Heatmap Explorer: an interactive gaze data visualization tool for the evaluation of computer interfaces

Antonio Diaz Tula[*]
diaztula@ime.usp.br

Andrew Kurauchi[*]
kurauchi@ime.usp.br

Flávio Coutinho[†]
flcoutinho@usp.br

Carlos Morimoto[*]
hitoshi@ime.usp.br

ABSTRACT

Eye gaze is an important source of information to evaluate computer interfaces. Typically, visualization of gaze data is performed using heatmaps and gaze scanpaths displayed on top of images of the interface, enhancing regions that have attracted the user's visual attention. Such tools work well for static interfaces but they are not appropriate to visualize dynamic interfaces where the object of interaction is always changing, such as games, web browsing, or even common applications that change the interface according to the status of the application. In this paper we introduce an interactive tool to explore the spatial-temporal distribution of visual attention called Heatmap Explorer (HME). HME allows the experimenter to control the visualization by selecting temporal intervals and adjusting filter parameters of the eye movement classification algorithm. We show results of three typical application scenarios and discuss how HME can be an effective usability evaluation tool.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords

Heatmap Explorer; eye gaze visualization; user interface evaluation; information visualization.

INTRODUCTION

An eye tracker is a device that measures eye movements in order to determine the point-of-gaze of an individual [6, 12]. Most of the eye trackers used today employ one (or more) camera(s) to capture images of one's eye(s). Computer vision techniques are then applied to detect and track eye features that reflect eye orientation (such as the pupil or iris center) and the point-of-gaze is computed by a mapping function such as a polynomial. The coefficients of the polynomial can be computed using regression from data collected during a calibration procedure, where the user must look at a set of known targets. Due to anatomical characteristics of the human eye, there exist a direct relationship between the orientation of the eyeball and what is being observed. Thus, the tracking of any eye feature related to eye orientation allows the estimation of the point-of-gaze.

Figure 1: Left image: a remote eye tracker in general placed below the computer screen and; Right image: a head-mounted eye tracker.

An eye tracker device is usually found in either of two types of configuration: *remote* or *head mounted*. Figure 1 shows commercial eye trackers available for each of these configurations. In a remote eye tracking system the camera is typically placed below the computer monitor and the point-of-gaze is computed relative to the computer screen. Remote eye trackers have the advantage of not requiring any kind of equipment to be worn or be in direct contact with user.

In head-mounted configurations the camera is normally attached to some kind of glasses (or helmet) that must be worn by the user, remaining fixed relative to the user's head, and the user is allowed to move freely around the environment. Head mounted eye trackers require a second camera pointed towards the environment (called scene camera), also fixed to the apparatus worn by the user. This second camera is responsible for capturing scene images that covers the user's field of view, and the computed point-of-gaze is estimated relative to these scene images. Despite being more invasive when compared to remote configurations, head mounted eye trackers are portable, can be used in analysis of daily activity, and estimation of the point-of-gaze is not restricted to a single plane (computer screen).

Because there is a strong correlation between the point-of-gaze and the focus of attention of a person, gaze data can be useful

[*]Institute of Mathematics and Statistics - University of São Paulo
[†]School of Arts, Sciences and Humanities - University of São Paulo

in a wide range of applications. Duchowski [2] classifies eye tracking applications into two major categories: *interactive* and *diagnostic*. Interactive applications use eye trackers as alternative input devices and react according to what a user observes. Diagnostic applications use data as evidence of the visual and attentional behavior of an individual to typically perform some kind of analysis that correlates gaze behavior to the execution of a task.

A virtual keyboard, drawn on the computer screen, in which characters are typed as they are gazed by the user is an example of an interactive application [11]. It also illustrates the potential of eye tracking as assistive technology to users that are not able to use conventional input devices. The main challenge in the development of gaze controlled interfaces, specially if the goal is to use the gaze as the only input modality, is to properly distinguish the actual intention of the user's gaze: control or observation [8]. Different interaction paradigms for gaze-based interfaces have been developed and proposed to deal with this challenge [2]. Another significant challenge is dealing with the relatively low accuracy of eye trackers (when compared to traditional pointing devices).

Studies and experiments related to the fields of neuroscience and psychology are examples of diagnostic applications. Usability evaluation of user interfaces supported by gaze data is a concrete example of diagnostic application that is directly related to the Human Computer Interaction (HCI) field. The design of interactive systems requires many iterations of designing, prototyping, and evaluation of the design before the system is actually implemented [14], and support of gaze data during evaluation can speed up this process. Information about the user's visual attention during an evaluation experiment is valuable in determining regions of interest, interface components that are hard to see or find, understanding the user's mental model when performing a task, etc. Therefore, eye tracking data can be used to complement the information collected during traditional user experiments in usability evaluation research.

From the brief description of the two application categories, it is possible to observe that eye tracking is strongly related to HCI either as a novel and not so explored input modality or as a useful tool to analyze user behavior.

The focus of this paper is on gaze data visualization tools for the evaluation of computer interfaces. Blascheck et al. [1] present a recent survey on state-of-the-art visualization techniques for eye tracking data and classify them into nine categories based on properties of the data, including aspects of the stimuli and the viewer, and on properties of the visualization technique, such as 2D or 3D scenes. For evaluating interfaces, heatmaps and scanpaths have been shown to be very useful in discovering areas of visual interest [15]. Such techniques are mostly suitable for the visualization of a static scene or photograph.

