# An Interactive Web-Based Visualization Tool for Analyzing Eye Tracking Data

Petru Radulescu, Sam Nijsten, Lucas Heutinck, Khang Ly, Cees Maessen, and Michael Burch

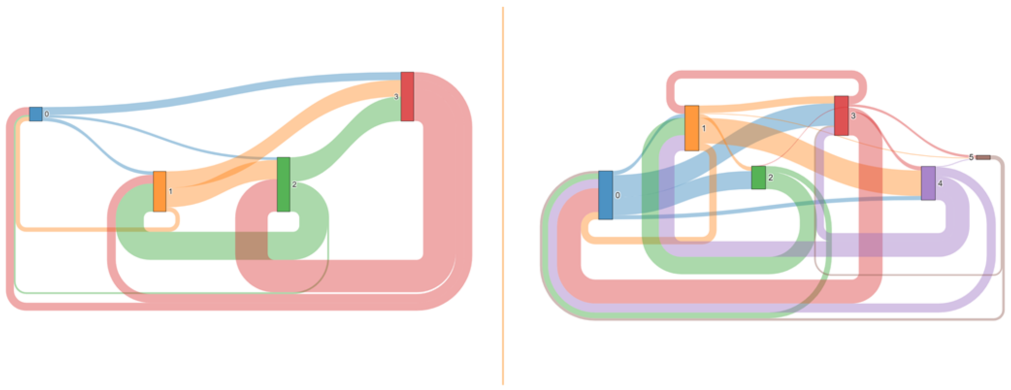


Fig. 1: An example of a Sankey diagram generated using the InfoVis tool for the public transport map *02_Berlin_S1.jpg*, showing transitions between areas of interest: (left) four clusters with the default node positioning, (right) six clusters with user-positioned nodes. Normally employed in the illustration of energy transfers, Sankey diagrams were found to be equally effective in the representation of AOI transitions.

**Abstract**—This paper describes an interactive and web-based visualization tool used to highlight and explore visual scanning patterns of users attempting to solve tasks involving a static stimulus. In order to illuminate the effectiveness of such a tool, real-world eye tracking data (concerning a repertoire of metro maps as visual stimuli) from a previously conducted experiment was explored. Viewing data in this way allows for comprehensive identification of common scanning patterns within the sample of users, which is the intended goal of the tool. Visualizations are enhanced with the inclusion of numerous interaction techniques and alternate views. The website itself prioritizes the user experience, with a variety of quality-of-life features; being web-based comes with its own set of advantages, notably allowing users to upload data sets, conduct explorations, and share results to relevant parties as needed.

**Index Terms**—Visual scanning, Data analytics, Interactive visualization, Eye tracking, Public transport maps.

## 1 Introduction

The method of eye tracking has been extensively used in the fields of human-computer interaction and cognitive science. Researchers have used this technique in order to achieve a better understanding of the cognitive processes underlying the processing of information. Eye tracking is based on the eye-mind assumption formulated by Just and Carpenter [20]. According to the assumption, eye movements indicate where attention is being directed towards. It is widely assumed that this is indeed the case during conscious processing of information [22].

The method of eye tracking follows a basic principle - visual exploration of a static or dynamic stimulus guided by one or several tasks. The primary data of interest is the scanpath for each user - an $n$-length sequence of 2D points $p_1 \to \ldots \to p_n$, where each $p_i$ with $1 \leq i \leq n$ carries satellite data in the form of the fixation duration $d_i$ (measured in milliseconds). However, such raw data is difficult to effectively analyze without the use of information visualization (InfoVis) systems.

InfoVis [1, 2, 7, 8, 12–15, 27] "produces interactive visual representations of abstract data to reinforce human cognition, thus enabling the user to gain knowledge about the internal structure of the data and causal relationships in it" [29]. Interaction is the crux of InfoVis, as an InfoVis system lacking interactivity is little more than a static image which struggles to maintain relevance as the dataset grows large. To support the investigation of eye tracking data, it is important to consider and implement the seven different categories of interaction defined by Yi et al.: Select, explore, reconfigure, encode, abstract, filter, and connect in the creation of the InfoVis system.

• *Eindhoven University of Technology*

This paper discusses an interactive and web-based InfoVis tool allowing users to upload their own eye tracking datasets (with some format limitations) and conduct explorations and analyses regarding the characteristics of aggregated scanpaths. These range from common scanning behaviors to "hot spots", which vary depending on the visual stimulus at hand. Being web-based also allows users to easily share their findings with relevant parties. The end goal of such a tool is to facilitate the data analysis process and gain useful insights in conducting future eye tracking studies. The implemented visualizations include both "traditional" plots (e.g. heatmaps and gaze plots with the relevant stimulus underlaid) and those prioritizing the relationships between clusters of data points (e.g. transition graphs and Sankey diagrams); they are also interactively linked, further simplifying the task of making data-driven discoveries regarding studies and their corresponding visual stimuli.

Regarding the development process of the InfoVis tool, administration and goal-setting were facilitated with the Scrum methodology. Formally, Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value [26]. Project development was divided into four sprints, with each sprint entailing a series of tasks (described in vivid detail) assigned to individual team members. Three hour-long meetings were held per week (led by the resident Scrum Master) to ensure active feedback and progress, which were substantiated by a sprint retrospective at the conclusion of each sprint. This process allowed continued reflection upon team performance, cohesion and the status of the tool.

As an example of the real-world applicability of this InfoVis tool, data from a previously conducted eye tracking study regarding participants' behavior in tracking a path through a metro map was employed. The paper also discusses other relevant details, as well as possible shortcomings and areas of improvement about the tool's implementation.

