Faculty 4: Computer Science   Institute for Web Science and Technologies   EYEVIDO GmbH

Intelligent Mapping of Eye-Tracking Gaze-Data on Fixed Web Page Elements

Master's Thesis

in partial fulfillment of the requirements for
the degree of Master of Science (M.Sc.)
in Web Science

submitted by
Hanadi Tamimi

First supervisor:	Prof. Dr. Steffen Staab Institute for Web Science and Technologies
Second supervisor:	Raphael Menges Institute for Web Science and Technologies
External supervisor:	Christoph Schaefer EYEVIDO GmbH

Koblenz, December 2017

Statement

I hereby certify that this thesis has been composed by me and is based on my own work, that I did not use any further resources than specified – in particular no references unmentioned in the reference section – and that I did not submit this thesis to another examination before. The paper submission is identical to the submitted electronic version.

	Yes	No
I agree to have this thesis published in the library.	☐	☐
I agree to have this thesis published on the Web.	☐	☐
The thesis text is available under a Creative Commons License (CC BY-SA 4.0).	☐	☐
The source code is available under a GNU General Public License (GPLv3).	☐	☐
The collected data is available under a Creative Commons License (CC BY-SA 4.0).	☐	☐

. . . .

(Place, Date) (Signature)

Zusammenfassung

Abstract

The output of eye tracking Web usability studies can be visualized to the analysts as screenshots of the Web pages with their gaze data. However, the screenshot visualizations are found to be corrupted whenever there are recorded fixations on fixed Web page elements on different scroll positions. The gaze data are not gathered on their fixated fixed elements; rather they are scattered on their recorded scroll positions. This problem has raised our attention to find an approach to link gaze data to their intended fixed elements and gather them in one position on the screenshot. The approach builds upon the concept of creating the screenshot during the recording session, where images of the viewport are captured on visited scroll positions and lastly stitched into one Web page screenshot. Additionally, the fixed elements in the Web page are identified and linked to their fixations. For the evaluation, we compared the interpretation of our enhanced screenshot against the video visualization, which overcomes the problem. The results revealed that both visualizations equally deliver accurate interpretations. However, interpreting the visualizations of eye tracking Web usability studies using the enhanced screenshots outperforms the video visualizations in terms of speed and it requires less temporal demands from the interpreters.

Contents

List of Figures

1. Introduction

Over time, Web design technologies have flourished to provide Web developers the freedom of design to build new generations of interactive websites for more authentic and engaging Web experiences. A satisfying Web experience comes from remarkable Web usability as the ISO defines *Web usability* to be concerned with the "effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments" [7], while the *Web experience* is concerned with "all aspects of the user's experience when interacting with the product, service, environment or facility" [9]. Therefore, it is crucial to understand the importance of measuring Web usability to assist in creating a better Web experience. Detecting the overt behavior of users, information about clicks, mouse moves, and scrolls, questionnaires, and "Thinking-aloud" methods are examples of traditional and successful strategies for examining the quality of usability.

Eye tracking technology has significantly improved in recent years. It plays an important role in the Human-computer interaction (HCI) field [17]. Eye tracking is used to assess search efficiency [14], online advertisements [11], navigation usability [15], overall design and other site components. Moreover, eye tracking provides analytical information about the user interaction between click events. Such information is shaped to deduce valuable knowledge concerning the significance of Web elements; i.e., which elements are the most eye-catching, which cause indecision and which elements are ignored. This knowledge is interpreted and then utilized as a modern strategy to investigate Web usability.

The interpretation of the eye tracking results on the Web can be achieved through analyzing the visualized output results. The results are represented with either a static screenshot of the Web page with users' fixations or a video representation of the Web interface from the user's perspective, where the Web page is shown with users' fixations in sequence. So far static representations cannot capture the dynamic transitions or event changes on the Web page. Inactive screenshots allocate less storage than the same session recorded as a video. Furthermore, static images demonstrate a full representation of the gazed Web page with fixations and time-stamped scan-paths, which implies less time for investigators to analyze user activity. Additionally, in static screenshots, it is possible to visualize data of multiple users on the same image to analyze the overall or individual user behavior on the Web page. On the other hand, video illustrations capture users' sessions with a detailed step by step user action, which reveal the current viewport with its Web elements and functionalities such as dynamic transitions and event changes. However, videos allocate more system storage, and they are recorded for each participant in the study, which implies that the investigator has to watch every user session to conclude the entire analysis of the study. Another time-consuming drawback of videos is that the investigator has to inspect the complete video to understand the user interaction during their session.