In this paper we introduce Heatmap Explorer (HME), an interactive visualization tool of remote and head-mounted eye tracking data that is suitable for the evaluation of computer interfaces. Before introduction of our HME application, we present in the next section a brief review regarding the use of eye tracking in usability evaluation and visualization of gaze data in particular. Following it, we introduce our HME application and then describe experimental results using three application scenarios where HME could be used to investigate usability issues.

USABILITY EVALUATION AND VISUALIZATION

Before presentation of related works, let us introduce some important concepts related to eye movements. In terms of how gaze data can be interpreted in the context of eye movement analysis, a *fixation* consists in a set of point-of-gaze samples that fall within a certain region during a continuous time frame. In other words, fixations correspond to the moments when the eyeball is mostly static so that an object of interest can be imaged by the retina [3]. A *saccade* is a fast eye movement that "jumps" from one fixation region to the next, while a *scanpath* is defined by a series fixations and the saccades that connect each pair of consecutive fixations. Analysis of scanpaths can be useful to reveal information about a user's behavior. For instance, during a search task, the optimum scanpath would be a straight line towards the desired target and the difference between the optimum and observed scanpaths can be used as an indicator of the quality of interface design [5]. Note that raw gaze data (sequence of all point-of-gaze samples) must be processed in order to obtain scanpath information.

According to Blascheck et al. [1], two common approaches to analyse gaze data include statistical analysis of the data or the use of visualization techniques. While statistical approaches can be used to obtain quantitative results, visualization allow analysis in a more exploratory and qualitative way.

A statistical approach typically uses metrics related to eye movements in order to characterize the user's behavior. Goldberg and Kotval [5], for example, proposes several scanpath related metrics that can be associated with the effectiveness of users' search patterns during interaction. Some of proposed metrics include: scanpath length and duration, convex hull area of the scanpath, spatial density of area covered by a scanpath, and transition matrix between areas of interest (AOIs) over the interface. Raiha et al. [13] argues, however, that before some metric can be defined or chosen in order to statistically analyse gaze data, visualization can be important to understand the characteristics of gaze behavior in a given context. Hence, the importance of visualization techniques and tools. Presentation and discussion of about several visualization techniques can be found in the survey by Blascheck et al. [1] and also in [13], [4], and [15].

The simplest possible form of visualization of eye tracking data would be to draw the sequence of estimated gaze points directly over the observed stimulus. Although suitable for either static or dynamic stimulus, individual point-of-gaze visualization is not very useful for analysis as little information is given and changes at a fast rate. Therefore, visualization techniques that summarize raw gaze data into more meaningful information are preferred.

Scanpaths and heatmaps are probably the most commonly used styles of gaze data visualization. In a scanpath visualiza-

tion fixations are normally represented by circles and saccades as lines connecting the fixation circles. The circles and lines are drawn over the stimulus used during gaze data acquisition. Some properties of the circle such as radius, color or opacity may be changed to indicate the duration of the fixation. Despite being useful to illustrate viewing order of the fixation regions, scanpath visualizations can become confusing if there is a lot of overlapping between several fixations and saccades, a problem that is further amplified if scanpaths from multiple users are displayed.

Heatmaps, also known as attention maps, discard the time dimension, displaying an aggregation of fixations. Regions over which lots of fixations occur are usually displayed in hot colors while regions with little or no fixations are displayed in cold colors, but other schemes to discriminate the intensity of fixations may be applied as well. Although temporal information is lost, heatmaps are useful to identify regions that attract more attention. Heatmaps is also a convenient visualization technique if data from multiple users need to be displayed.

A disadvantage of both scanpath and heatmap visualization is that they are not suitable for dynamic stimuli, such as videos or systems whose interface changes as users interact with them. Regions that draw more user attention at a given time frame, resulting in a set of fixations, may change substantially over time, rendering the previously detected fixations obsolete with respect with the current state of the stimulus. Kurzhals et al. [10] overcome this limitations by suggesting motion compensated heatmaps as well as a space-time cube visualization.

In addition to the challenges involving visualization over dynamic stimuli, Blascheck et al. [1] also point out a general lack of visualization techniques and tools that allow interactive analysis of eye tracking data.

HEATMAP EXPLORER

Heatmap Explorer, HME for short, was developed to visualize user's gaze behavior on dynamic interfaces, helping interaction designers to quickly identify design issues. Instead of using a static image of the interface, HME uses a video recording of the interface synchronized with gaze data to present a spatial-temporal representation of the user's visual attention during the interaction. It should be noted that a static interface can be understood as a single frame repeated throughout the whole interaction, thus HME can also be applied to such interfaces.

In a dynamic interface the visual element focused by the user's gaze may not be available anymore after some time. For example, an icon representing that some data is being loaded will disappear as the data is loaded. The information that the user was looking at that specific icon will be out of context in the new visual state of the interface. For this reason, HME only displays gaze information within a temporal window before the current time. The ideal size of the temporal window depends on the application: highly dynamic interfaces, such as game interfaces, may require a short temporal window, while slowly changing interfaces might use longer temporal windows. For this reason, the length of the temporal window is defined by the user.

