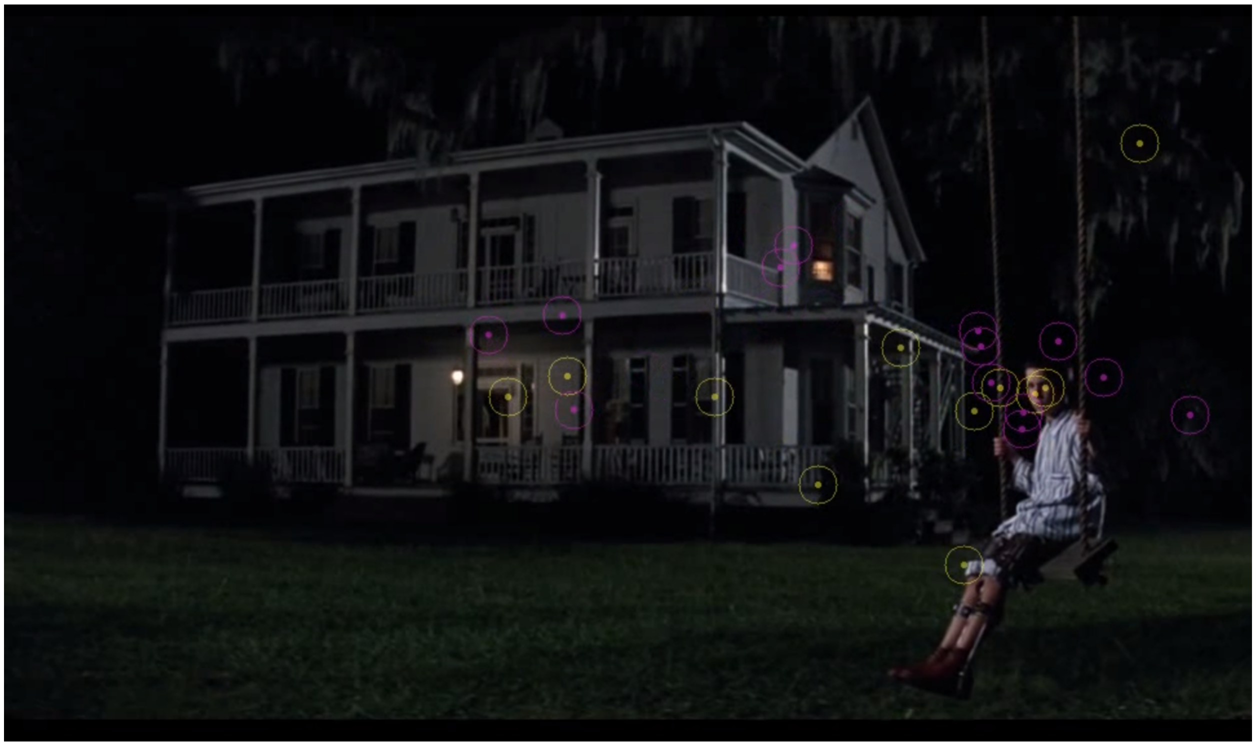


Fig. 1: One frame from the segment within the first run of the movie with the **lowest** average group-level similarity. The circles represent participants center of eye gaze.

