Increasing Our Understanding of Attention Patterns: A Widely Accessible Interactive Visualization Tool

Nick Broeks, Lulof Pirée, Nynke Boonstra, Paul Vlaswinkel, Silke Franken, and Vince van Wijk

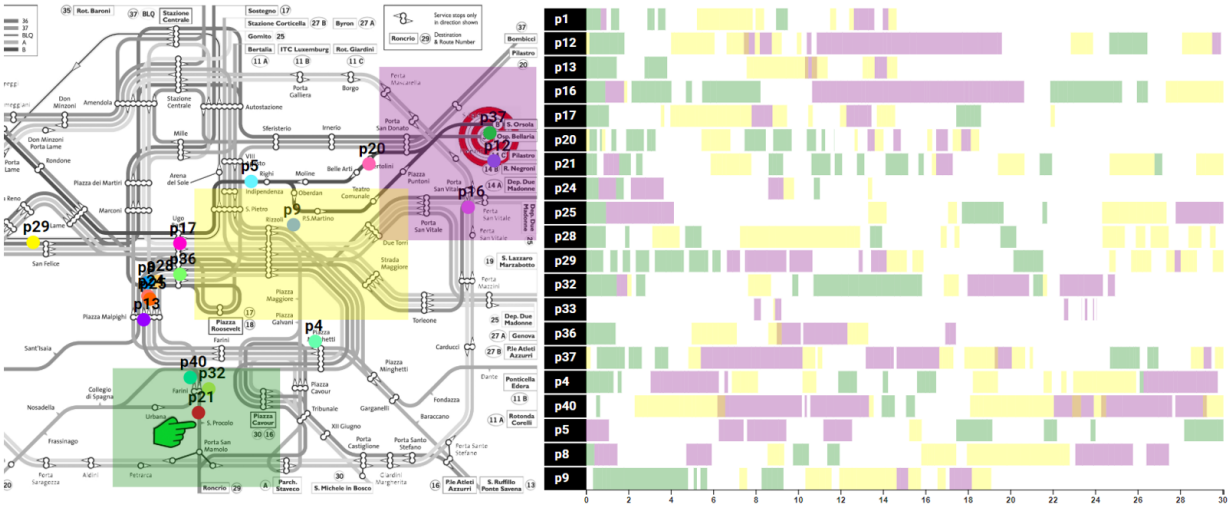

Fig. 1: A combination of the scarf plot (right), taking data from the colored areas defined in the beeswarm (left) The displayed stimulus is from the metro maps data set, and is named `11b_Bologna_S2.jpg`. The scarf plot shows which user looked at what colored AOI over time.

Abstract— GazeMap is a suite of eye tracking data visualization tools accessible via a website. Users can upload their own data set of time-stamped coordinates of eye focus points, and inspect the data from several perspectives simultaneously. With this website, we were aiming to create a user-friendly and easy accessible tool to interpret attention patterns and facilitate data analysis for large eye tracking data sets. The available visualization techniques are a 2D density plot, a scanpath, a bee swarm, and a scarf plot. The latter technique is based on areas of interest, which can be selected by the user. All techniques are supported by several interaction techniques. Moreover, users can select data in one visualization, and the website will highlight the same data points in all active visualizations for accessible comparison of data subsets. The tool also provides functions that make it possible to upload both private or public data sets, and can generate URLs to share the data and settings of customized visualizations.

Index Terms—Eye tracking, Visualization techniques, Human-centered computing, Public transport maps.

1 Introduction

The science of recording human eye movement has grown from its origins in the nineteenth century [22] to a field touching many areas of research. Nowadays, sophisticated tools and programs for generating eye-movement data are available (see for example [7]). Knowing and understanding people's gaze patterns has helped to improve society, by facilitating people's daily experiences. The use of eye tracking devices within research is commonly used in cognitive sciences, and has led to many great discoveries [13, 21]. Nonetheless, eye tracking data has proven useful in many more scientific areas, and study topics range from dyslexic reading behavior to the impact of scrolling past Facebook messages [18, 25].

The focus of this work lies on eye movement data, a subset of eye tracking data. Eye movement data concerns people's gaze points with associated moments in time, as recorded by eye tracking devices on participants who observe a certain stimulus. After the recording, the data can be aggregated into the two components of eye movement: fixations and saccades [2]. It is often important to create an appropriate visualization of the data to obtain qualitative information related to the research question. Eye tracking systems often already provide limited visualization options, but these are usually insufficient for fully understanding the data; eye movement data sets can quickly become large, which often results in overplotting and visual clutter [19]. In an attempt to provide all researchers with accessible and complete tools, we present a web-based solution to visualization software. This tool does not require installation, is easy-to-use, and most importantly, it presents all visualization techniques necessary to enable researchers to analyze every aspect of their eye movement data. On top of that, it is also possible publish their data in order to easily share their insights with all interested parties. Our tool is available at `https://www.p4ul.nl/dbl/`.

A complete eye movement data set of the form that is used in this paper generally has many data points containing the following attributes: a *timestamp*, a *fixation duration*, a *fixation point* (x- and y-coordinate), a *user ID* (uniquely identifying a participant of the measurements), and a *stimuli name*. Optionally, a *description* may be provided, related to the studied task given to the participants. Usually, an eye tracking study

- *n.broeks@student.tue.nl*
- *n.boonstra@student.tue.nl*
- *l.l.piree@student.tue.nl*
- *p.r.vlaswinkel@student.tue.nl*
- *s.w.franken@student.tue.nl*
- *v.f.v.wijk@student.tue.nl*

investigates the relation between the semantic context of the stimulus and human attention patterns, possibly combined with the *description*. To prevent these stimulus semantics becoming hidden underneath the visualization overlay, users are able to mark these so-called areas of interest (AOIs) [2]. Our tool not only supports single rectangular AOIs, but also compound AOIs of multiple rectangles. Furthermore, sets of data points can be selected to identify the same data across different visualizations.

In isolation, visualization techniques are typically limited in value (for example, heatmaps do not show any sequential or temporal information [2], and gaze plots are susceptible to visual clutter [5]). Our tool addresses this issue by providing four visualization techniques concurrently: a *scanpath*, a *2D density plot*, a *bee swarm* and a *scarf plot*.