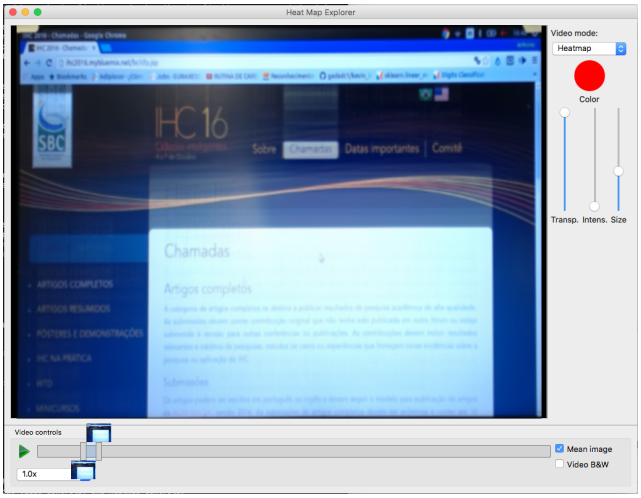


Figure 2: Example of mean image representing the interface being scrolled down.

The gaze information within the temporal window is displayed and updated as the video is played. The designer can pause the video at any time to carefully inspect the user's visual attention during that particular moment of the interaction. Besides showing gaze data on the video and on a static frame of the video, HME also has the option of showing a **mean image**. The mean image is a frame composed by the average pixel for each position for a subset of frames from the temporal window. The resulting image is a way of representing motion in the interface with a static image. A subset of frames is used instead of all the frames in the temporal window to avoid getting the mean image to be too blurry, also the computation of the mean image is sped up by doing so. An example of the user scrolling down a page is shown in Figure 2. With the combination of the mean image with the gaze data the designer can make sense of the user's spatial-temporal visual attention. For example, the user following the mouse pointer with their gaze can be represented in a single static image.

Advantages and challenges of head-mounted eye tracker data

A head-mounted eye tracker can be used to collect both the eye gaze data and the video of the interface. It not only facilitates synchronization, but also allows for a wider range of interfaces and devices to be tested with the same method. Interaction with large displays, computer monitors, tablets or smartphones could be analyzed in the same way. A caveat, considering the current accuracy of eye trackers, the study on small screens may not yield good results. The magnitude of the gaze estimation error may be too large compared to the screen size. The results for small screens may be more reliable as eye tracking technology develops further.

Using the scene camera to record the interaction gives the designer more flexibility to test interfaces in a wider range of devices. This flexibility, however, comes with a price. As the scene camera is mounted on the head of the user, the screen will be moving in the recorded video. Even if the user wearing

the head-mounted eye tracker tries to keep their head still, some involuntary head motion will inevitably occur. Also, the scene camera often captures more than just the interface. The surroundings of the surface of interest (such as a computer monitor) may be an unnecessary visual clutter when exploring the user interaction data.

Chin rests have been employed with remote eye trackers to help the user to keep their heads still. However this does not solve the problem of surrounding visual clutter. Also, it may cause discomfort to the user and affect their interaction with the interface.

HME solves both problems by stabilizing the interface image with computer vision algorithms. The interface is initially detected and extracted from the image. The extracted interface is then mapped to a frontal view. This process is repeated for all frames in the scene video. The result is a video containing only the interface as viewed from the front, independently of the position of the user's head during recording.

The interface of HME is shown in Figure 2. It is composed of 3 main parts: the video window, video controls, and heatmap controls.

The Video Window

The interface video overlaid with the heatmap is displayed by the video window. After selecting the directory containing the interface video and the gaze data, the interaction designer can visually inspect the user's gaze data in this window. The video window updates the visualization in real time as the designer adjusts the parameters in the video and heatmap controls. This may facilitate fine tuning the parameters to highlight the desired information.

Video Controls

The video controls are shown at the bottom of the interface (Figure 2). The available controls are the play/pause button, the progress bar, the speed selector, and two checkboxes (on the right) for toggling on/off the black & white and mean image options.

The play/pause buttons are the standard controls present in any modern video player. The progress bar is composed by a large rectangle that represents the overall video duration and two slider controls. The rightmost slider shows the current video frame, similar to most video players. The leftmost slider is used to define the size of the temporal window. This temporal window determines the gaze samples that are displayed in the current video frame. The size of the temporal window is proportional to the distance between the two sliders. The farther the distances between the sliders, the larger the number of gaze samples displayed in the current video frame. This temporal window also determines the video frames used to compute the mean image. A small thumbnail is shown close to both sliders, to give feedback about the initial and final frames of the temporal window (Figure 2).

The speed selector determines the playback velocity. The speed value 1.0x is the recorded playback velocity, which is typically 30 frames per second. The video playback velocity

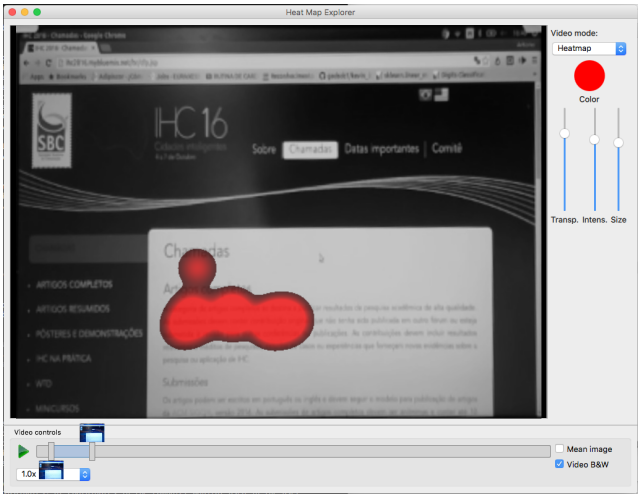


Figure 3: The interface can be displayed in black and white to facilitate the visualization of the heatmap information.

is slower for values smaller than 1.0x and faster for values greater than 1.0x.