There are two methods for image creation in static screenshots; the first one is to capture a screenshot of the entire Web page when it is loaded, which will not only produce an image of the current viewport; rather the complete page content (Expanded-Viewport Screenshot), and the second method is to take multiple snapshots of the current viewport on either a timeout or a page event, such as window scrolling, and eventually combine all captured images in one single screenshot (Stitched-Viewport Screenshot). We argue that the latter method to be more robust than the first one for the reasons explained in the following paragraphs.

First of all, in the Stitched-Viewport Screenshot method, the images of the viewport are captured on page events, which indicates that the Web page is processed step by step within the recording session. The final stitched screenshot will guide the investigator to where the user lost interest, for example, stopped scrolling, without revealing the unseen parts of the Web page. On the other hand, the created image in the Expanded-Viewport Screenshot method will result in acquiring regions that are unseen by the user, which are of no importance to the investigator, as well as the trouble of producing extra stored pixels in the system.

Secondly, several different types of technologies are used in Web development and design. Some Web pages are static, which means that their components are available on page load, and others are dynamic in the sense that not every part of the Web page is visible when the page is loaded, and the Web page will only acquire the dynamically added parts on user interaction. For example; when the user clicks on something or endless scrolling as in infinite Web pages. In the dynamic case, the Stitched-Viewport Screenshot method wins the debate for maintaining the ability to capture the dynamically created elements on the page, since the images are taken on page scroll events. In contrast, the Expanded-Viewport Screenshot method might fail in obtaining the complete rendered Web page with the dynamic parts on page load, since they are not generated at the moment of page load.

Finally, Web elements are diverse in style. While some elements are always visible in the same position of the viewport on window scrolling such as fixed header menus and fixed footer, others are scrolled. The fact that these fixed positioned components are always visible in the viewport proposes that their gaze data on the screenshots will be equivalent to the scroll position of their appearances. This point has raised an issue in both argued strategies (Expanded-Viewport Screenshot and Stitched-Viewport Screenshot) with some differences. In the Expanded-Viewport Screenshot method, the fixed located element is placed on the output image by its first appearance on the viewport, and whenever the user is scrolling the Web page, the gaze data of the fixed element will have their position as they were recorded on scroll positions and not on their intended fixed element. In this case, the representation of the results will have separated gaze data from their fixated fixed elements. Which causes difficulty for the investigators to assume which gaze data refer to which Web elements. Likewise, the same issue appears on the Stitched-Viewport Screenshot method, except that rather than only having the separated gaze data on several scroll positions, the fixed elements themselves are replicated on the final screenshot beneath their intended gaze data.

The focus of this master thesis is to establish a solution for the problem of lacking the intelligent linking between the gaze data and their intended fixed elements on scrollable Web pages, which produces corrupted visualization results. Similarly, the problem is associated with cursor movements since the fixed elements are also traced with the mouse on different scroll positions. Accordingly, the approach in this thesis applies the Stitched-Viewport Screenshot method and proposes to bind gaze data and mouse trails to their intended fixed elements. The association provides the ability to locate gaze data, mouse trails and fixated fixed elements in a better visualization without having duplicate appearances in the final analytical representation of the Web page. The objective of the approach is to assist in forming enhanced outcomes for more beneficial interpretation and analysis using our proposed Enhanced Stitched-Viewport Screenshot.

Until our Enhanced Stitched-Viewport Screenshot, the problematic behavior was produced in the screenshot visualizations when gaze data are created on fixed elements on different scroll positions. However, it is not the case in the video visualization since it displays the viewport with the fixed elements on their all-session position. Therefore, the gaze data lay on their intended fixed elements throughout the video. Consequently, in the evaluation, we held a comparison between the two approaches that master the problem of this research (our Enhanced Stitched-Viewport Screenshot and the Video). The evaluation results assisted in gaining better insights on which method is more efficient, which one produces greater effectiveness in data visualization and higher interpretation satisfaction.