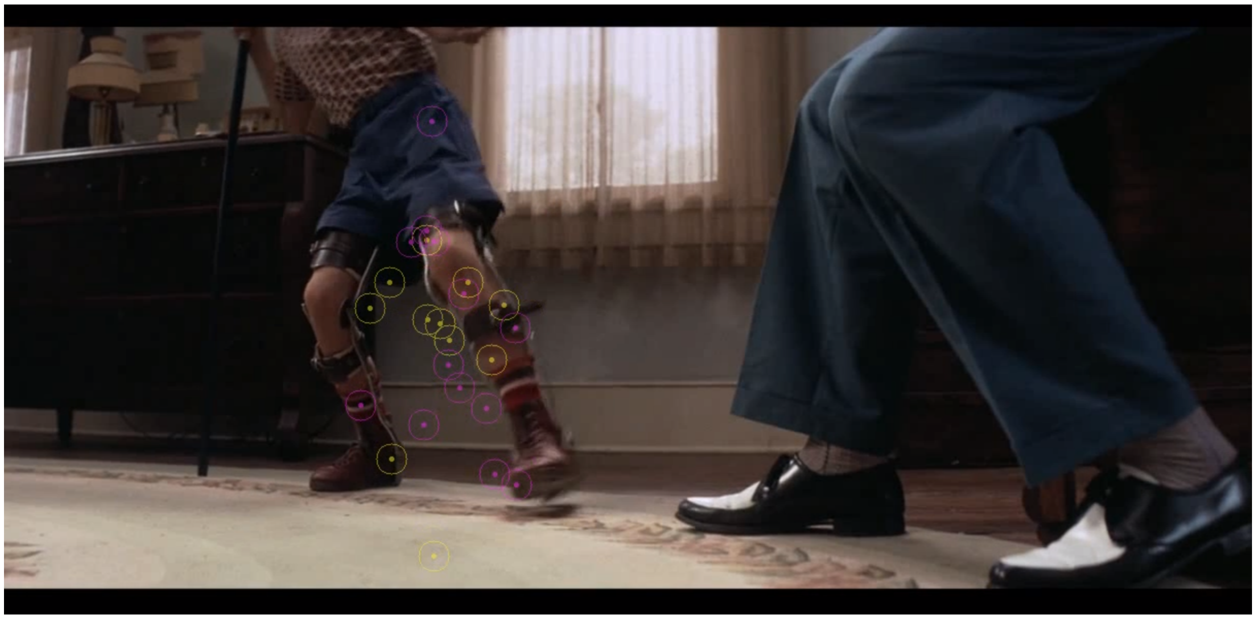


Fig. 2: One frame from the segment within the first run of the movie with the **highest** average group-level similarity. The circles represent participants center of eye gaze.

Variable	mean [SD]	median
Shape	0.97 [0.01]	0.97
Position	0.88 [0.03]	0.89
Length	0.96 [0.01]	0.96
Duration	0.54 [0.05]	0.55
Direction	0.72 [0.05]	0.71

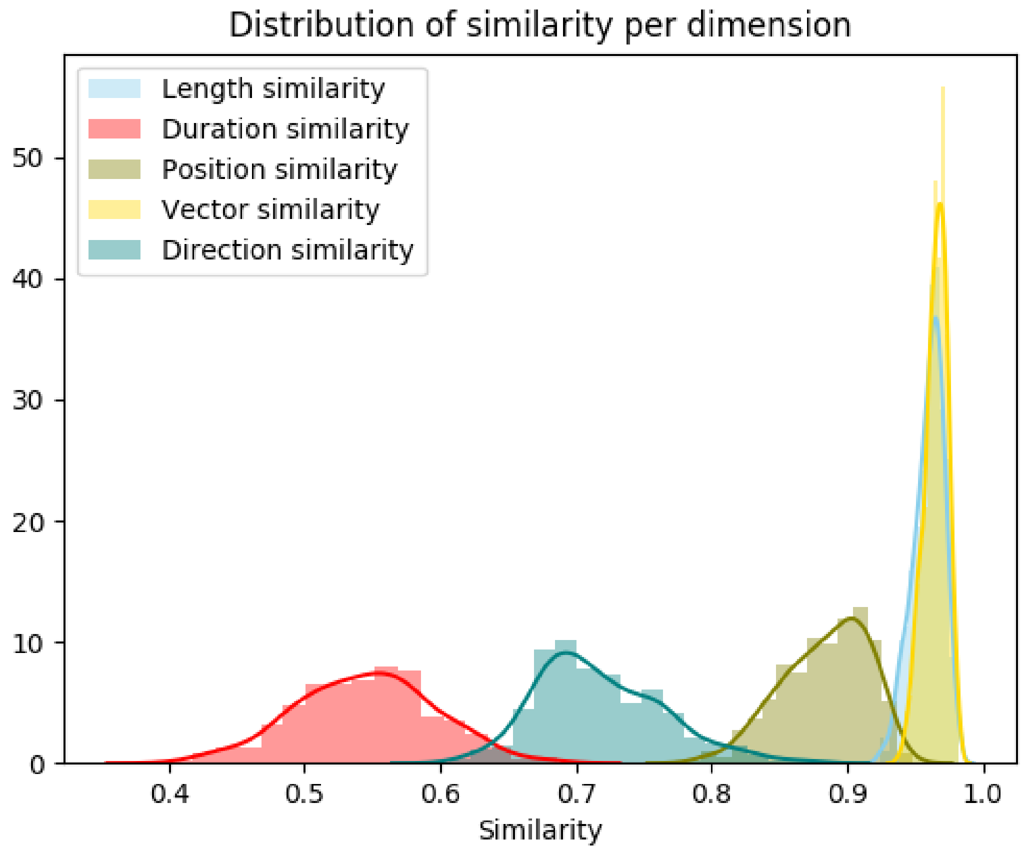


Fig. 3: Distribution of similarity measures throughout the movie. Note the extremely high position and length dimension.