Visualizing the interface in black and white may be useful to see the heatmap information more clearly (Figure 3). This option can be set by selecting the black & white checkbox. If the mean image checkbox is selected, HME will display the mean image of the temporal window, which, as explained above, can be used to visualize motion in the interface.

Heatmap controls

The Heatmap controls are composed of a combobox to select the visualization mode, a color picker to select the color of the heatmap visualization mode, and three slidebars to control different parameters of the visualization, that will be described next.

Data collection

The data required by HME is composed of the video from the scene camera and the user's gaze data acquired by a head-mounted eye tracker. On the interface the interaction designer can select the directory containing such data. HME then loads the video and the estimated gaze data. In case the gaze information is not synchronized with the video frames, HME synchronizes the two data streams.

In this version of HME, data collection is done with the Pupil Eye Tracking software (`https://github.com/pupil-labs/pupil` [9]). The Pupil software has the option to calibrate the eye tracker and also to define surfaces, such as the computer monitor, in the scene camera with fiducial markers. While running, the software estimates the user's gaze on the scene image and, if any surface is defined, it estimates also the gaze on that surface. The gaze samples are composed by the coordinates of the gaze position on the scene image and a timestamp to localize it temporally. The recording option of the pupil software creates a folder with several files, containing

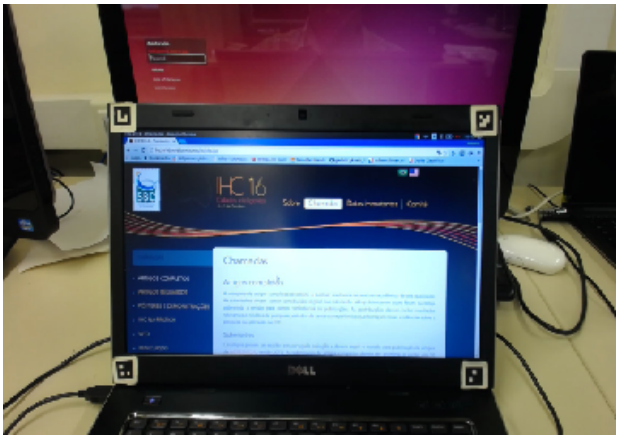
information about the user calibration, gaze estimation on the scene image and on the surfaces that were defined. It also records information about the position of the fiducial markers in the scene image and the homographies [7] that maps coordinates from the scene image to the surface and vice-versa.

The steps to make a recording using the Pupil software, that can be used later with HME, are the following:

1. Place the fiducial markers in the corners of the screen where the application under evaluation will be running. It can be a computer screen, a tablet or even a smartphone, among others. Those markers will facilitate to robustly detect the screen with low computational cost.
2. Connect the eye tracker cameras to the computer and run the Pupil software.
3. Adjust the scene camera to cover the entire computer or mobile screen where the application under evaluation will run. Adjust also the eye camera so that the user's pupil is completely visible.
4. Run the calibration routine of the eye tracker. In case the computer running the Pupil software is the same running the interface under evaluation, the calibration will show several markers on the monitor and the user must look at those markers. If the interface is running in a different device, the Pupil software has additional options to perform calibration, not described here (but can be found in the Pupil documentation).
5. After calibration, the user should look at the fiducial markers to verify that the Pupil software correctly estimates the gaze over the observed marker in the scene image.
6. Press the "Record" button in the Pupil software, and then proceed to the interface under evaluation to begin the user task, while data is being recorded in a folder. This folder can be configured in the Pupil interface.

When HME opens a recording folder, it first loads the gaze information and synchronizes it with the video frames. A gaze sample provided by the eye tracker is composed of a time stamp t, the gaze coordinates (x, y) that correspond to a coordinate on the screen (determined by the markers), and a confidence measure c, which indicates how reliable the gaze data is. For example, c can be zero when some eye feature, such as the pupil, is not detected during an eye blink.

The video frames of the scene camera contain the application screen, but also some other visual, background content, as shown in Figure 4a. This background is of no interest to the evaluator, so it must be removed. This is accomplished by detecting the four fiducial markers in each video frame, and computing a homography transformation [7] to map the polygon defined by the markers into the undistorted view of the computer screen. The resulting image is displayed in the HME's video window. The image after detecting the screen and applying the homography transformation is shown in Figure 4b. The gaze data on the undistorted image is obtained by applying the same homography transformation to the gaze points.



(a) Scene image captured with the scene camera. Observe the fiducial markers placed in the screen corners.

(b) Screen image after the homography transformation. Note that all background has been removed and the screen image is undistorted.

Figure 4: Computer screen with fiducial markers. The markers are used to track the screen in the head mounted eye tracker scene camera.

Implementation details

In this section we provide some implementation details about the different visualization modes. Visualization modes were implemented modularly, so new visualizations can be easily added without modifying the implementation of HME.