## 2 Related Work

Eye tracking has become an increasingly popular field of study due to the wide range of real-world applications stemming from analysis of fixation and scanning behaviors. Recent developments have led to increased accuracy and decreased costs of running such tests. Hence, visualization of eye tracking data has become an interdisciplinary practice [4]. To use an example from neuroscience, eye movement data can substantiate the evaluation and diagnosis of oculomotor dysfunction, which is a marker for numerous serious conditions, including schizophrenia; common observational tests yield subjective results [5]. An example from the field of data visualization would be the work by Burch et. al in using eye tracking data to identify exploration behavior in tree structures [10]. Broadly, the interpretation of fixation data depends on the context and purpose of the study at hand; for the most part, they indicate relevant points or areas of interest (POIs/AOIs) within the stimulus.

Jarodzka et al. developed a vector-based measure to compare scanpath similarity between users for ETRA 2010. Such a method provides a simplified view of scanpaths that preserves the distinction between fixations and saccades which can then be analyzed through vector geometry [19]. Making connections to the original stimulus and comparisons between users are hampered by the lack of an aggregated scanpath view and the paths not being overlaid onto the original stimulus, limiting the application in exploratory data analysis.

For ETRA 2012, Duchowski et al. presented their methods of gaze visualization with real-time heatmaps using a GPU implementation. Heatmaps naturally provide a much cleaner description of aggregate gaze; the order of fixations is obscured in exchange for more coherent depictions of AOIs across users [18]. However, their work is restricted to single visualizations. Viewing the same dataset from from multiple perspectives and angles can facilitate the identification of desired user behavior and drawing of conclusions. This is especially feasible for heatmaps, which are easy to implement and interpret, and fast to execute.

Burch et al. described an AOI rivers model, intended to demonstrate the evolving visual attention behavior of participants for a chosen stimulus [11]. While such a tool is able to quickly identify transitions between AOIs, perception can become challenging as there is no inherent hierarchical structure in the eye movements as illustrated by the model [9]. The gaze stripes model presented by Kurzhals et al. provides good scalability with respect to the number of participants and the time length of gaze sequences [21]. Nonetheless, it proves unsuitable for the task of visualizing general scanning behaviors and consistencies in scanpath data on a per-user basis.

The report on the state-of-the-art of eye tracking visualization by Blascheck et al. indicated that most visualization models concerning static stimuli lacked interactivity [3]. Additionally, most of the tools and methods discussed do not possess a web-based implementation. Hence, the tool described in this paper aims to substantiate this niche through the implementation of interaction techniques as defined by Yi et al. [29]; all visualizations and alternate views are linked, with various aesthetic settings available to the user.

## 3 Data Model and Processing

This section concerns the overall structure of the raw data input and the functions involved in filtering the data for visualization.

### 3.1 Data Model

Input data for this visualization tool comes in the form of a .csv file. The data consists of seven columns, notably the $(x, y)$ coordinates of each fixation along with the duration and associated user. The data is pre-sorted by the recorded timestamps and fixation indices; either of these columns could act as a primary key for the dataset. This raw data does not conceal any complex structures, as the main quantity of interest is simply the fixation points, mathematically modeled as ordered pairs of $(\textit{MappedFixationPointX}, \textit{MappedFixationPointY})$. As such, the data does not need extensive parsing before use in the Jupyter notebook (only conversion into a DataFrame is required).

### 3.2 Data Processing

After conversion into a DataFrame, further processing is required to extract meaningful subsets of rows which can then be visualized accordingly. This was achieved through the use of several key functions, which are structured in such a way as to facilitate interactive widget callbacks. Figure 2 contains the full node-link diagram of the tool. The functions perform the following tasks:

- Iterate through the main DataFrame and return a list of users who were provided the stimulus specified in the argument.
- Create a subset of the main DataFrame with a list of users (from the previous function) and stimulus as arguments. Returns a ColumnDataSource (a fundamental data structure which drives almost all Bokeh glyphs/plots, can be easily converted back into a DataFrame for other libraries).
- Various plotting functions which take a ColumnDataSource as an argument and output the relevant plot (with the necessary styling and formatting).
- A callback function which executes whenever a displayed widget has its value(s) changed. Updates the subset provided as an argument to the plotting functions, clears the output, and re-displays the updated plots.

The function implementations are relatively compact, consisting primarily of DataFrame manipulations and calculations, un-nested loops, branching statements and figure generation (where applicable). As such, the running time remains mostly constant regardless of the scope of user-defined parameters (e.g. list of users, no. of clusters, etc.). Generating an individual visualization of any type takes approximately five seconds. This also means the code is not memory-intensive in the slightest, and ensures the program maintains a straightforward workflow.

#### 3.2.1 Data Transformations

Additional steps are taken to restrict the domain/range and aggregate data depending on the visualization generated. For example, the heatmap functions standardize the fixation coordinates through the mapping $(x, y) \to (\frac{x-\mu_x}{\sigma_x}, \frac{y-\mu_y}{\sigma_y})$. This improves the comprehension of visualizations and ensures the input dataset can easily be compared to other datasets, even if their original domains/ranges differ drastically.

Fixation points were aggregated into clusters for the definition of AOIs through the $k$-means algorithm. The algorithm is suitable because it does not result in overlapping clusters, resulting in uniquely defined AOIs. The number of clusters is a user-defined parameter; because clustering is an unsupervised technique, there is no predisposition regarding the optimal number of clusters [23]. To provide some basic guidelines, the elbow method was used: a basic line plot with the inertia (within-cluster sum-of-squares) of the model on the $y$-axis for each $k$ number of clusters, with ideal $k$ being where the curve begins to flatten out [17]. Testing on several stimuli revealed that three to four clusters is suitable given the dataset.