We describe the construction process of this tool, the intuitive use of the interface of the website and visualization techniques, and the knowledge that can be gained while using the tool. Example data is used from an eye tracking study about public transport maps in major cities [14].

2 Related Work

Many visualization techniques exist to analyze the large amount of trajectories in eye tracking data [2]. One of the most popular techniques is a scanpath [2]. It shows the location, order, and often also the duration of fixation points. The fixation points are indicated by circles and their radius indicates the duration of the fixation. The saccades are indicated by lines that connect the circles to each other. Although scanpaths provide a lot of information concerning gaze trajectories, their large disadvantage is that they can easily lead to visual clutter [19]. To address this issue, we implemented a dispersion-based algorithm to cluster the data points into more general fixation points [20]. In addition, users can interactively change the dispersion value to a minimum of zero, at which the original gaze map is plotted. This way there is an option to remove visual clutter, but the original scanpath is still preserved.

Another popular visualization technique is a heatmap [2], which gives an overall view of the fixation points. It clearly shows the most interesting areas of a stimulus. Spakov and Miniotas [26] used the heatmap as a basis for their 2D density plot. Since some details get lost when all data points are combined into one visualization, we attempted to optimize this technique by adding the option to still be able to identify loci on the underlying stimulus.

In contrast to a heatmap, in which data points are aggregated over time, the bee swarm visualization technique provides an overview of all fixation points on a timeline [2]. It shows the location of fixation points at a selected point in time.

In their paper, Blascheck et al. distinguish between two main categories of visualization techniques: point-based and AOI-based methods [2]. The previously mentioned visualization techniques are all point-based. However, AOI-based methods are useful for including semantic information of the stimuli into the analysis: users are able to select areas of interest within a stimulus, after which that information can be integrated in the visualization. An example of such an AOI-based visualization is the scarf plot. It shows at which predefined AOIs different observers were looking over time.

The visualization techniques used for our tool are based on the four techniques described above. Combining several techniques will lead to an enhancement of the overall analysis. In other words, the whole will be greater than the sum of its parts; by merging several techniques of analysis, the probability of gaining new insights is increased. By making use of both point- and AOI-based visualization techniques, users can have a detailed look at specific data points, but also focus on areas of a stimulus with semantic meaning [2].

A great advantage of providing a web-based tool is that it allows direct interactions between the visualizations and the user, and between different visualizations. Yi et al. [24] describe seven general categories of interactions in information visualization, and address their importance. For instance, when the user is able to select certain areas or to filter the data, this will lead to a more efficient and personalized experience. All implemented interaction techniques fall in one of these seven categories.

3 Graphical User Interface (GUI)

The appearance of the user interface is shown in Figure 2. The tool consists of two important parts, a navigation menu and a page of 'panels'. The navigation menu can be found on the left side, and allows the user to upload data and navigate through the main functions of the website. The larger right pane forms a grid of 'panels', in which the visualizations of choice can be displayed. The layout, order and amount of panels can be customized. Individual panels are resizable, and also support their own setting and filter options. Active visualizations can be moved from one panel to another, by means of dragging the 'tab' located at the top of a panel. A panel can hold multiple visualization-tabs, of which one is shown at a time.

A user can choose which visualization technique is used within a panel-tab by means of a drop-down menu. After choosing a visualization, the user can select the preferred data set and stimulus. Optionally, specific participants can be filtered out of this instance of the data set.

The navigation menu, shown on the left side of Figure 2, presents the following functionalities:

- **Upload File:** This button allows the user to choose a `.csv` file, `.zip` file, a `.jpg` or a `.png` file to upload to the tool. Consequently, it becomes possible to choose this data for every selected visualization technique. After selection of a data set, all related stimuli will appear in the image selection menu. By uploading their data set, users also have the option to make it publicly available, which means that every visitor of the website will be able to access and visualize the data.

- **Generate Random Data:** Pressing this button will generate a data set, based on the amount of users and data points per user. These parameters can both be set by the user. The data set will be bound to a stimuli chosen by the user. This feature is mostly intended for testing and benchmarking of the tool.

- **Browse Catalog:** This button opens a list of public data sets. When users decide to make their data set publicly available, it will appear in the catalog. This makes it easier for visitors to have access to interesting data sets of other researchers. After selecting a file, it becomes possible to transfer the file to the user's local storage for usage.

- **Export Instance:** This button saves the current settings, and shows a new URL. The tool at this URL has the same settings as the user currently has. This makes sharing data and findings easier.

- **File Manager:** Here users can see all files they have uploaded, or imported from the catalog. This way it is possible to easily keep track of all uploaded data sets.

- **Add Panel:** This button constructs a new tab in which another visualization can be shown. This option makes the website easier to personalize.

- **Change Layout:** This button opens a popup menu, where you can choose the layout. This option will also make the website easier to personalize and is implemented for better usability.

- **Help:** By pressing this button, a description on how to use the website and the different visualization techniques will be shown in a pop-up window. For every provided visualization method there is an option to follow an interactive tutorial about the meaning of all sliders and switches.