4.5 Discussion

As evident from the previous table and figure, scan paths were almost perfectly similar on the dimensions vector length and vector position. This is likely at least partially due to the scan path alignment based on the scan path shape. Scanpaths were also highly similar on the position dimension, which demonstrates a strong gaze control of the movie stimulus. Subjects scan paths differed more substantially on the dimensions direction and duration, which indicates differences in fixation dwelling times and saccadic angle. Thus, the general points of interest (as evident from high similarities in position, length and shape) were similar across subject, but differences in direction and duration might indicate interindividually different exploration strategies. All dimensions show a remarkable consistency in similarity measures as evident from the small standard deviations. This might indicate a consistently high level of exogenous attentional control by the movie stimulus. This finding is consistent with research on viewing behavior during movies:

Unlike during static image viewing, the spatio-temporal gaze behavior of multiple viewers exhibits a substantial degree of coordination in movie watching. Smith and Henderson (2008) cued the term *attentional synchrony* for this phenomenon. During attentional synchrony, viewers gazes cluster around a small portion of the screen at any one moment. Goldstein et al. (2007), for example, found the distribution of fixations of viewers to occupy less than 12% of the total screen area in more than 50% of the time in six Hollywood movies. In a comparison between different types of static and dynamic visual stimuli, Dorr et al. (2010) found the highest consistency between viewers eyegazes during professionally produced (Hollywood) movies, likely largely due to the use of cinematic composition of scenes, deliberate camera work and editing. Hasson et al. (2008) found high correspondence in gaze behavior across subjects, even for backwards presentations of movies.

The results obtained with the multimatch algorithm from the Hollywood movie Forrest Gump, therefore, are consistent with known properties of gaze behavior during movie watching. This analysis has furthermore demonstrated one way of using multimatchs scan path comparison on a grouplevel similarity computation per segment. If you have any questions about this example, please ask here.

4.6 References

Dorr, M., Martinetz, T., Gegenfurtner, K. R., & Barth, E. (2010). Variability of eye movements when viewing dynamic natural scenes. Journal of vision , 10 (10), 28. https://dx.doi.org/10.1167/10.10.28

Goldstein, R. B., Woods, R. L., & Peli, E. (2007). Where people look when watching movies: Do all viewers look at the same place? 37 (7), Computers in biology and medicine ,957 - 964. https://doi.org/10.1016/j.compbiomed.2006.08.018

Hanke, M., Baumgartner, F. J., Ibe, P., Kaule, F. R., Pollmann, S., Speck, O., … Stadler, J. (2014). A high-resolution 7-tesla fmri dataset from complex natural stimulation with an audio movie. Scientific data , 1 ,140003. https://doi.org/10.1038/sdata.2014.3

Hanke, M., Adelhöfer, N., Kottke, D., Iacovella, V., Sengupta, A., Kaule, F. R., … Stadler, J. (2016). A studyforrest extension, simultaneous fmri and eye gaze recordings during prolonged natural stimulation. Scientific data , 3 ,160092. https://doi.org/10.1038/sdata.2016.92

Hasson, U., Landesman, O., Knappmeyer, B., Vallines, I., Rubin, N., & Heeger, D. J. (2008). Neurocinematics: The neuroscience of film. Projections , 2 (1), 1-26. https://doi.org/10.3167/proj.2008.020102

Häusler, C. O., & Hanke, M. (2016). An annotation of cuts, depicted locations, and temporal progression in the motion picture" forrest gump". F1000Research , 5. https://doi.org/10.12688/f1000research.9536.1

Sengupta, A., Kaule, F. R., Guntupalli, J. S., Homann, M. B., Häusler, C., Stadler, J., & Hanke, M. (2016). A studyforrest extension, retinotopic mapping and lo- calization of higher visual areas. Scientific data , 3 , 160093. https://doi.org/10.1038/sdata.2016.93

Smith, T. J. (2013). Watching you watch movies: Using eye tracking to inform film theory. http://dx.doi.org/10.1093/acprof:oso/9780199862139.003.0009

Smith, T., & Henderson, J. (2008). Attentional synchrony in static and dynamic scenes. Journal of Vision , 8 (6), 773-773. https://doi.org/10.1167/8.6.773