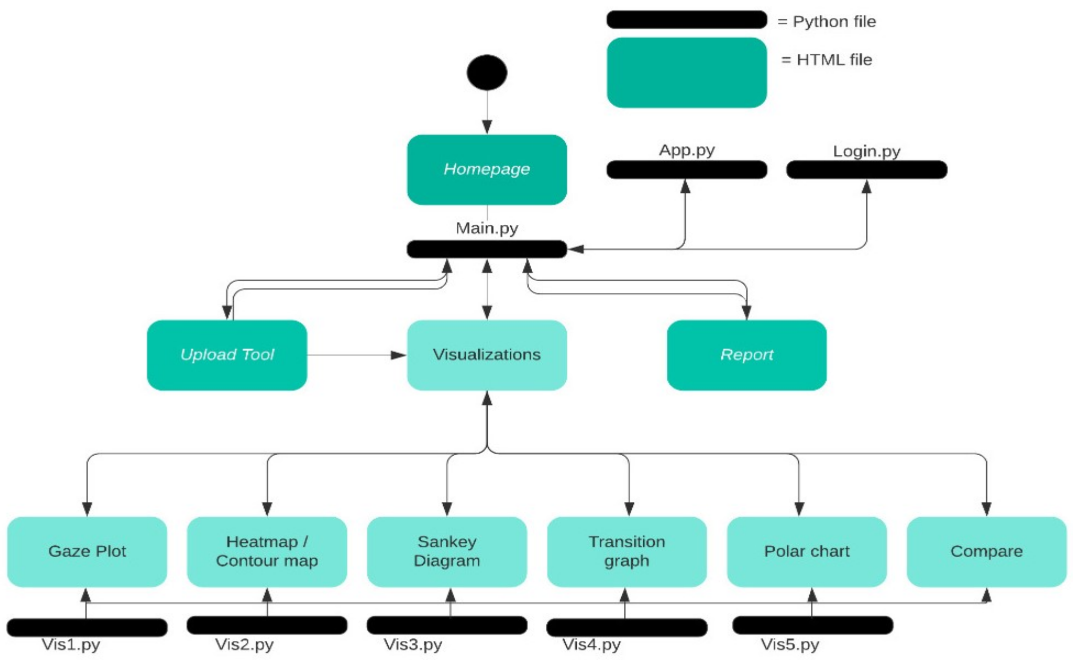


Fig. 2: Node-link diagram of the InfoVis tool. The black circle indicates the starting point.

## 4 Visualization Tool

This section concerns the individual visualizations that constitute the InfoVis tool. The overarching design approach and relevant implementation details are also discussed.

### 4.1 Design Criteria

Several design criteria were formulated for the visualization tool. First of all, unsurprisingly, the tool has to be able to process eye tracking data from a visual stimulus and create at least four different kinds of data visualizations. The tool is usable with static stimuli only. Secondly, a user will be able to select and interact with the different visualizations through a GUI. These include being able to select a single visualization or being able to compare two different visualizations next to each other. In addition the gaze plot overlays the selected stimuli, making it easier for a user to compare between visualizations. Thirdly, the tool runs on a local server and is able to visualize the *MetroMapsEyeTracking* files provided by the Technical University of Eindhoven. These files contain: an Excel file with $(x, y)$-coordinates characterizing gaze points, fixation duration for each participant in milliseconds and a file folder containing images of the provided stimuli. Lastly, for automatic generation of AOIs the $k$-means clustering algorithm was implemented. Users can select the desired amount of AOIs through the GUI. Further elaboration on specific design criteria is given later in the report.

### 4.2 Implementation Details

The programming language decided upon to build the tool was Python, with Anaconda and Jupyter notebooks as the main IDE. The main library with which the visualizations and interactions were created was Bokeh 2.0.2. This library allowed the team to create visualizations, host local servers and incorporate GUI components. Plotly 4.7.1 was used for user interactions. In order to analyze and manipulate data Pandas 1.0.3 was used. Other miscellaneous libraries were applied to a lesser extent where necessary.

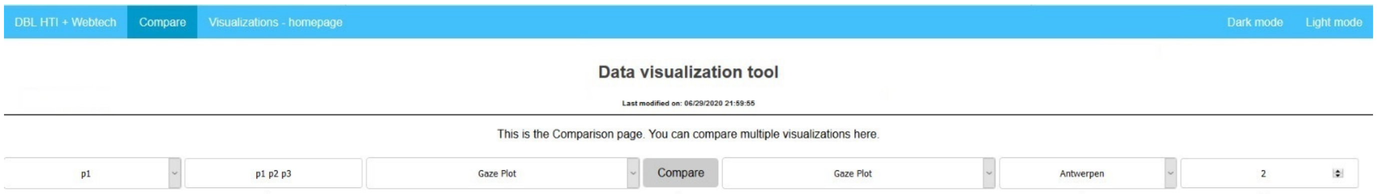
In order to display the visualizations in a clear and cohesive way, the team decided to put them on a web page. To make sure the web pages interact correctly with the databases and can be accessible by multiple users at once, the team opted for the use of the Flask library, which also offers compatibility with the aforementioned libraries. The user login details are transferred to the server by using session data. When a user logs in for the first time, both an upload folder and an HTML folder are created. Here all the uploaded databases and HTML files get saved, ensuring users can login at a different time resuming their work where they left off. The password system is implemented in a similar manner.