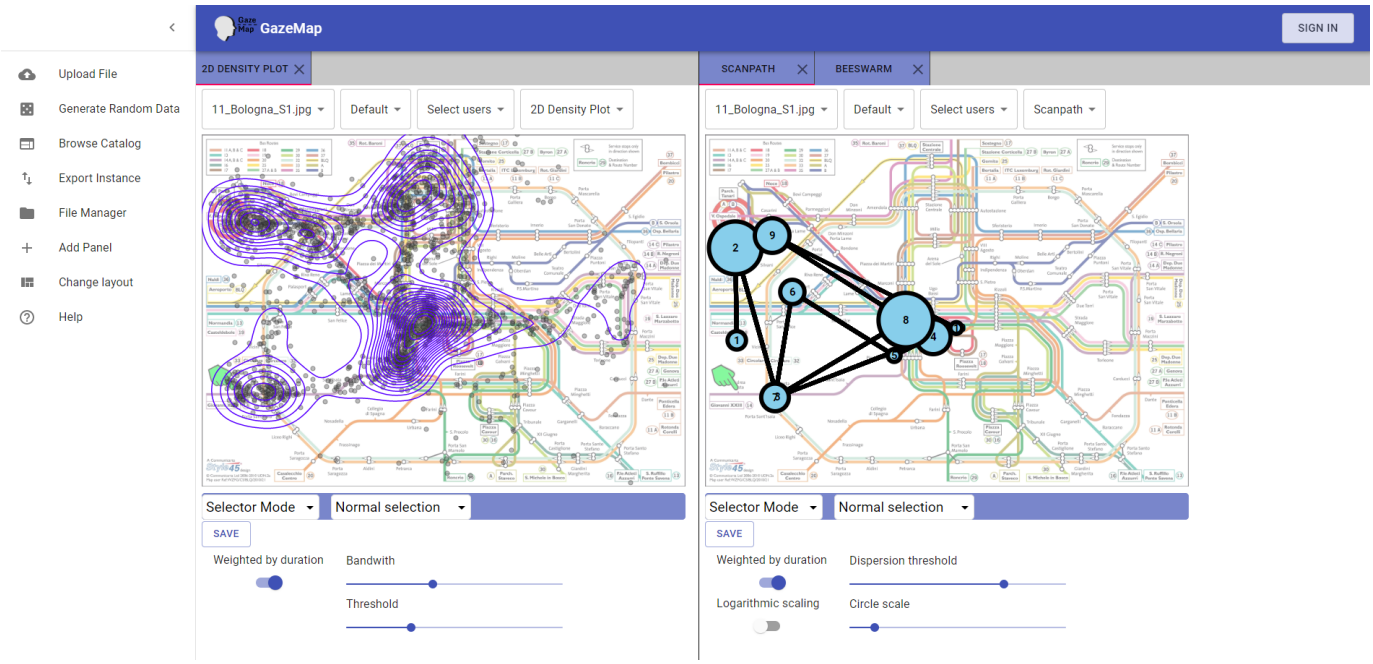

Fig. 2: The Graphical User Interface of our interactive visualization tool. The navigation menu on the left is unfolded. The data was taken from the metro maps data sets. In the right panel, a 2D density plot is created on top of the stimulus named `11_Bologna_S1.jpg`. The sliders are left at default values. All users are selected. On the left, a scanpath is created on top of the same stimulus, using data from participant `p36`. The dispersion threshold is set to 571 and the circle scale to 0.58. Another visualization-tab has been opened in the right panel, displaying a beeswarm.

Implementation

The logical structure of the front-end code base has been included in Appendix A in Fig. 10. For building the user interface, we used *React* [23], together with *Material-UI* [15]. React is a flexible component-based JavaScript framework. With knowledge about JavaScript, React is easy to learn. Using this library resulted in a professional looking website, which efficiently updates and renders the necessary components when a change in the data is made. Material-UI adds React components which have a nice default look, but can be easily modified for a highly customizable look.

4 Visualizations

This section describes the four visualization techniques implemented in our web-based tool.

4.1 2D Density Plot

A 2D density plot effectively avoids the problem of overplotting in a scatter plot, by aggregating close data points within dense areas. Density plots provide an effective way to prevent visual clutter and visualize the hidden pattern behind the data [9]. Information about individual data points becomes lost as a side effect, but this is often compensated by the improved readability. An example of a 2D density plot is shown in Figure 2 (left).

4.1.1 Implementation

The D3.js JavaScript Library [4] comes with a built-in option to create 2D density charts. In the area of eye tracking visualization, the pixel locations on the stimulus in the x- and y-direction are interpreted as a scatter plot, which is used to generate a density chart. To highlight the relation between the semantics of the stimulus and the data, the corresponding stimulus is added as a background to the chart.

Before creating contour lines, all data points with their corresponding coordinates are aggregated over time and participants. In this process the element of fixation duration is lost. The contours that are created next represent the amount of data points on each part of the grid. To compute the density, a library is needed called d3-contour plugin [3]. This library can compute contour polygons from an array of numeric values, with the principle of marching squares. This technique will result in "a piecewise-linear approximation to a two-dimensional object" [12]. The function called `d3.contourDensity()`, which implements fast two-dimensional kernel density estimation, generates paths that can be rendered using `d3.geoPath()` [10].

4.1.2 Interactions

The 2D density plotting algorithm uses several parameters than can be reconfigured by the user using the GUI. The bandwidth, defined as the standard deviation of the Gaussian kernel [3], can be adjusted with a slider, and roughly corresponds to the scope of the contour lines.

The amount of visible contour lines can be adjusted as well, by changing the *threshold* value. The amount of thresholds roughly defines the amount of contour lines. More contour lines result in a more detailed plot, but also causes more clutter.

The last option that is provided lets the user choose whether the density plot should be weighted by duration. When this option is selected, not only the fixation locations, but also the fixation *durations* are included in the calculation of the density plot.

4.2 Scanpath

A scanpath, also known as a gaze plot, visualizes the path an observer has followed with their eyes, by showing *fixation points* and *saccades*. Fixation points are locations of the stimulus where the observer has fixated his or her gaze for an extended duration. A saccade is a rapid eye movement, moving from one fixation point to another. A sequence of fixation points and saccades forms a scanpath, which is visualized by representing the fixation points as circles and the saccades as lines. The circles have a radius proportional to the duration the participant was fixated on the point. Additionally, the circles are numbered to clarify the order of the fixation points [8]. A gaze plot is, together with a heatmap, the most used visualization technique among experts [2].

4.2.1 Implementation

A scanpath created by our implementation is shown in Figure 3 and Figure 2 (right). Salvucci and Goldberg [20] describe and compare various algorithms to identify fixation points. The choice was made