Each video frame has a corresponding gaze point, that was computed while loading the recording folder. At any given time, HME has a current video frame and a temporal window size. If the current frame is the i-th video frame and the window size is k, then HME will consider the gaze information from the $(i-k)$-th to the i-th frame to compute the visualization. When the user adjusts the temporal window size by making it larger or smaller, a greater or fewer number of gaze samples will be used. The larger the window, the older the gaze samples will be displayed in the current video frame.

Thus the visualization modes receive a video frame (the i-th frame and a list of gaze samples) and returns a video frame overlaid with the gaze visualization. When the user selects the "mean image" checkbox (Figure 2), HME computes the mean image with a uniformly distributed subset of the frames in the

temporal window, and passes this mean frame together with the gaze information to the visualization mode.

Gaze points are displayed as a sum of 2D Gaussians. Each Gaussian is centered at the (x, y) coordinate of a gaze point, and has a given σ value that can be configured. The σ value determines the shape of the Gaussian: small values will result is a smaller gaze point area visualization, whereas large values will result in a larger visualization area. Hence, the first step is to compute the sum of several Gaussians centered at the gaze samples. It could be interesting for the designer to distinguish the more recent gaze samples from the older ones, to have a clearer idea about the user gaze behavior. Hence, each Gaussian is weighted according to its temporal position in the window. We used a power distribution to weight the Gaussians, so those corresponding to more recent gaze samples are given a higher coefficient than older gaze samples. Figure 5a shows an example of the Gaussians for 120 gaze samples. The Gaussian coefficients are weighted by the power curve shown in Figure 5b and clipped to the interval [0, 1].

The next step is to compute the blending layer. This layer will be blended with the original frame using a α blending coefficient. The user can configure the value of α to control the blending: larger values of α will emphasize more the original image, whereas smaller values will emphasize the gaze visualization. We defined 3 different visualization modes:

1. **Blurring**: in this mode the entire image is blurred using a Gaussian filter. The size of the blur can be configured in the HME interface. The area observed by the user is shown without any blur, as shown in Figure 6a.

2. **Fogging**: in this mode the original image is covered by a snow or shadow. This effect is controlled by a parameter that can be configured in the HME interface: higher values produce a snow effect, whilst smaller values produce a shadow effect. The area observed by the user, defined by the Gaussian matrix, is shown without any fog or shadow effect (as shown in Figure 6b).

3. **Heatmap**: this mode shows the original image, with the area observed by the user filled with a given color. An example of this visualization is shown in Figure 6c. The user can change the color used to display the gaze information, and also to control the transparency of both the original image and the heatmap, and also the blending coefficient between the two layers (i.e. emphasize more the heatmap or the information behind the heatmap).

EXPERIMENTAL RESULTS

To exemplify how HME can be useful for the evaluation of user interfaces, in this section we show results from a short experiment using three different application scenarios: game playing, web browsing, and a desktop application.

The temporal dimension is represented by the video playback. As we can only present static images of the interface in the figures the temporal dimension is not represented. To interpret how the gaze of the user behaved on the interface it would be necessary to watch the video playing and not only the static images shown in this paper. Considering this limitation in the report, we present the results.

Evaluation of Graphical User Interfaces

In a typical user experiment to evaluate a Graphical User Interface (GUI), the user is given a task to be performed. The evaluators try to infer the user's mental model, which parts of the interface are easy to use and which are not, by observing their behavior.

To demonstrate how HME can help the evaluation of GUIs, we will use the OpenOffice suite, in particular, the creation of a presentation using OpenOffice Impress.

The task given to the user was to create a slide with a title in bold. Figure 7 shows the visualization of the interaction using blurring, fogging, and heatmap. During the time window shown in the figure the user was trying to change the title style to bold.

By observing the behavior of the user's visual focus we can learn if the user's mental model is correct, i.e., if s/he is looking at the correct regions and selecting the correct actions required to accomplish the given task. In this example, the designer can evaluate if the user is able to change the title to bold. By observing the visualization window shown by HME in Figure 7, it can be inferred that the user fixated at the beginning and at the end of the text (likely to confirm that it was correctly selected), and then looked at the tool bar to find the bold option. The larger fixation area corresponds precisely to the area where the bold option is located in the tool bar. Hence the user found the bold button without the need to visually search another parts of the interface.

This simple example shows how HME can be an effective tool to evaluate GUIs, helping developers to quickly identify problems with the graphical design.

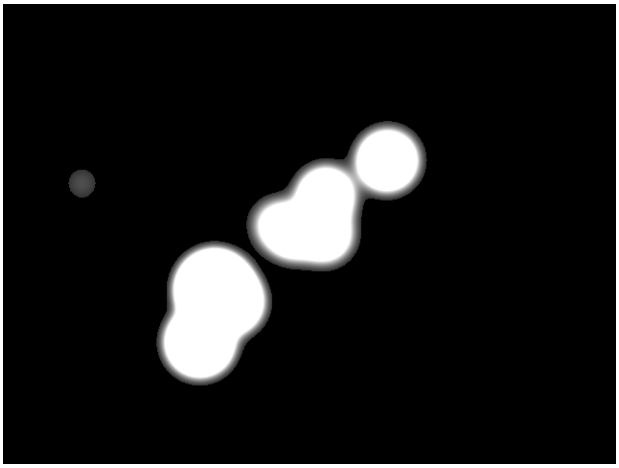
Web Browsing

Web pages can present very different behaviors, from simple, mostly static content pages such as Wikipedia, to interactive forms such as Google Docs and Facebook, and very interactive and dynamic online games. Different than just evaluating a particular GUI, the design of web pages might require its content to be available in different platforms, such as mobile phones, tablets, and desktop computers.