Initially, we hypothesized that our Enhanced Stitched-Viewport Screenshot allows faster interpretations for the investigators than the Video visualization approach. After studying 10 experts while they're analyzing both visualization methods of a gaze dataset, we concluded that the Enhanced Stitched-Viewport Screenshot is more time-efficient than the Video method in analyzing the second participant since there was a big time difference between both used methods for the second participant investigation. Therefore, we accept our hypothesis with a more accurate detail: our Enhanced Stitched-Viewport Screenshot allows faster interpretations for the investigators in analyzing non-initial participants than the Video approach. Furthermore, our results have shown that the Video approach is more effective in understanding the users' demands in accomplishing their tasks using behavioral analytics since experts were closer to the ground truth values in interpreting the level of difficulty that the participants had while completing their tasks using the Video approach. In addition, the results have reported same level of effectiveness in both visualizations since there were no significant differences in the accuracy of the reported statistical values of the gaze data in both methods.

The results of the evaluation are the key contribution of this master thesis. We also provide two additional technical contributions: first of all, we have implemented our methodology of the Enhanced Stitched-Viewport Screenshot that deals with mouse moves only (gaze data recognition is not yet added) as an open source Web browser example. Secondly, we have combined our methodology with the support of gaze data recognition in a crowd eye tracking software provided by Eyevido in order to use the software in the evaluation.

2. Background and Related Work

This chapter is split into the following topics: Technical work, eye tracking research, Web elements extraction, and image processing.

2.1. Technical details

There are a few technologies and technical terms that we used along with this master thesis:

Cascading Style Sheet (CSS) is a styling language to describe the presentation of the document. There are three different ways to attach CSS properties to Web elements: External, internal, and in-line. To extract the values of all the CSS properties of an element after applying the active style-sheets and resolving any basic computation those values may contain we need to use the *getComputedStyle()* JavaScript method. We focus on the "position" property to catch the fixed positioned elements, and the "visibility", "display", and "opacity" properties to confirm that the element is visible to the viewer eye in the viewport.

Fixed Elements are theoretically all Web elements that are always visible in the same position in the viewport even when the viewer scrolls the Web page. Practically, there are many ways to assign an element to be always visible in the viewport; one option is to assign the *"position"* CSS property to *"position:fixed;"*, another option is to manipulate the element by scripts to keep it in the same position on *onscroll* events. In the Methodology section of this master thesis, we discuss ways to handle the aforementioned techniques, however, we have only implemented the algorithm to handle the elements with *"position:fixed;"* CSS property. Therefore, all future notations of fixed elements indicate the ones with a CSS property of *"position:fixed;"*.

JavaScript (JS) is a programming language, which we used to extract the properties of the fixed elements in Web pages. By injecting our own JavaScript file in the viewed Web pages, we are able to query the Document Object Model (DOM) for fixed elements and get their properties with the *getBoundingClientRect()* method, that returns the size of an element and its position relative to the viewport. The properties are passed on with a call back function in the C++ browser implementation, which are then used to locate the fixed element in the viewport renderer to crop them out from the image. Furthermore, we use the mouse move event to log the coordinates of the mouse trails in a call back function in the browser implementation.

Qt is a C++ framework. We use an example implementation of Chromium[1] to implement our methodology (Enhanced Stitched-Viewport Screenshot with mouse trails). The Qt WebEngine provides functionality for rendering Web content. Our implementation utilizes two main classes of the engine. First is the *QWebEngineView*; a Web engine view is the main widget component of the Qt WebEngine, and it is used to display Web content live from the Internet. The second is the *QWebEnginePage*; A Web engine page holds the contents of an HTML document, the history of navigated links, and actions.

Eyevido GmbH is a crowd eye tracking company, that provides two software infrastructures in Eyevido Lab to conduct Web-based eye tracking studies; the first one is the Recording Tool, which is used to record the gaze and mouse data on Web pages as well as creating a screenshot of the Web page using the Stitched-Viewport Screenshot method. We have contributed our Enhanced Stitched-Viewport Screenshot to the Recording Tool. Secondly, the Analytical Tool, which is a client portal that visualizes the gaze and mouse data for the analysts to use for interpretation on screenshot or video visualizations. We have used both tools to record gaze data and analyze it in the evaluation of this master thesis.