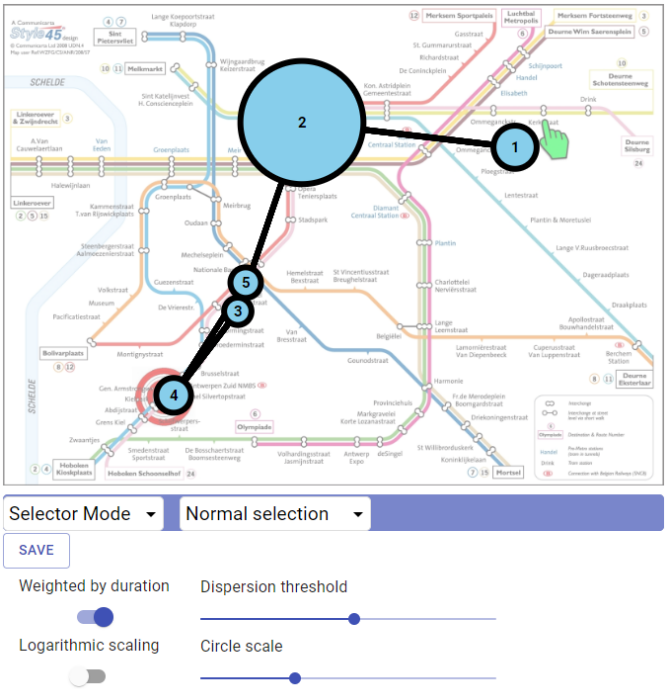

Fig. 3: Scanpath from user p1 created on top of the stimulus `01_Antwerpen_S1.jpg` (a map of the metro lines in Belgian city of Antwerp), which was taken from the metro maps data set. The dispersion threshold is set to 416, and the circle scale to 1.6.

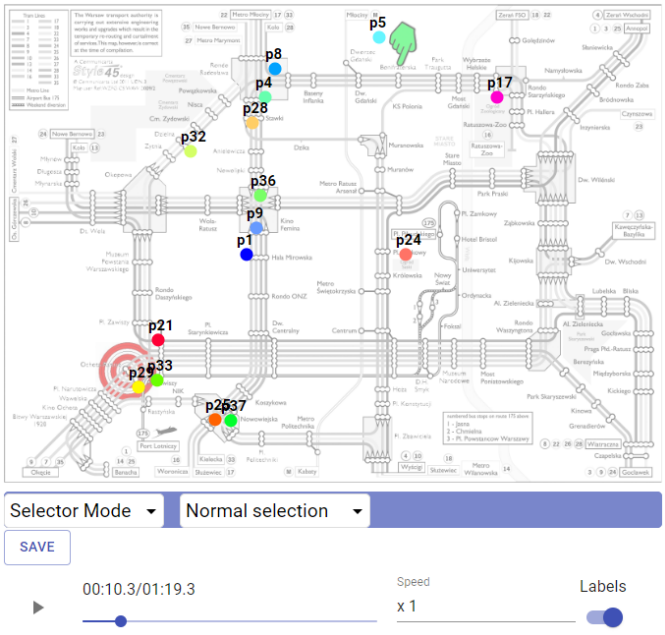

Fig. 4: Beeswarm created on top of the stimulus `23b_Warschau_S2.jpg` (a map of the metro lines in the Polish capital city Warschau), which was taken from the metro maps data set. All users are selected.

to implement the *Dispersion Threshold I-DT* algorithm. According to the paper, this algorithm provides 'very good' accuracy and robustness, and 'good' speed and ease of implementation.

When a scanpath is rendered, the first performed action is the drawing of the lines between the points. After that, circles are drawn at the locations of the the fixation points. The location of these fixation points is found by calculating the 'centroid' of the points in a specific window. The window is a set of points that are close to each other, defined by the dispersion threshold. The paper does not specify how to calculate the 'centroid of the window points' [20]. The decision was made to calculate the centroid by calculating the average x- and y-coordinates of the points in the window. These averages are the coordinates of the centroid. The implementation also supports calculating the centroid with weighted values, where every coordinate is weighted by the fixation duration.

4.2.2 Interactions

The scanpath allows a user to adjust several parameters of the visualization algorithm (an interaction technique which Yi et al. describe as *reconfiguring* [24]).

The most important parameter of the scanpath is the *dispersion threshold*. This integer value determines how far points can be apart to still be clustered in one fixation point. Hence a larger value will usually lead to fewer and larger displayed fixation points (unless the measured points are far apart).

The user can also set the circle size though another slider, which linearly changes the relative diameter of all fixation point circles. Because there might be large differences in diameter, there is a switch-button to scale the circles logarithmically with the number of associated points instead of linearly. Another switch allows to weight the points by duration, that is, points with a longer recorded fixation contribute more to the diameter of the rendered point than points with a shorter duration.

4.3 Bee Swarm

A bee swarm shows the location of fixation points of observed participants over a stimulus over time (see Fig. 4). The fixation points of participants are shown as little dots, accompanied with the name of the participant. The relative time from the moment a participant started viewing the stimulus has been taken as a starting point, and by synchronizing these times for all participants it becomes possible to compare the order in which stimuli areas were observed, and for how long. This technique acts as an extension of the density plot and gaze plot, by adding a temporal dimension.

4.3.1 Implementation

For every point in time (relative from the starts of the measurements) and for every selected participant, a small circle is created in a distinct color. All these circles, together with the corresponding name of the participant, are shown on the stimulus at the coordinates of the user's fixation point at the selected point in time. When the timestamp changes, the positions of the circles are updated to the fixation point of the participant at the new time.

The colors of the circles are set using a function which takes the number of selected participants as a parameter. This function creates a range of colors which are sufficiently distinguishable from each other. This is possible by setting the RGB value of the colors in a way that one of the color values will always be 0, and one of them will always be 255. The resulting bright color palette can be seen in Fig. 4.

4.3.2 Interactions

The bee swarm makes use of a time dimension, which the user can control via a slider and a play/pause button. The slider shows the currently displayed (relative for each user) point in time, and can be used to manually select another point in time. The play button makes the visualization chronologically run through all the recorded time points, at an user-adjustable speed. Via a switch-button it is possible to show or hide the labels of the users next to their respective points on the map.