The visualizations appear on the website with the use of either an iframe or a passable script, depending on the library used to create the visualization. If it was created with the use of Plotly, the .py file saves the image in an HTML file to the path selected by the user (as described above). The iframe then searches for this folder and displays the HTML file containing the visualization inside of the web page. Visualizations made with anything except Plotly can directly be embedded onto the website by passing on both a Javascript and CSS script. The compare page does however also use an iframe for these visualizations. This is to ensure that they line up perfectly and do not leave any gaps.

### 4.3 Graphical User Interface (GUI)

The graphical user interface or GUI is the main way in which a viewer of the website can modify the information displayed or the general look of the visualization. There are many options that can be implemented, like sliders, dropdown menus, text fields, or check box options (Figure 3). Some of the more rudimentary interactivity components (e.g. zoom, pan) are built-in to their respective visualization functions. Below is a list of supported interactivity on a per-visualization basis:

- Gaze plot: user selection – legend toggle, stimuli selection – dropdown menu, participant selection - dropdown menu.
- Heatmaps: One user or multiple users option - dropdown menu, user selection - text box, stimuli selection – dropdown menu, type of heat map selection - dropdown menu.
- Transition graph: user selection - dropdown menu, stimuli selection - dropdown menu, divisor selection - slider, and show all users – dropdown menu.
- Sankey diagram: stimuli selection - dropdown menu, and the number of clusters - text box.
- Polar chart: Stimuli selection - dropdown menu, number of clusters - text box, spinning the plot - clicking and dragging mouse.



Data visualization tool

Last modified on: 06/29/2020 21:59:55

This is the Comparison page. You can compare multiple visualizations here.

p1 | p1 p2 p3 | Gaze Plot | Gaze Plot | Antwerpen | 2

Fig. 3: Annotated version of the *Compare* tab showing instructions for the tool. See subsection 4.3 for more details on visualization-specific interaction.

### 4.4 Gaze Plot

The first visualization created was a gaze plot (Figure 4), which is the most basic form of eye tracking data visualization. Formally, a gaze plot can be described as an undirected graph, defined by the 2-tuple $G = (V, E)$, where the set of vertices $V$ contains the fixations (with a visual attribute - dot size - storing duration), and the set of unweighted edges $E \subseteq V \times V$ contains the saccades [28].

The gaze plot shows where and for how long a given participant looks at parts of the stimulus. Fixation points are visualized as dots whose diameter gives information about fixation duration. The larger the diameter, the longer the participant's gaze. Lines are drawn between fixation points to indicate saccades. Users are able to click and select which set of participants' gaze plots they wish to see by selecting them in the legend. Additionally, the $x$ and $y$ position of the fixations can be seen by hovering over them with the cursor, as well as the exact duration.

### 4.5 Heatmaps

The heatmaps (Figure 4) are used to visualize how frequently users fixate on certain areas of the stimulus. The first implementation illustrates the fixation points through hexagonal bins. The count of each bin can be visualized directly by hovering over the bin or indirectly using the color scale. This makes it clear which areas were most watched by users. This plot was generated through Bokeh; bins were defined, and the built-in hex tile function was used in combination with a linear color mapping on a chosen scale. Displaying the counts on hover was possible with a Bokeh HoverTool.

On the contour plot implementation the same points are visible. In this visualization the density of gaze points is computed and divided in areas of different colors, this way it can be seen where the density of gaze points is the highest. This plot was generated through Plotly; a base figure was created, to which traces of the contours and scatter coordinates were added. The contour was then filled with a heatmap scheme using a chosen color scale.

Both implementations work based on density over a certain area and both work with standardized data to increase the clarity of the visualization. Heatmaps are usually implemented onto eye tracking data to get an overall view of points of interest and points of relative insignificance; in this tool, heatmaps are applied for the same reason, as well as to analyze the main routes users took to get from point $A$ to point $B$.

### 4.6 Illustrating AOIs and Transitions

The following subsection describes the aspects of the visualization tool that aggregate fixation coordinates into clusters and allow users to analyze AOI definitions and transitions per stimulus.

#### 4.6.1 Transition Graph

The transition graph (Figure 4) is meant to give insights into the way users traverse the stimuli. An AOI will be represented by a blue circle with a radius equal to the fixation duration normalized to be $\in [0,1]$. This assures that the AOIs with a larger fixation duration do not overpower those with a smaller one, whilst still retaining the apparent differences between them. A transition graph is formally represented by a 5-tuple $< Q, \Sigma, \delta, q_0, F >$, where [16]:

i. $Q$ stands for the finite set of states. In our case $Q$ will be the set of AOIs.

ii. $\Sigma$ stands for the finite set of symbols, also called the alphabet. It contains in essence the labels associated with the inputs.

iii. $\delta$ stands for the transition function, $\delta$: $Q$ x $\Sigma \rightarrow Q$. This comes down to the transition between labels of the AOIs for our case.

iv. $q_0$ stands for the initial state. In our case this is the first AOI.

v. $F$ stands for the subset of $Q$ of accepted states. For our purposes this is not relevant.

The interaction is dealt with using Plotly. The user can choose between each of the users for each of the stimuli. They also have the choice to group all users of a certain stimulus together or display them separately. The bin size can also be chosen (bin size is $\in S$ = [5, 10, 15, 20]). By use of the $math.isclose()$ function, AOIs get aggregated, meaning that the values associated with the amount of times it is visited and the fixation duration get added while the unused rows get dropped from the database.

#### 4.6.2 Sankey Diagram

The Sankey diagram is another useful tool in visualizing the general trend in which users transition from one AOI to another given a certain stimulus. Frequently employed in the evaluation of energy transfers, a Sankey diagram is a type of node-link visualization illustrating flow rates between sources. For the purposes of eye tracking data, nodes indicate AOIs and links represent possible transitions. Hence, the number of links is at most equal to the size of the Cartesian product $A \times A$, where $A$ denotes the set of AOIs; the width of each link indicates the number of times a user made that transition.