2.2. Eye tracking research

Recent research in eye tracking systems is directed towards the implicit interpretation of the eye movements [10] or focused on the explicit interaction of the target objects by the gaze [16], while this master thesis focuses on improving the interpretation of the gaze data, precisely on Web pages, for perceptible analysis that provides easier interpretations. Moreover, eye tracking plays a significant implicit role in diverse Web applications; in some scenarios, it is used to test the Web usability, while in others it is used to understand the user attention in online advertisements and search services analysis. The results of the studies were significant for decision makers in various scenarios.

Eye tracking has recently been used in various research fields such as in online advertisements. Studies were conducted using eye tracking technologies to capture the effect of certain types of advertisements on customer behavior [11] "Do users look at banner ads on Facebook?". The results have raised the level of understanding on which kinds of advertisements get the most attention and the reason behind that attention. Such analysis helps in figuring out how to successfully publish ads on the Web page layout, taking into consideration important principles that were withdrawn from the research to draw customers awareness. In this particular study area, accurate visualization of the gaze data of user attention is crucial. The representation of the gaze data and the analyzed ad would be corrupted if the ad is fixed positioned on the Web page. When users scroll the page while gazing the ad, the output screenshot will have duplicate images of the ad and its gaze data

[1]Chromium: http://www.chromium.org/

on several scroll positions. Therefore, this master thesis develops an approach for generating the Stitched-Viewport Screenshot with the gathered gaze data and cursor movements on the fixed positioned ads without duplications or separated gaze data.

Furthermore, eye tracking technologies were also utilized in Web usability and design field. Web usability is an essential factor in user satisfaction and, sequentially, business success. It aims to increase the ease of use of websites for users which is achieved by learning users' behavior on the website via eye tracking tools, for example. Studies were conducted to understand how users allocate their visual attention when observing Web pages [8]. Acquiring such knowledge on users' response helps in diagnosing usability problems and enables different roles to make changes to improve users' experience and performance, such as developers, designers, publishers, advertisers and advertising agencies, depending on their perspective. However, to reach the improvements phase, the data visualization analysts need to have complete results that cover all of the Web page elements efficiently, where many Web pages have different kinds of layouts and used technologies. Therefore, the visualization of the results should include all types of possible Web elements (hence, fixed elements) in an organized way for the analysts to have an accurate interpretation without any complications.

A gaze-controlled Web browser was developed to allow users to interact with the Web through eye movements [16]. The explicit eye tracking research of GazeTheWeb[2] mentioned a major challenge is the identification of the interactive Web elements and the ability to control them with eye tracking. The difficulty was tackled by extracting the location of selectable objects on Web pages, such as text input fields, hyperlinks, scrollable sections, select fields, etc. and attach the gaze-control to it. In respect to our research topic, identifying Web elements is a vital challenge. Particularly, part of this research centers on the recognition of the fixed elements and the extraction of their positions to be utilized in the screenshot stitching process.

2.3. Web Elements Extraction

The "Web Block Extraction System Based on Client-Side Imaging for Clickable Image Map" [19] project describes an extraction system of Web elements that are most appealing to users. Users recognize Web elements by selecting a cropping area on the Web page, and then the system extracts the selected Web element with their image and HTML content. Afterwards, the system gathers the selected elements including hyperlinks and images in a cloud storage system for later user access. The extraction in the Web Block Extraction System is a client-side process, whereas a browser extension was implemented to allow users to select the cropping areas. The extension has constituents for capturing a full Web page screenshot, cropping the image, and obtaining hyperlinks for the selected areas. The final generated product is a clickable image map of the selected element. However, the approach imple-

[2]GazeTheWeb: A Gaze-Controlled Web Browser

mented in this master thesis handles the extraction of the Web elements automatically on page events with no user interaction, and then their positions and images are stored in a data structure for later usage.