4.4 Scarf Plot

The scarf plot shows at which AOIs different observers are looking over time. Every selected AOI attains a different color. A separate time bar is shown for every participant. Within these bars the colors are visible that correspond to the AOI to which the observer is looking at

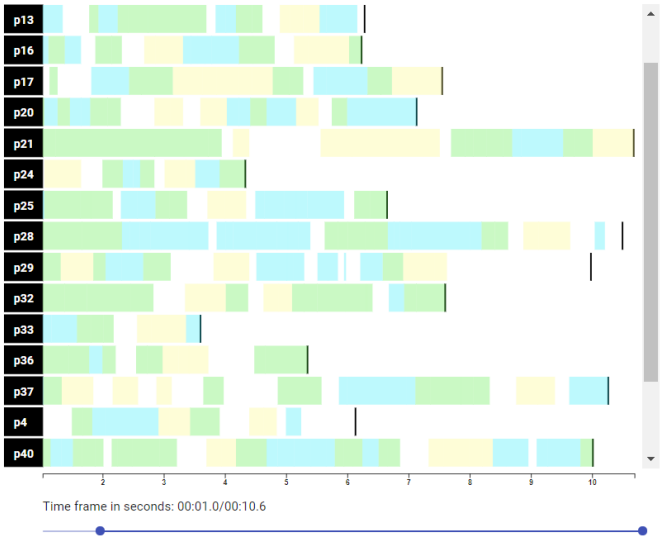

Fig. 5: A demonstration of the scarf plot based on the three AOIs selected in Fig. 7. The colors of the rectangles correspond to the colors of the AOIs and the slider on the bottom can be used to adjust the selected time frame. The black vertical lines indicate when a participant stopped looking at the stimulus (`01_Antwerpen_S1.jpg`, a map of the metro lines in Belgian city of Antwerp). Although all users were selected, they are not all visible. The scroll bar on the right can be used to view all users.

every indicated moment in time [2]. The black vertical lines indicate the time at which a user stopped looking at the stimulus. The scarf plot can be used to compare the gazing behavior of different observers to each other.

4.4.1 Implementation

A scarf plot created by our implementation is shown in Fig. 5. To visualize the scarf plot, a user first has to create one or more AOIs in one of the other visualizations. The scarf plot receives the selected areas, and checks for each available data point whether it is located in one of the selected AOIs. For every data point that is located inside an AOI, a segment of the time bar of the related participant is colored after the AOI, located on the timestamp of the data point. The width of the colored segment corresponds to the associated *fixation duration* of the data point.

4.4.2 Interactions

When a user changes the selected subset of 'users', the number of rows and their height automatically adjust so that the available display space is optimally used. When the number of selected users reaches a certain threshold, a vertical scroll bar appears to the right of the scarf plot. This interaction technique is also knows as panning, which belongs to the interaction category *explore* [24].

Another interaction technique that is specifically used for the scarf plot is the range slider. By moving two dots, a user can adjust the start and end time of the time frame in which data is displayed. This interaction is also known as the Direct-Walk technique, which belongs to the interaction category *explore* [24]. This option allows users to focus on specific points in time or to view the whole trajectory, which makes it easier to notice trends in fixation behavior.

5 General Interactions

Our tool supports all seven types of interactions as classified in [24]. Several of the implemented interaction techniques interact with different visualizations concurrently. As a result, the spatial, temporal, and individual elements of the data set can be easily compared, which would not be possible when using only one visualization technique. The following techniques are available in multiple visualizations: *selecting*

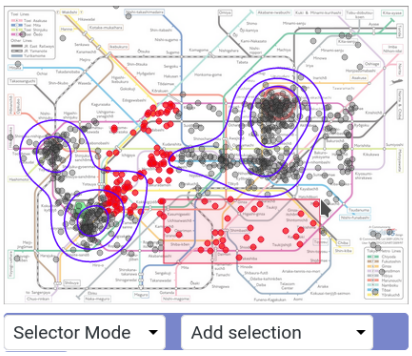

Fig. 6: A user using the *Addition* function to select points in the 2D density plot (on the stimulus `09_Tokyo_S1.jpg`).

(including 'brushing') of data points, *reconfiguring* visualization parameters, *connecting* data points chronologically and *filtering* by the user-attribute of data entries. The following section will go more in depth about the selection tool, the filtering options and the interaction called 'details-on-demand'.

5.1 Selections

One important objective of the visualization tool is to ease the task of comparing the same subset of data in different visualizations. A user may, for example, wish to examine whether a cluster in one visualization also appears as a cluster in another visualization. An intuitive way to accomplish subset comparisons is to allow a user to select data in a visualization where he/she can quickly identify the data points of interest, and then highlight the selected points in all visualizations (this is also known as 'brushing').

In our tool the user can make a selection by creating a transparent rectangle between two points using the cursor (similar to icon selection implemented in many desktop environments). The data contained within the rectangle can be selected in two different ways, at request of the user, as described below.

5.1.1 Selecting Points

The first method of applying a selection is to select the data points entries contained within the drawn rectangle: this results in the same data points being highlighted in red in all active visualizations, regardless of their relative positions (see Fig. 6). This type of selection applies to the scanpath, the 2D density plot and the bee swarm, as these visualizations represent the data by points rendered on top of a stimulus. Note that the rendered points in the scanpath represent multiple data entries: selecting one such 'cluster' will highlight all the containing points in other visualizations. And vice versa, the clusters will be highlighted as soon as at least one selected data entry is contained within the cluster.

Rectangular selections may by themselves be limited to subsets that fit in a rectangle, but with a single added setting in a drop-down menu the user may create any complex selection. This 'selection-mode' defines how data is added or removed from the set of selected data. The following modes are available:

- **Normal**: the previous selection is erased, and data within the rectangle becomes the new active selection.
- **Addition**: data within the selection rectangle is *added* to the active selection.
- **Inverted**: the previous selection is erased, but all and only all data *outside* the selection rectangle becomes the new active selection.
- **Subtraction**: data within the selection rectangle is *removed* from the active selection.

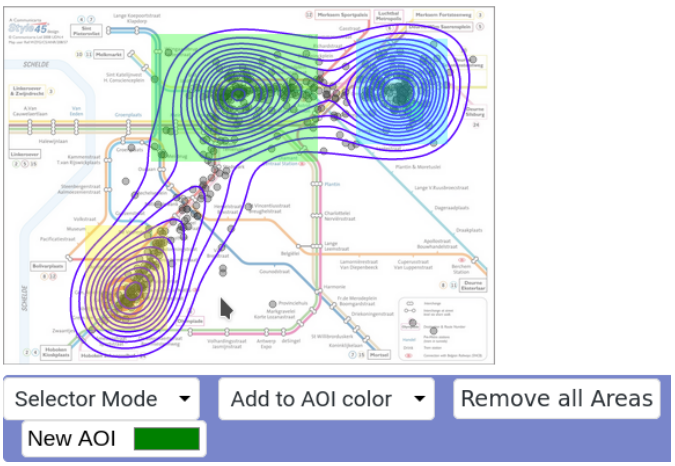

Fig. 7: A demonstration of three active selections with custom colors. The button *Add to AOI color* allows the user to add a new selection to an existing color, and the button *New AOI* allows users to define new colors by means of a pop-up color picker.