The $k$-means clustering algorithm offered by Scikit-Learn was used to define AOIs. The number of clusters defined is a user-adjustable parameter (defaulting to four, which is the recommended value given the applied dataset), however larger values may lead to the definition of unevenly distributed AOIs. Transitions were counted as follows: for each coordinate, if their assigned cluster differed from that of the previous coordinate (hence indicating a transition), the corresponding source-target pair had its count incremented. The diagram itself is generated using the Sankey function within the Plotly library, with adjustments to the styling, such as color coding links according to their source node to better differentiate them.

#### 4.6.3 Polar Plot

The final visualization giving information on the different AOIs is the polar plot. In the plot the total fixation duration of all participant at an individual cluster is shown. This visualization functions exactly like a bar chart, but looks more interesting and has more interactive elements. A user can zoom in and out by clicking and dragging their mouse within the plot. Furthermore, a user can click and drag outside the plot to make it spin around. It was made using the polar bar chart function of the Plotly library. The radius shows the total time spent at a particular cluster in seconds. The angle of rotation divided by 36 characterises the cluster number. This odd way of portraying the cluster number is due to limitations of the Plotly library. Defining the AOI

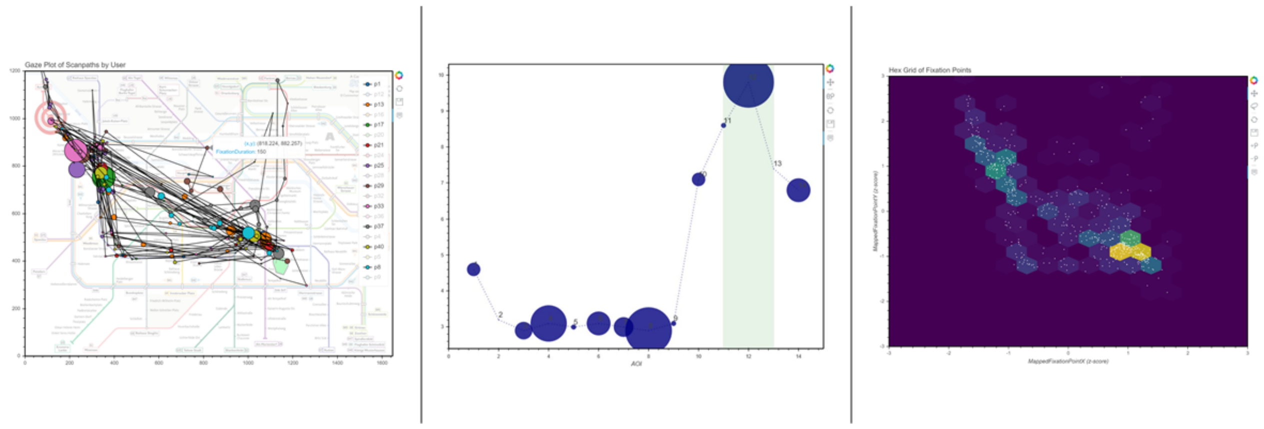


Fig. 4: Examples of certain visualizations that can be generated using the InfoVis tool: (1) gaze plot with individual scanpath toggling, (2) transition graph illustrating stimulus traversal in terms of AOIs, (3) hexbin heatmap indicating the density of visual attention in specific areas of a stimulus for all users.

clusters is done with Scikit-Learn's *k*-means clustering algorithm in the same way as with the Sankey diagram.

### 4.7 Interaction Techniques

The discussed InfoVis tool implements the following interaction categories as defined by Yi et al.:

- **Select**: For a selected stimulus, users can choose between two heatmap types to visualize natural data clusters based on personal preferences. If using the hex grid visualization, users can select a specific bin by hovering over them, which will change their color and alpha settings. This is achieved through the HoverTool function within the Bokeh library.

- **Explore**: Depending on the active plot type, users have the option of panning and changing the zoom level of the plot. The panning mode can be accessed by clicking the relevant button in the toolbar displaying alongside each plot. Zoom settings can be changed in a similar manner; the contour plot also supports zooming in on a rectangular area through a click-and-drag method. The web tool also implements a *Compare* tab, allowing two visualizations to be viewed in parallel.

- **Reconfigure**: When viewing the transition graph, users have the option of changing the bin size through the use of a slider; specifically, the slider determines the factor by which the total number of elements is divided, thus updating the bin size and subsequently the radius of the markers plotted. In the Sankey diagram, users can select the number of clusters to be defined in the *k*-means algorithm, changing the final layout. Finally, users can also change the color scale used in the heatmaps (from a list of the predefined Matplotlib palettes: *Viridis, Plasma, Magma*, etc.).

- **Encode**: Data points in the heatmap implementations are standardized; the resulting *z*-score indicates the deviation of data points (measured in standard deviations) from the mean. This is done to facilitate comparability to other datasets while also preserving the original shape of the data.

- **Elaborate**: Upon hovering over a specific bin in the hex grid visualization, an overlay will display the count of elements grouped in that bin, should the user wish to evaluate clusters more precisely than just relying on the color scale. Hovering over any point in the contour plot will also display the corresponding *z*-score. Hovering over any node in the Sankey diagram displays the incoming and outgoing flow count, as well as the total number of users originating from that source.

- **Filter**: For any of the implemented visualizations, users can choose to display only a restricted range of participants/scanpaths through a multi-select menu. This can be narrowed down further by using the lasso-select tool, which causes only those data points within the confines of the user-defined shape to be highlighted.