2.4. Image processing

Furthermore, the ghosting problem has elevated for its difficulty of elimination in image stitching procedures. Ghosting represents the overlap of a dynamic region in multiple images when they are combined into one image. Many de-ghosting algorithms were implemented to exclude the replicated artifact from the main frame. A proposed algorithm in "De-ghosting Method for Image Stitching" [20] plans to find a stitching line that passes around the ghost artifact with an image subtraction method and later on stitch the images respectively to that line. The issue of the ghosting artifact also occurs in this master thesis approach, where some content on the Web page is scrollable, and other content is constantly in a fixed positioned relatively to the viewport. The suggested solution for this challenge relies on the inspection of the scroll position and the locations and dimensions of the fixed elements. The previous properties are acquired by traversing the Document Object Model (DOM) of the Web page, which makes the de-ghosting process handled differently by identifying the properties of the ghost artifact (the fixed element) when stitching the multiple screenshots into one screenshot. The replicated artifacts are cropped on each image creation, and finally, they are stitched in one position.

Many Internet browser extensions have applied a different concept of capturing a full Web page screenshot than our pursued method [2]. For example, FireShot [1] provides an option to capture the entire Web page image. The extension injects the existing fixed Web elements with absolute positioning style and scrolls down the page to capture an image of each visible viewport on scroll positions. Hence, the originally fixed positioned elements will not be visible in each viewport image. Therefore, the full screenshot is made by stitching all captured viewport images. In our proposed method, it is not possible to adjust the style of the fixed elements to be absolute since the page is viewed by the user at the time of capturing the viewport images. Modifying the design of the elements on the Web page does not provide an authentic Web experience. Therefore, we propose to identify the fixed elements on each scroll position in order to crop them out from the viewport image when it is captured. This technique prevents the duplication of fixed elements when stitching all captured viewport images.

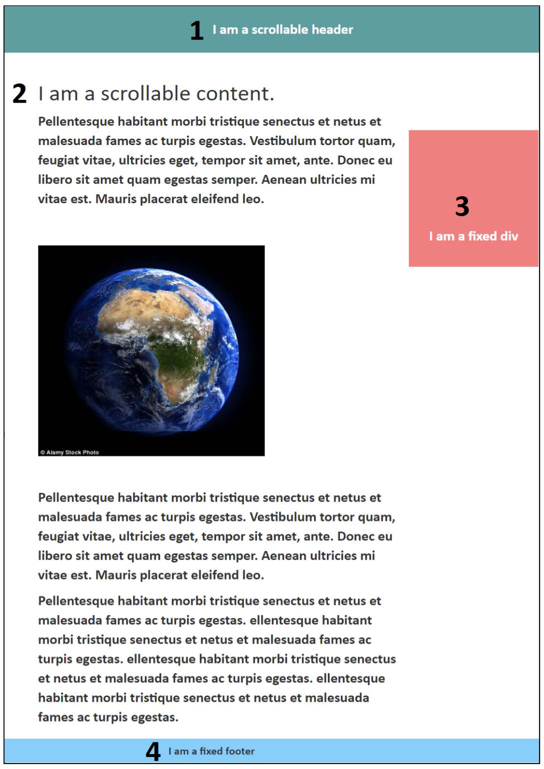
In conclusion, to assemble our solution for the research problem, we have combined together the previous ideas of the introduced research works; recording the gaze on Web pages, identifying and extracting fixed elements, as well as the stitching method in image processing. To the extent of our knowledge, we do not know of any research that explores the tackled topic. Therefore, we believe that this work is the first research on the pursued problem.

3. Research Problem

Eye tracking is a powerful tool used in Web usability studies. However, the analysis of gaze data on scrollable Web pages can be difficult, for the reason that some Web elements are positioned relatively to the viewport and remain visible for the users during the recording session on a fixed position.

The research problem is a combination of two main factors: fixated fixed Web elements, and scrollable Web pages. Capturing gaze data on the fixed elements in scrollable Web pages is challenging to visualize because the gaze data are created on various scroll positions on real-time recordings, and are not combined into one location. This behavior is considered problematic since it presents duplicate fixed elements on recorded scroll positions with their intended gaze data on the created screenshot of the Web page.