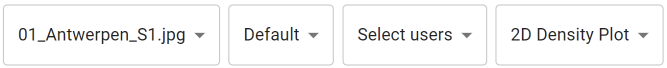

Fig. 8: The selection buttons visible in the GUI, at the top of each panel. These buttons allow the user to filter on stimulus, data set, users and visualization technique.

5.1.2 Selecting Areas of Interest

The other mode of selecting data, as supported by our tool, is to store the user-drawn rectangles as Areas of Interests. These AOIs (and the data they contain) will be used as input to generate the scarf plot, and are each identified by their color (see Fig. 7).

After drawing a rectangle, the user can choose to mark it with a certain color. The default color is green, but the user can define more colors by using a color-picker. The rectangle will change its color and persist when the user draws a new rectangle. Multiple rectangles can be given the same color, which will be interpreted as one (discontinuous) AOI by the scarf plot. Similar to highlighting points, the colored rectangles will be synchronized between the different visualizations that render the stimulus.

5.2 Filtering

After a data set is loaded, a user can choose a stimulus from the data set and a subset of 'users' (by their labels as recorded in the data set) whose data to display. These functions are available through buttons located in the top of each panel (as shown in Fig. 8). The leftmost button (displaying the name of the current stimulus) can be used to choose a stimuli, and the 'Select users' button allows a user to check or uncheck users whose data to display.

5.3 Labels for Additional Details

The tool will show a label with additional information about a single data point, when a user hovers the cursor over the point (see Fig. 9). All visualizations except for the scarf plot support this feature. The displayed information covers the recorded position, the user, and the timestamp associated with the point. In the scanpath there are more data entries associated with a single point, so the total duration instead of a timestamp will be displayed.

6 Metro Maps Application Example

We provide an example application to a real-world data set to demonstrate the features and uses of our tool. The examined data stems from an eye tracking study about metro maps [14]. This study looked at three different factors of the stimuli. The first factor was color-coding,

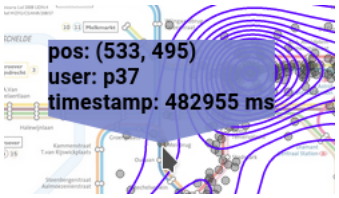

Fig. 9: A label with additional information pops up when a user hovers with the cursor over a data point in the 2D density plot, the scanpath and the bee swarm.

and for this reason half of the stimuli were gray-scaled public transport maps rather then fully colored ones. To examine the other factors - map complexity and task difficulty - all maps were divided in three levels of complexity and were assigned a task difficulty. The recorded data set consists of 188,126 entries, for a total of 96 different stimuli. We show how the different visualizations can be used to explore different aspects of the data set. It should be noted that more insight in individual data points or sets of data points can be obtained by using the selection mechanisms, to compare the same data across visualizations.

It is common to start by utilizing the *2D density plot*, to get a general overview of the data, the apparent areas of interest and to detect possible outliers. When selecting this option, data of all participants will be shown on top of the first stimulus in the data set. When utilizing multiple panels, it allows the user to investigate differences across stimuli or settings. One aspect that seems to reappear is the fact that most data points are cluttered around the metro lines between the indicated start and end point.

The *scanpath* is most effective when used for one participant at a time. Drawing conclusions from the data based on factor comparison is harder when using this method, since the factors of color-coding and task difficulty were tested using a between-subjects design. However, relative fixation duration for specific locations can be compared for the same participant.

Utilizing the *bee swarm* visualization technique can provide more insight into the general time-related aspects of the data. For static stimuli like metro maps[1], it is possible to look at the average time it took the participants to complete the task, and to compare for example the fixation locations during start, middle, and end.

Finally, the *scarf plot* provides the most relevant information for the hypothesis that people spend more time on gray-scale than colored stimuli, because it depicts relative fixation duration on all selected AOIs. The scarf plot makes it easy to compare the fixation behavior between different participants and it shows which AOIs attract the most attention. By utilizing the option of our tool to look at multiple panels simultaneously, it is possible to look at a bee swarm and a scarf plot at the same time for the same stimulus. When we choose the areas of interest in the bee swarm to be the start and end point (visualized by respectively the green hand and the dart board) the data can be further investigated when looking at the scarf plot. In Fig. 5, blue rectangles refer to the starting point of the stimulus in Fig. 7, while yellow areas refer to the end point. From this scarf plot, it can be derived that one cannot easily make assumptions about overall viewing behavior: when looking at a scarf plot for this one stimulus, attention patterns seem to appear random.

7 Data Processing and Storage

The visualization tool runs in a website with the following address: `https://www.p4ul.nl/dbl/`. This section describes all aspects related to the implementation of this tool and its features regarding the storage of personal data.

[1]Bee swarms are typically used in combination with dynamic stimuli, where they provide the extra dimensions of smooth pursuit [6]. Nonetheless, it is a valuable addition to our static-stimuli based tool.

7.1 Back End

The back end of the site is running on a dedicated server. The server is using an i7-4790k clocked at 4GHz, has 32GB DDR3 1333MHz RAM memory, and a 240GB SSD. The server using FastAPI [17], an accessible and one of the fastest web frameworks running on Python [1].

7.2 Authentication

The visualization tool gives the user the ability to create an account and login in order to have his or her data available everywhere. The authentication system is made with Firebase [11], which is easy to use and ensures that the user data is safe. When a user logs in for the first time, a call is made to the back end to create a new folder in the */authuser* folder of the server, with the ID of the user as the name. This folder stores the uploaded files of the user.

7.3 Data Storage

When a new user visits the site, a call is made to the back end to create a new user and subsequently a folder is created in the */users* folder on the server. This user ID is a universally unique identifier (UUID), which ensures that it cannot be easily guessed by brute-force. This user ID is stored in the browser's session storage. This ensures that all the data that is previously uploaded will still be available if a user accidentally closes and reopens the page, or refreshes it.

In order to make sure the storage of the server does not fill up, each temporary user folder is purged one day after its creation.