- **Connect**: When the user changes a variable (user, duration, or stimuli) the value will remain and be implemented in the other visualizations. Meaning that when the user switches from one visualization to another, the variables remain constant. This is implemented with the use of a session variable. The session variable retains its value as long as the user does not quit the browser. Once the user logs out of their account the session variables will return to the initial state (user: *p1*, stimulus: *Antwerpen*). Additionally, cluster definition is shared across the Sankey diagram and polar plot; they will both display information about the same clusters upon refreshing.

## 5 Metro Maps Application

To illustrate the relevance of the tool, eye movement data collected from a previous eye tracking study by Netzel et al. in 2017 with the use of various metro maps as visual stimuli was employed [24]. The overall objective of the study was to conclude whether differences in visual attention behavior were apparent between colored and gray-scale maps (coming to the conclusion that indeed they were); the data consists of fourty scanpaths per stimulus (twenty for each of the colored and gray-scale versions). Participants were requested to visually trace a path between two indicated points with no prior exposure to the stimulus at hand. After uploading the dataset, the tool can then be used to easily identify common scanning behaviors and strategies.

When booting up the tool, this dataset is loaded by default. The users can then access the main visualization pages through the navigation bar. If they wish to approach the data from different perspectives, there is a *Compare* tab (Figure 5) allowing them to choose any two (or more) visualizations and display them in parallel for a different subset of users and/or stimulus. Notably, this functionality can be used to provide great insight regarding the differences in scanning behavior between colored and gray-scale versions of the same map.

For this example, the map of Tokyo (*09_Tokyo_S1.jpg*) is chosen. Viewing the gaze plot is useful for the purposes of identifying prevalent scanning strategies, as the plot includes the stimulus as a translucent background. Basic information about the density of visual attention can

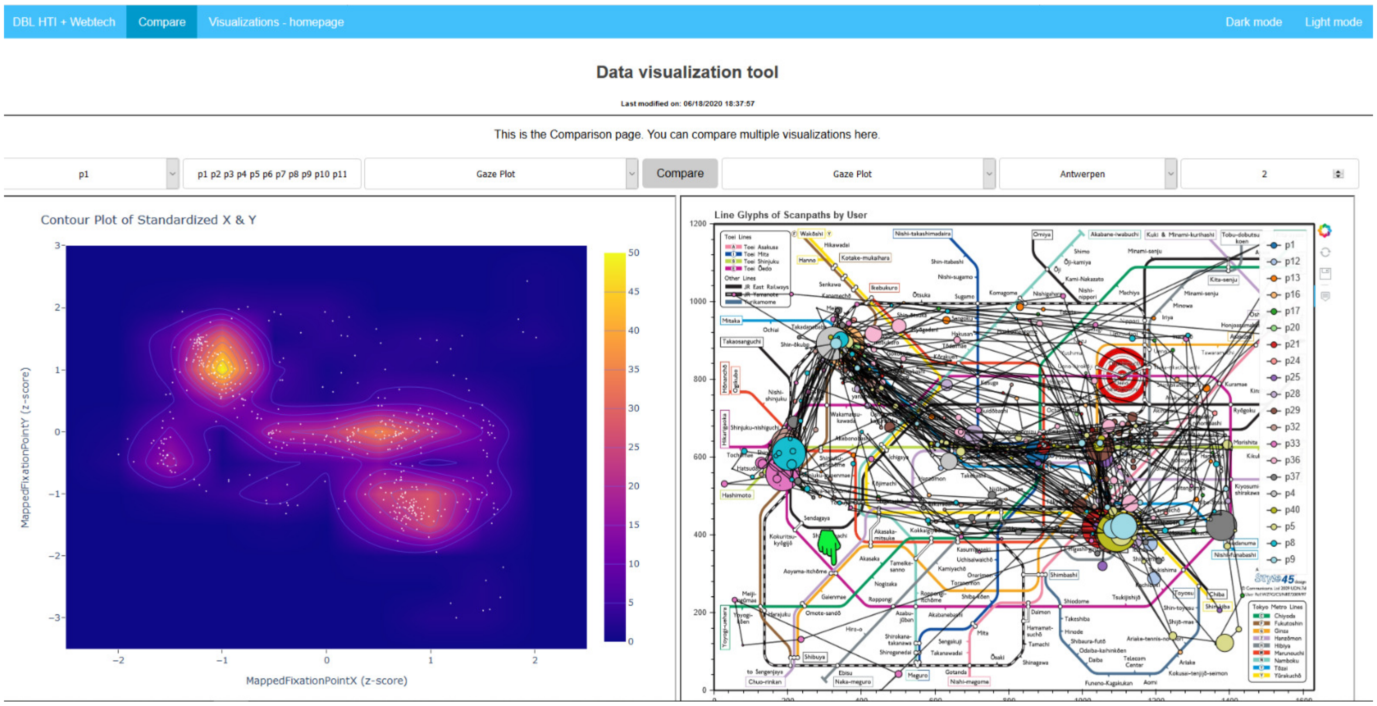

DBL HTI + Webtech
Compare
Visualizations - homepage
Dark mode
Light mode
Data visualization tool
Last modified on: 06/18/2020 18:37:57
This is the Comparison page. You can compare multiple visualizations here.
p1
p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11
Gaze Plot
Compare
Gaze Plot
Antwerpen
2
Contour Plot of Standardized X & Y
MappedFixationPointY (z-score)
MappedFixationPointX (z-score)
Line Glyphs of Scanpaths by User
p1
p12
p13
p16
p17
p20
p21
p24
p25
p28
p29
p32
p33
p36
p37
p4
p40
p5
p8
p9