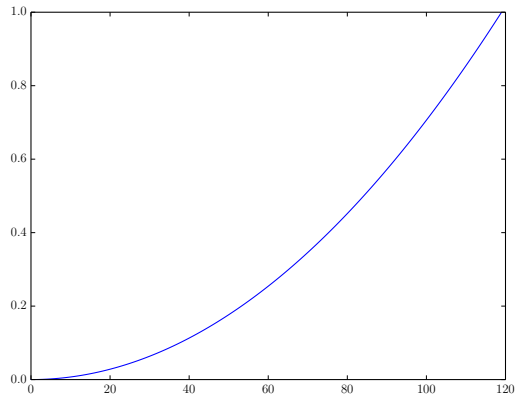
Because HME is able to process gaze data from remote and head-mounted eye trackers, the user performance when browsing a web page using different platforms can be analyzed using the same visualization tool.

Figure 6 shows the visualization of gaze data of a user when browsing the web page of the *XV Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais - IHC'16*, using blurring, fogging, and heatmap. During the time window shown in the figure the user was skimming through the main text area of the call for papers. Observe that it is clear that the user is just browsing the titles and not reading the text.

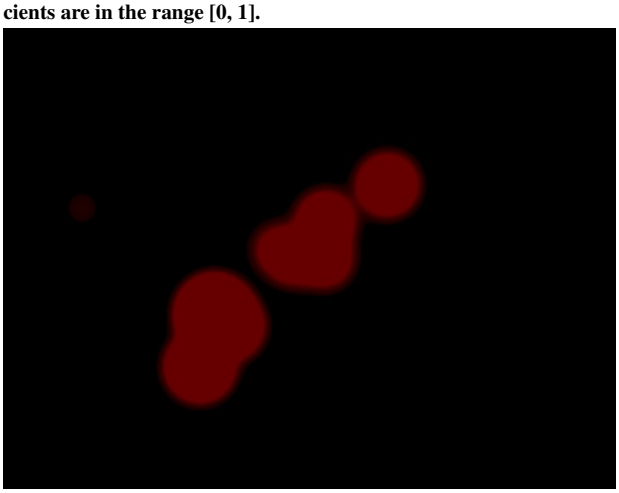
By observing the behavior of the user's visual focus we can learn about the parts of the interface, in different platforms, that mostly attract the user's attention. The lack of attention in



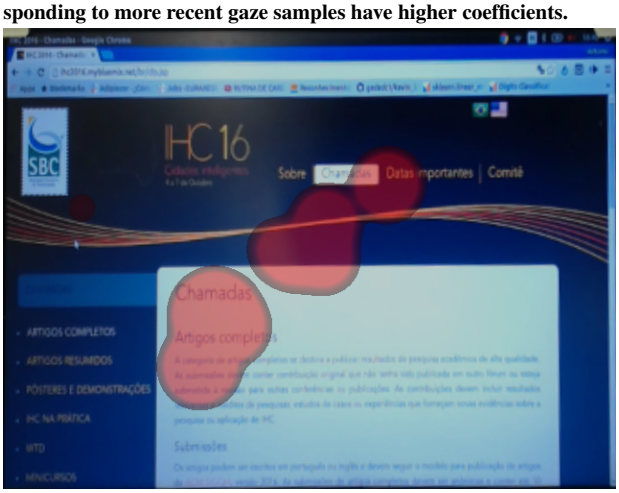
(a) Sum of Gaussians corresponding to 120 gaze samples. Coefficients are in the range [0, 1].



(b) Power curve used to weight the Gaussians, so Gaussians corresponding to more recent gaze samples have higher coefficients.



(c) Heatmap layer computed from the Gaussians matrix.



(d) Result of blending the screen image with the blending layer using a α coefficient for the blending process.

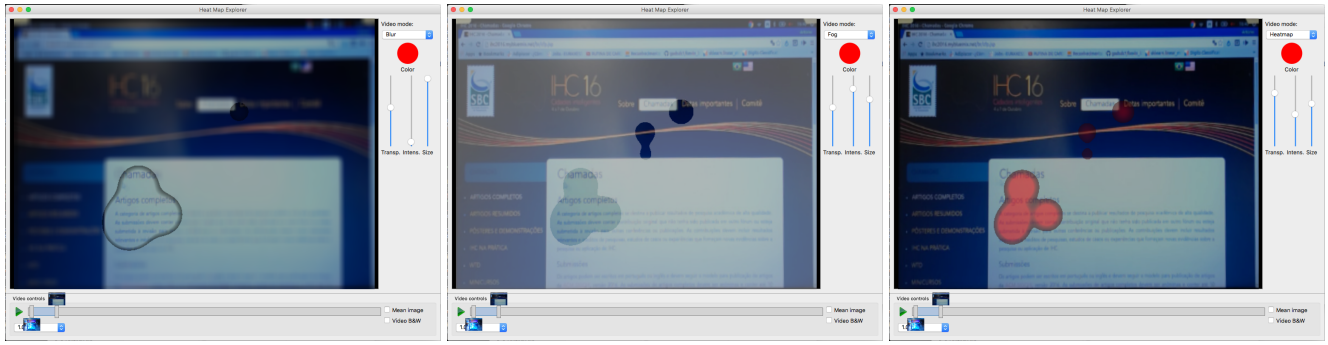
Figure 5: Implementation details for the visualization modes.

certain parts can indicate lack of interest or problems in the visual design of the web page.