7.4 Session Sharing

The user has the ability to export their current session, when the user chooses to do this, an API call to the server is sent containing all the settings of the user. This data is then processed, and the data sets and images from the user are moved to a separate directory in the */export* folder. The name of this directory is a unique, random 8 character alphanumeric string. A new JSON file is also created in this directory with as name *settings.json*. This file stores the panel object of the users and all the settings from the panels.

When all the data is processed a response is sent to the user with the name of the directory. The site then displays the URL *https://dbl.p4ul.nl/share/(name of the directory)*. When a user visits this URL, another API call is made which loads all the data sets and images from the export folder into the user folder and copies the settings for the panels from the server to the user. The export folder is not removed and the URL can be reused.

7.5 Processing Input Files

When a file is uploaded to the server, it is converted to UTF-8 encoding to ensure that the D3-library is able to load the data correctly. If the user is logged in with the authentication system, the file is stored in the folder of the user in the */authuser* folder on the server. If the user is not logged in, the data is stored in the */users/userid* folder on the server.

When a `.zip` archive is uploaded, all files with the file extension *.csv* are extracted and moved into the user's data set folder. All files with file extension *.jpg* and *.png* are moved into the user's images folder. If a file being uploaded is marked public, the file is uploaded to the public folder, where users can view and load it from the catalog.

7.6 Data Generation

For testing purposes our site also features a data generation option. When a user wants to generate random data for an image, an API call is made to the back end, which generates random data for the image and stores it in the user's folder. The data generation is using Numba [16].

8 Discussion

Like any software product, our tool has several limitations in performance and features, and there are potentially useful extensions possible.

8.1 Scientific Value

Naturally, the visualization techniques presented in this tool do not include rigid statistical techniques and they cannot be used to draw general conclusions. Nonetheless, utilizing this tool can provide insight in large data sets and eventually produce hypotheses for further investigation.

8.2 Implementation Challenges

The React [23] framework manages the DOM (Document Object Model, the HTML structure) of the web-page, and employs optimized algorithms for rendering and updating parts of the page. The D3-library, on the other hand, relies on directly manipulating DOM-elements. This causes issues when React re-renders the elements edited by D3.

We experienced issues with making the effects of D3 visible, and binding buttons to functions that employ D3. Many trade-offs had to be made that introduced much additional complexity and thus harm readability and maintainability of the code base, such as using two distinct button mechanisms (one based on React, the other on D3). With hindsight, combining React and D3 is not worth the technical issues, or at least not with a carefully considered pattern.

8.3 Performance Limitations

Using larger data sets, especially data sets with many users, will increase the loading time of visualizations. The bee swarm, for example, is very sensitive for the number of users displayed, especially when the user-labels are displayed as well. When many users are selected, the animation runs noticeably slower. It should be noted that the theoretical run-time complexity of the algorithm deciding which points to display in the bee swarm is only $O(n)$ (where n is the number of data points in the data set). This is because each point can be rendered independently of the other points.

The scanpath visualization has to calculate the position and the radius of the fixation points to display. The algorithm used for this has run-time complexity of $O(n^2)$ where n is the number of data points in the data set.

9 Conclusion and Future Work

We present a web-accessible visualization tool for eye tracking data, that offers four different visualization techniques. These techniques are a 2D density plot, a scanpath (or gaze plot), a bee swarm with animation and a scarf plot. The main value of our tool is the set of available interaction techniques, and the interaction by means of selection synchronization between tools. Furthermore, we offer a simple platform where users can upload and store data sets, share visualizations and access public data sets.

Our tool currently only supports 2D density, gaze, bee swarm, and scarf plots, but many more visualization styles exist in relevant literature and can potentially be integrated into our tool. With more visualization styles, more opportunities of pattern or trend discovery in the data are provided.

Furthermore, our tool has extensive options regarding visualizations and their corresponding interactions. However, ease of use could be further investigated and optimized. Also, more options to reconfigure the visualization techniques can potentially be added.

Our tool has much potential for implementation optimization. Currently it is written in ill-compatible libraries. A more fundamentally designed framework with stable patterns for using libraries could provide major running speed improvements, and maintainability flexibility. Ideally a visualization tool allows to use user-made plug-ins, making the tool extensible with more interactions and visualizations.

References

[1] Techempower framework benchmarks, Jul 2019.

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

[3] M. Bostock. `https://github.com/d3/d3-contour`, August.

[4] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[5] M. Burch, A. Kull, and D. Weiskopf. AOI rivers for visualizing dynamic eye gaze frequencies. *Comput. Graph. Forum*, 32(3):281–290, 2013. doi: 10.1111/cgf.12115

[6] S. Dash and P. Thier. Chapter 6 - cerebellum-dependent motor learning: Lessons from adaptation of eye movements in primates. *Progress in Brain Research*, 210:121–155, 2014. doi: 10.1016/B978-0-444-63356-9.00006-6

[7] B. Farnsworth. 10 free eye tracking software programs [pros and cons], August 2019.

[8] J. H. Goldberg and J. Helfman. Visual scanpath representation. *6th Symposium on Eye-Tracking Research & Applications - ETRA 2010*, Jan 2010. doi: 10.1145/1743666.1743717

[9] Y. Holtz. 2d density plot. `https://www.data-to-viz.com/graph/density2d.html#`. Accessed: 08-06-2020.

[10] Y. Holtz. Basic contour chart. `https://www.d3-graph-gallery.com/graph/density2d_contour.html`, 2018. Accessed: 10-06-2020.

[11] A. L. James Tamplin. Firebase. `https://firebase.google.com/`, 2011.

[12] C. Maple. Geometric design and space planning using the marching squares and marching cube algorithms. In *2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings*, pp. 90–95, 2003.

[13] M. W. Matlin and T. A. Farmer. *Cognition*. Wiley Custom, 9 ed., 2017.

[14] 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 and Computation*, 17(1-2):39–64, 2017. doi: 10.1080/13875868.2016.1226839

[15] H. Nguyen. Material-ui: A popular react ui framework. `https://material-ui.com/`, 2013.

[16] T. Oliphant. Numba: A high performance python compiler. `http://numba.pydata.org/`, 2012.

[17] S. Ramírez. Fastapi. `https://fastapi.tiangolo.com/`, 2019.