Fig. 5: *Compare* tab showing the gaze plot and contour heatmap in parallel for the *09_Tokyo_S1.jpg* stimulus. See section 5 for application details.

be obtained by looking at the marker sizes and concentration, but is less apparent. At this stage, the transition graph can also be used to provide an alternate, broader perspective on traversal strategies, concentrating more on transitions as opposed to the exact scanpath. Switching to either of the heatmap implementations (or displaying them in parallel with the gaze plot) provides a solid foundation for understanding both scanning strategies and visual hot spots in the map. For example, matching the hot regions on the contour plot to the map display reveals that most users followed either a Euclidean diagonal path straight to the red marker, or the path corresponding to the Hanzōmon (pink) line. Toggling the individual scanpaths on the gaze plot confirms that this is indeed the case. Moreover, if the user wishes to further refine the dataset by locating outliers (i.e. scanpaths which are drastically different from the norm), such a view simplifies this task [6].

The heatmaps are also decent baseline tools to manually visualize AOIs. As mentioned, the tool automatically defines a user-defined number of AOIs through the *k*-means algorithm, with each AOI pertaining to relevant aspects of the map (e.g. the start and end markers, areas of interchange, and/or areas of dense visual attention). Visualizing the probability of AOI transitions is facilitated through the Sankey diagram; the width of individual flow links between AOIs directly corresponds to the number of users who made that transition. This can provide context to the behaviors revealed through analysis of the gaze plot and heatmap. For example, the links between two AOIs (e.g. $1 \rightarrow 3$ and $3 \rightarrow 1$) are comparable in width - this indicates that users provided the Tokyo stimulus were inclined to double-check their optimal solution to the problem.

In conclusion, the InfoVis tool is an acceptable platform for the purposes of exploratory data analysis concerning the general scanning behaviors and strategies associated with the metro maps. The included features could also assist in the creation of future metro maps.

## 6 Discussion and Limitations

As described in the previous section, the tool aptly enables exploratory data analysis regarding the general visual scanning behaviors and strategies apparent in the dataset. However, there remain several lackluster aspects of the tool wherein further improvements could be made.

### 6.1 Visualization Diversity

The diversity of the visualizations ensures that the user is not limited to a certain point of view. The fact that you are able to see and compare two different visualisations on our website gives the possibility of actually testing the results that are discussed in the paper that our project is based on. You can visualize the effect that a black and white map has on the way people use metro maps. Furthermore, considering that we have implemented five different visualisations such analysis can be done in a broader spectrum, concerning multiple types of quantifiable data.

### 6.2 Responsiveness

The InfoVis tool is logically limited by the programming language and packages used. The combination of Flask, Bokeh/Plotly and Python can sometimes lead to low responsiveness. While not invasive, it is a noticeable phenomenon in the tool. An majority of the functions used can have an upper asymptotic time complexity of $O(n^2)$ or $O(n^3)$ [25]. The combination of these functions is the culprit of the slow responsiveness.

### 6.3 Interactivity

The interactive elements are also quite limited. They are built upon HTML input forms that pass data on via Flask-routes to the Python files. This is a cumbersome and time-consuming process, meaning that these HTML forms are mainly used for passing through the input variables such as user(s) and stimulus. The more complex interactivity is hence implemented in Bokeh and Plotly itself. This does however limit us to the interactivity provided by these libraries. A possible way to improve this was to implement a Dash server on top of the Flask server. Dash is a Python framework for building web applications. It is built on top of Flask, Plotly and React. With the use of Dash the limitations introduced by HTML would be eliminated. However, it would contribute to an even larger time complexity, further slowing down the tool.

### 6.4 Flexibility

Another limiting factor is a lack of flexibility, due to the way the data is processed by the program. It needs to be ordered in a certain way in an Excel spreadsheet. You also can't upload many different files and then let the program combine them to make the visualizations work, to do this you would need to recode for this exact scenario. This limits the data you can explore using the tool; it also means that you need to know the original data base to be able to create a new usable one and if a mistake is made it would be quite impossible to remedy the error as there is no error being reported on the website. Error reports could be a future feature to look into as they are essential in many other possible scenarios.

## 7 Conclusion and Future Work

In conclusion, the InfoVis tool allows users to conduct extensive exploratory data analysis on eye-tracking datasets (specifically, data from the provided Metro Maps experiment) through the use of interactive visualizations and certain user experience features. Visualizations employed include a gaze plot, transition graph, Sankey diagram, polar chart and heatmaps, generated with a combination of Plotly and Bokeh functions. Among other features, users can adjust visualization appearance, filter the dataset and compare visualizations in parallel. The tool also includes numerous options to enhance user experience, such as a login tool and the ability to toggle between light and dark mode. As it stands, the tool is a good starting point in the analysis of eye-tracking data. However, it remains a work-in-progress, as there are numerous changes that could be made to improve performance and usability, as described in the previous section. To summarize, these changes include: improving responsiveness by implementing a different language and/or changing algorithmic structure to reduce time complexity, implementing a more substantial interactivity framework and including support for different data formats.

## References

[1] F. Beck, M. Burch, and S. Diehl. Matching application requirements with dynamic graph visualization profiles. In *Proceedings of 17th International Conference on Information Visualisation, IV*, pp. 11–18. IEEE Computer Society, 2013.

[2] F. Beck, M. Burch, T. Munz, L. D. Silvestro, and D. Weiskopf. Generalized pythagoras trees: A fractal approach to hierarchy visualization. In *Proceedings of International Conference on Computer Vision, Imaging and Computer Graphics - Theory and Applications - International Joint Conference, VISIGRAPP*, vol. 550 of *Communications in Computer and Information Science*, pp. 115–135. Springer, 2014.