Game Playing

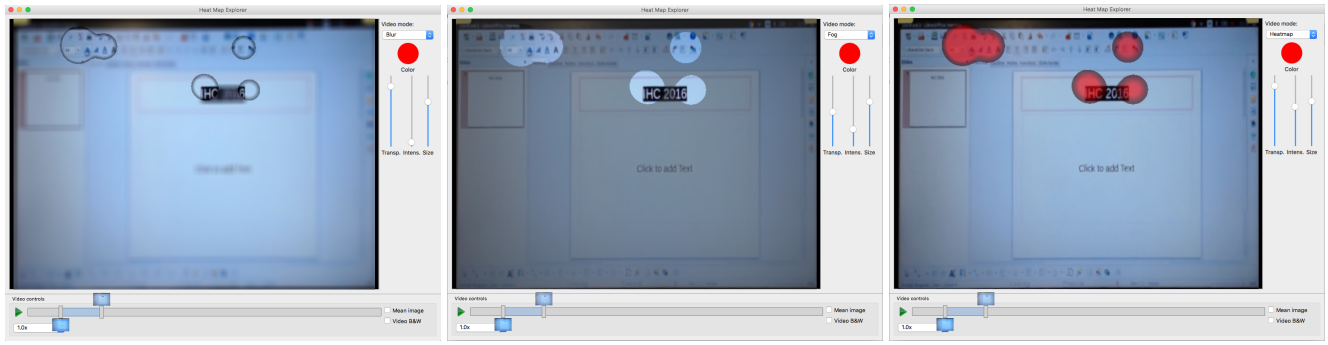
The observation of the user's visual focus of attention in games is particularly challenging due to the multitude of possible platforms but also due to the dynamics of the interface required by the game, such as real-time control of virtual characters or an evolving board game. To demonstrate the use of HME for evaluating user performance during game playing we will show results of a user experiment for the game Minesweeper. Minesweeper is a classic puzzle game that became very popular because it was once distributed as part of the Microsoft Windows operating system. The objective of the game is to clear a rectangular grid where each position might contain a hidden mine, without detonating any mine. Initially, the contents of all grid positions are unknown to the player. The player must select one at random, and then the computer reveals the content of the selected grid position. If a mine is revealed, the player loses the game. Otherwise, a digit corresponding to the number of mines around that position is revealed and adjacent positions with zero mines surrounding them are automatically revealed.

Figure 8 shows the visualization using blurring, fogging, and heatmap. The difference in the shapes of each visualization is due to the selection of slightly different time instants and temporal windows. By observing the behavior of the user's visual focus we can learn about the user's strategy and explain possible errors. For example, Figure 8 shows that the user looks at the edges of the board with unknown content and is particularly interested in grid positions with high numbers, the grid with value '3' in this case. The visual focus on the positions with high value is an indication of high cognitive load due to the importance of the value to avoid the selection of an adjacent grid position with a mine. Therefore, gaze data can be used to identify regions of interest and that demanded higher cognitive load. When the user selects a position adjacent to high values without displaying the required cognitive load,



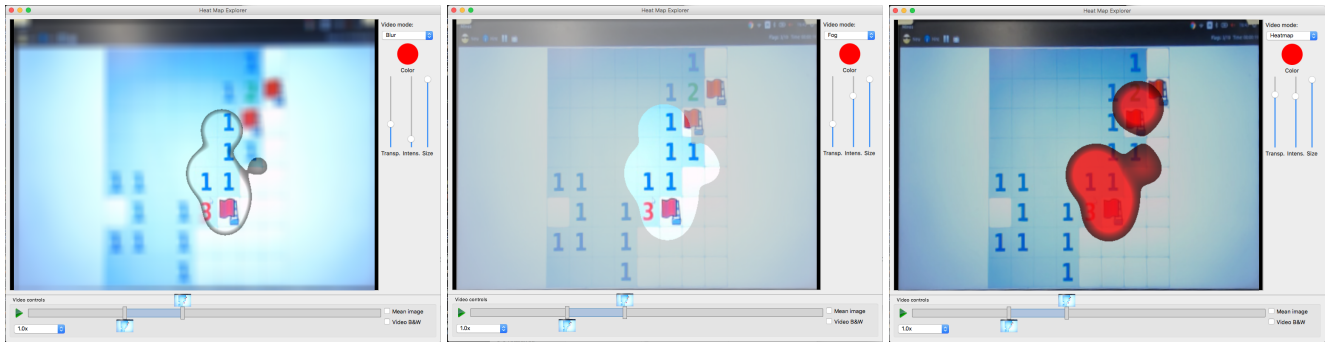
(a) Blurring (b) Fogging (c) Heatmap

Figure 6: Visualization modes available in HME. Experimental results showing the gaze behavior when browsing a web page.