An example on a test Web page is shown in Figure 1, which identifies possible different components a Web page may consist of. Figure 1 illustrates four main elements of the Web page. Elements 1 presents a scrollable header with an absolute top position, elements 2 presents a scrollable content, where on page scroll the content will be scrolled over, element 3 presents a fixed element as a <*div*> container, and 4 shows a fixed footer.



1 I am a scrollable header

2 I am a scrollable content.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

3 I am a fixed div

© Alamy Stock Photo

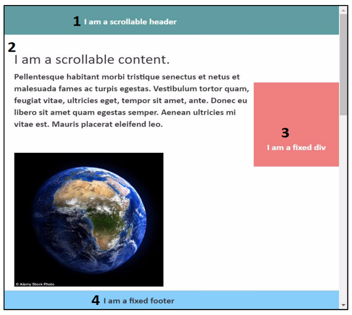
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

4 I am a fixed footer

Figure 1: A test website to identify four parts of the Web page; Element 1 a scrollable header, element 2 a scrollable content, element 3 a fixed <*div*>, and element 4 a fixed footer

To reveal the issue visually, two more figures are created to demonstrate the scrolling issue on fixed elements using the Stitched-Viewport Screenshot method. Figure 2 displays the created screenshot of the examined Web page, where the participant has not scrolled the page. Therefore, all elements are in their initial position. Figure 3 illustrates the created screenshot of the experimented Web page, where the participant has scrolled the page. Accordingly, the fixed elements are replicated along the length of the stitched screenshot.


1 I am a scrollable header

2

I am a scrollable content.

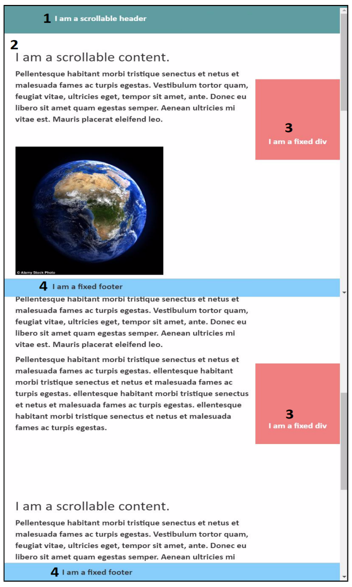
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

3 I am a fixed div

© Alamy Stock Photo

4 I am a fixed footer

Figure 2: The created stitched screenshot for the test Web page (using the Stitched-Viewport Screenshot method) without page scrolling. Hence, the user did not scroll while participating in the study. Notice the stability in the fixed elements positions.


1 I am a scrollable header

2

I am a scrollable content.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

3 I am a fixed div

© Alamy Stock Photo

4 I am a fixed footer

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

3 I am a fixed div

I am a scrollable content.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi

4 I am a fixed footer

Figure 3: The created stitched screenshot for the test Web page (applying the Stitched-Viewport Screenshot method) with scrolling activity. Hence, the user scrolled the page while participating in the study. Notice the duplicate blue fixed footer number 4, and the fixed red <*div*> number 3.

An example on Facebook[3], the online social media website, was chosen to visualize a real-life scenario of the research problem. Figure 4 depicts three fixed elements in the Facebook main Web page (the news feed page): 1. header, 2. left side navigator, 4. sponsored advertisements area, and 5. the chat sidebar. In addition, the figure shows a non-fixed scrollable content in number 3, that represents the news feed.