[18] E. Robertson and J. Gallant. Eye tracking reveals subtle spoken sentence comprehension problems in children with dyslexia. *Lingua*, 228(102708), 2019. doi: 10.1016/j.lingua.2019.06.009

[19] R. Rosenholtz, Y. Li, J. Mansfield, and Z. Jin. Feature congestion: A measure of display clutter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, p. 761–770. Association for Computing Machinery, New York, NY, USA, 2005. doi: 10.1145/1054972.1055078

[20] D. D. Salvucci and J. H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. *Proceedings of the Eye Tracking Research and Applications Symposium 2000*, pp. 71–78, 2000. doi: 10.1145/355017.355028

[21] M. Tanenhaus. *Eye Movements, A Window on Mind and Brain*, chap. 20, pp. 443–469. Elsevier Science, 2007. doi: 10.1016/B978-008044980-7/50022-7

[22] N. J. Wade. *Eye Movements, A Window on Mind and Brain*, chap. 2, pp. 31–63. Elsevier Science, 2007. doi: 10.1016/B978-0-08-044980-7.X5000-9

[23] J. Walke. React - a javascript library for building user interfaces. `https://reactjs.org/`, 2013.

[24] J. S. Yi, Y. A. Kang, J. T. Stasko, and J. A. 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

[25] W. ying Sylvia Chou, N. Trivedi, E. Peterson, A. Gaysynsky, M. Krakow, and E. Vraga. How do social media users process cancer prevention messages on facebook? an eye-tracking study. *Patient Education and Counseling*, 2020. doi: 10.1016/j.pec.2020.01.013

[26] O. Špakov and D. Miniotas. Visualization of eye gaze data using heat maps. *ELEKTRONIKA IR ELEKTROTECHNIKA MEDICINE TECHNOLOGY T*, 115, 01 2007.

A Front-End Class Diagram

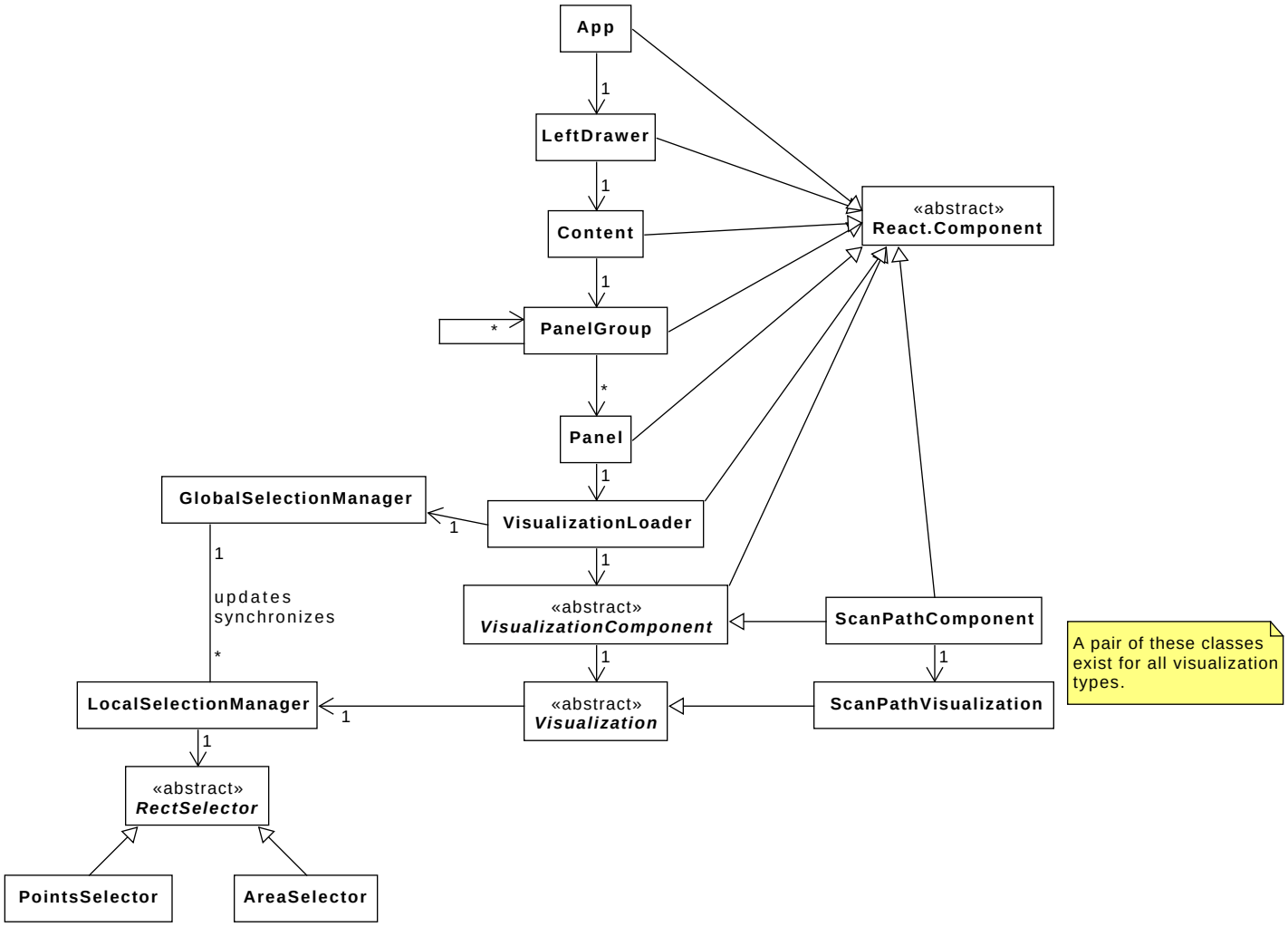

Fig. 10: Simplified class diagram of the website's front-end. Many visualized components implement a Component of the React-framework, but the classes that use features of the D3.js library are constructed via usual JavaScript class-like constructors. Each visualization is framed in a `Panel`. The `LocalSelectionManager`, of which one exists per visualization, uses the *State* pattern to switch between point and AOI selection. A single instance of the `GlobalSelectionManager` is used to synchronize active selections between visualizations, and receives updates from `LocalSelectionManager`s using the *Observer* pattern. `VisualizationComponent` and `Visualization` are abstract base classes, providing hook methods for drawing and scaling, that are implemented differently for each visualization.