(a) Blurring (b) Fogging (c) Heatmap

Figure 7: Experimental results showing the user's gaze behavior during an experiment to evaluate the usability of a graphical user interface.



(a) Blurring (b) Fogging (c) Heatmap

Figure 8: Experimental results showing the gaze behavior during a Minesweeping game.

this might be an indication of the level of expertise of the user or the level of interest or attention of the user.

CONCLUSION

In this paper we have introduced Heatmap Explorer (HME), an interactive visualization tool for off-line evaluation and exploration of user visual behavior during the use of computer interfaces. Gaze information recorded during an user experiment using a remote or head-mounted eye tracker is loaded into the HME, that combines the video of the experiment with the gaze data and provides 4 different visualization modes: video with heatmaps, user visual focus with fog background, user visual focus with dark background, and user visual focus with blurred background. Most applications available for visualization of gaze data are limited to static images. HME allows the user to define temporal regions where relevant interface events can be better analysed. HME also allows the user to visualize the gaze data over a mean image computed by sub-

sampling the video frames contained in the temporal region. This mode allows the user to visualize temporal information on a static mean image, for example, to check if the user's visual focus was over a moving object. We have presented experimental results using three application scenarios to show the effectiveness of HME: game playing, web browsing, and graphical user interfaces. The main contributions of our paper are: the design and implementation of HME, the development of a video stabilization mode to be used with data collected with head-mounted eye trackers, the visualization of gaze data on an temporally averaged image, and the definition of application scenarios where HME can be used to improve the analysis of user experiments.

A possible extension of this work would be showing multiple surfaces. For example, if the user has access to a printed help material the gaze information could be shown in both the screen (the main surface) and in the printed material (the secondary surface). Another possible extension is to allow the creation of tags in the video. Such tags could be used both to more easily identify key frames in the video and to set different temporal window lengths depending on how dynamic the interface is at that moment.

HME will be available as an open source software after the publication of this paper.

Acknoledgements

The authors would like to thank São Paulo Research Foundation (FAPESP), grants 2011/00267-1 and 2013/06791-0, for the financial support.

REFERENCES

1. T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl. 2014. State-of-the-art of visualization for eye tracking data. In *Proceedings of EuroVis*, Vol. 2014.
2. Andrew Duchowski. 2002. A breadth-first survey of eye-tracking applications. *Behavior research methods, instruments, computers* 34, 4 (2002), 455.
3. Andrew T. Duchowski. 2007. *Eye Tracking Methodology: Theory and Practice*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
4. Andrew T. Duchowski, Margaux M. Price, Miriah Meyer, and Pilar Orero. 2012. Aggregate Gaze Visualization with Real-time Heatmaps. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. ACM, New York, NY, USA, 13–20. DOI: http://dx.doi.org/10.1145/2168556.2168558
5. Joseph H Goldberg and Xerxes P Kotval. 1999. Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics* 24, 6 (1999), 631 – 645. DOI: http://dx.doi.org/10.1016/S0169-8141(98)00068-7
6. Dan Witzner Hansen and Qiang Ji. 2010. In the Eye of the Beholder: A Survey of Models for Eyes and Gaze. *IEEE Trans. Pattern Anal. Mach. Intell.* 32, 3 (March 2010), 478–500. DOI:http://dx.doi.org/10.1109/TPAMI.2009.30
7. Richard Hartley and Andrew Zisserman. 2003. *Multiple View Geometry in Computer Vision* (2 ed.). Cambridge University Press, New York, NY, USA.
8. Robert J. K. Jacob. 1991. The Use of Eye Movements in Human-computer Interaction Techniques: What You Look at is What You Get. *ACM Trans. Inf. Syst.* 9, 2 (April 1991), 152–169. DOI: http://dx.doi.org/10.1145/123078.128728
9. Moritz Kassner, William Patera, and Andreas Bulling. 2014. Pupil: An Open Source Platform for Pervasive Eye Tracking and Mobile Gaze-based Interaction. (April 2014). http://arxiv.org/abs/1405.0006
10. K. Kurzhals and D. Weiskopf. 2013. Space-Time Visual Analytics of Eye-Tracking Data for Dynamic Stimuli. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (Dec 2013), 2129–2138. DOI: http://dx.doi.org/10.1109/TVCG.2013.194
11. Paivi Majaranta and Kari Raiha. 2002. Twenty years of eye typing: systems and design issues. In *Proceedings of Eye Tracking Research & Applications, ETRA 2002*. ACM Press, New Orleans, LA, 15–22.
12. C.H. Morimoto and M.R.M. Mimica. 2005. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding* 98, 1 (2005), 4–24.
13. Kari-Jouko Räihä, Anne Aula, Päivi Majaranta, Harri Rantala, and Kimmo Koivunen. 2005. *Static Visualization of Temporal Eye-Tracking Data*. Springer Berlin Heidelberg, Berlin, Heidelberg, 946–949. DOI: http://dx.doi.org/10.1007/11555261_76
14. Yvonne Rogers, Helen Sharp, and Jenny Preece. 2014. *Interaction Design: Beyond Human - Computer Interaction* (4th ed.). Wiley Publishing.
15. O Špakov and Darius Miniotas. 2015. Visualization of eye gaze data using heat maps. *Elektronika ir Elektrotechnika* 74, 2 (2015), 55–58.