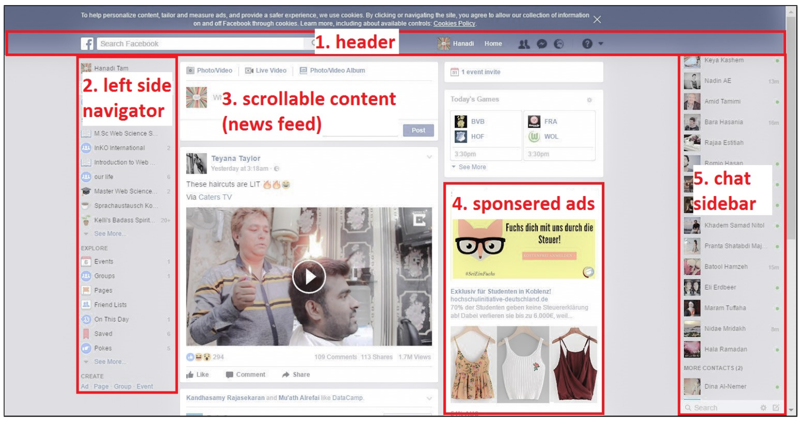


Figure 4: The viewport of Facebook news feed page with numbered sections. (1) A fixed header, (2) A fixed left side navigator, (3) Scrollable content —the news feed —, (4) Fixed sponsored ads and (5) A fixed chat sidebar.

The Facebook example was analyzed with eye tracking. Figure 5 depicts the gaze data on a created stitched screenshot of Facebook taken with the naïve Stitched-Viewport Screenshot approach using a heat-map representation, where the form of the gaze data is painted as warm and cold rainbow colors based on the intensity and duration of the gaze. Starting with blue being the coldest to red being the warmest, the more strong the gaze is the warmest the color is. In the same figure, notice the clutter that resulted when the participant has scrolled down the page, where all the before-mentioned fixed elements are replicated along the expanse of the created stitched screenshot with their gaze data scattered on different recorded scroll positions.

In the first section of the Web page, one may notice that the chat sidebar is duplicated throughout the range of the image with its gaze data in various scroll positions. The reason behind the image replication of the fixed elements is the used methodology in creating the screenshot (Stitched-Viewport Screenshot). The stitched images constantly have the viewport representation and they are taken on scroll events. The header has its position set to a fixed location with specified (top, left, right, bottom) pixels relatively to the viewport, which indicates its first and all-session-long location. The same explanation goes for elements two, four and five of the figure, where the header, the left side navigator, and sponsored advertisements are also replicated with their scattered recorded gaze data. The mouse trails in the screenshot would show the same behavior as the gaze data.