[3] T. Blascheck and T. Ertl. Towards analyzing eye tracking data for evaluating interactive visualization systems. In *Proceedings of the Fifth Workshop on Beyond Time and Errors Novel Evaluation Methods for Visualization - BELIV 14*, 2014. doi: 10.1145/2669557.2669569

[4] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl. Visualization of eye tracking data: A taxonomy and survey. *Computer Graphics Forum*, 36(8):260–284, 2017. doi: 10.1111/cgf.13079

[5] P. Blignaut, E. J. V. Rensburg, and M. Oberholzer. Visualization and quantification of eye tracking data for the evaluation of oculomotor function. *Heliyon*, 5(1), 2019. doi: 10.1016/j.heliyon.2019.e01127

[6] M. Burch, K. Ayush, and K. Mueller. Finding the outliers in scanpath data. In *11th ACM Symposium on Eye Tracking Research Applications (ETRA)*, pp. 1–5, 2019. doi: 10.1145/3314111.3318225

[7] M. Burch, M. Hlawatsch, and D. Weiskopf. Visualizing a sequence of a thousand graphs (or even more). *Computer Graphics Forum*, 36(3):261–271, 2017.

[8] M. Burch, M. Höferlin, and D. Weiskopf. Layered TimeRadarTrees. In *Proceedings of 15th International Conference on Information Visualisation, IV*, pp. 18–25. IEEE Computer Society, 2011.

[9] M. Burch and A. Jalba. Qualitative evaluation of the aoi rivers. 2020.

[10] M. Burch, N. Konevtsova, J. Heinrich, M. Hoeferlin, and D. Weiskopf. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2440–2448, 2011. doi: 10.1109/tvcg.2011.193

[11] M. Burch, A. Kull, and D. Weiskopf. Aoi rivers for visualizing dynamic eye gaze frequencies. *Computer Graphics Forum*, 32(3.3):281–290, 2013. doi: 10.1111/cgf.12115

[12] M. Burch, S. Lohmann, F. Beck, N. Rodriguez, L. D. Silvestro, and D. Weiskopf. Radcloud: Visualizing multiple texts with merged word clouds. In *Proceedings of 18th International Conference on Information Visualisation, IV*, pp. 108–113. IEEE Computer Society, 2014.

[13] M. Burch, C. Müller, G. Reina, H. Schmauder, M. Greis, and D. Weiskopf. Visualizing dynamic call graphs. In *Proceedings of the Vision, Modeling, and Visualization Workshop 2012*, pp. 207–214. Eurographics Association, 2012.

[14] M. Burch and D. Weiskopf. A flip-book of edge-splatted small multiples for visualizing dynamic graphs. In *Proceedings of the 7th International Symposium on Visual Information Communication and Interaction, VINCI*, p. 29. ACM, 2014.

[15] M. Burch and D. Weiskopf. On the benefits and drawbacks of radial diagrams. In W. Huang, ed., *Handbook of Human Centric Visualization*, pp. 429–451. Springer, 2014.

[16] P. Capello. Chapter 6: Transition graphs. Department of Computer Science, University of California, Santa Barbara.

[17] I. Dabbura. K-means clustering: Algorithm, applications, evaluation methods, and drawbacks. Towards Data Science, 2018.

[18] A. T. Duchowski, M. M. Price, M. Meyer, and P. Orero. Aggregate gaze visualization with real-time heatmaps. In *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA 12*, 2012. doi: 10.1145/2168556.2168558

[19] H. Jarodzka, K. Holmqvist, and M. Nyström. A vector-based, multidimensional scanpath similarity measure. In *Proceedings of the 2010 Symposium on Eye-Tracking Research and Applications - ETRA 10*, 2010. doi: 10.1145/1743666.1743718

[20] M. A. Just and P. A. Carpenter. A theory of reading: From eye fixations to comprehension. *Psychological Review*, 87(4):329–354, 1980. doi: 10.1037/0033-295x.87.4.329

[21] K. Kurzhals, M. Hlawatsch, F. Heimerl, M. Burch, T. Ertl, and D. Weiskopf. Gaze stripes: Image-based visualization of eye tracking data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1005–1014, 2016. doi: 10.1109/tvcg.2015.2468091

[22] M. L. Lai, M. J. Tsai, F. Y. Yang, C. Y. Hsu, T. C. Liu, S. W. Y. Lee, and C. C. Tsai. A review of using eye-tracking technology in exploring learning from 2000 to 2012. *Educational Research Review*, 10:90–115, 2013.

[23] N. Minaie. K-means clustering for unsupervised machine learning. Towards Data Science, 2019.

[24] R. Netzel, B. Ohlhausen, K. Kurzhals, R. Woods, M. Burch, and D. Weiskopf. User performance and reading strategies for metro maps: An eye tracking study. *Spatial Cognition Computation*, 17(1–2):39–64, 2017.

[25] M. K. Pakhira. A linear time-complexity k-means algorithm using cluster shifting. *2014 International Conference on Computational Intelligence and Communication Networks*, 2014. doi: 10.1109/cicn.2014.220

[26] K. Schwaber and K. Sutherland. The definitive guide to scrum: The rules of the game. The Home of Scrum, 2017.

[27] C. Vehlow, M. Burch, H. Schmauder, and D. Weiskopf. Radial layered matrix visualization of dynamic graphs. In *Proceedings of 17th International Conference on Information Visualisation, IV*, pp. 51–58. IEEE Computer Society, 2013.

[28] E. Weisstein. Simple graph. MathWorld–A Wolfram Web Resource.

[29] J. S. Yi, Y. A. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007. doi: 10.1109/tvcg.2007.70515