[3]Facebook: https://www.facebook.com

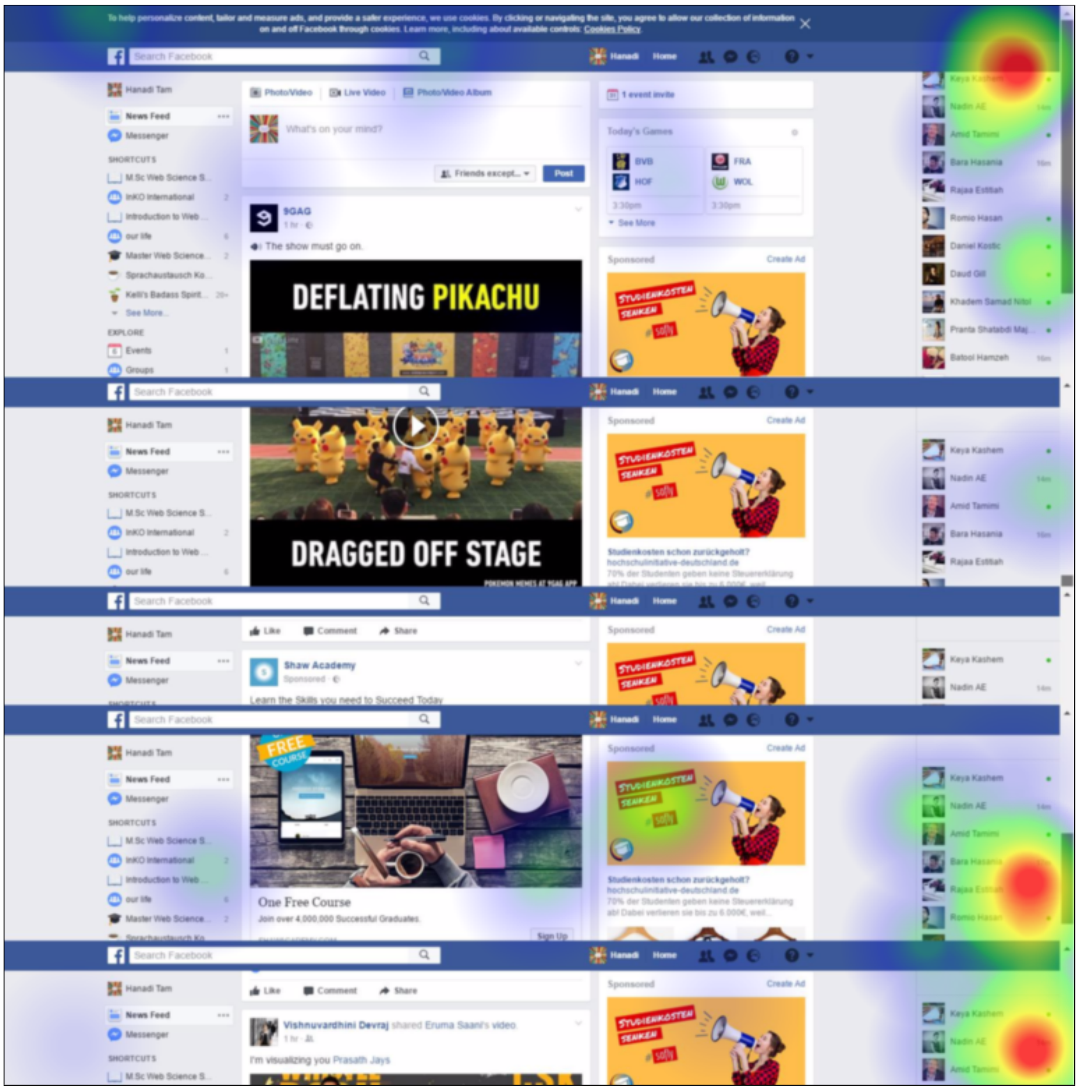


Figure 5: Illustrates the created stitched screenshot of a participant in a Facebook eye tracking study (using the Stitched-Viewport Screenshot method). Notice the replication of the fixed elements (left side navigator, sponsored ads, header, and the chat sidebar.). The gaze data are scattered on each duplicated fixed element as were recorded in the session.

In order to enhance the visualization of the created screenshot by eliminating the replication of fixed elements and the sparse gaze data, we suggest that the expected behavior regarding the fixed elements is to place them on final fixed calculated positions in the final stitched screenshot, as we have defined the final calculated positions to be the following: if the element is shown in the first half of the viewport, the fixed position for it should be in the top of the final created screenshot, however, if the element appears in the second half of the viewport, its last position should be the bottom of the final screenshot. Furthermore, when the expected behavior of the fixed elements is applied, the gaze data would still be scattered on their recorded scroll position. Therefore, the expected behavior of the gaze data is to be gathered throughout the complete Web page and positioned on their intended fixed element area. Lastly, the anticipated behavior of the mouse trails is to be assembled and pointed to the positions of the fixed elements.

The difficulties in accomplishing the desired behavior lay in the following matters: identifying fixed elements in the Web page since they possess various design patterns, handling the gaps in the screenshot that are caused by the never seen content underneath the fixed elements in fast scrolling, the final positioning of the gathered gaze data and their fixed elements on the final stitched screenshot, and many other challenges that are discussed in details in the Challenges 4.1 subsection and their proposed solutions in the Approach concept 4.2 subsection of the following Methodology 4 chapter.

The aim of this master thesis is to improve the visualization of the gaze data on Web pages taking into consideration a special case of fixed elements appearance on scrollable Web content. For efficiently collecting and analyzing gaze data, it is important to deliver the most reliable output of the recordings on Web pages.

4. Methodology

The followed methodology will be discussed in the sequence of the challenges to be faced, the theoretical concept of the proposed approach to solving the research problem as well as overcoming the reported challenges, and the technical implementation of the algorithm. Figure 6 presents mechanism of the Stitched-Viewport Screenshot method that was first discussed in the Introduction, as well as the expected output of our enhancements on the method. The first infographic on the left shows the input, which are multiple viewport screenshots on different scroll offsets. As seen, the viewport has a dashed-line fixed rectangle and a scrollable text. The second illustration shows the final screenshot of the stitched captured viewport screenshots, where the fixed rectangle is replicated in two scroll positions. The third part represents the expected output, which is achieved by the Enhanced Stitched-Viewport Screenshot method that is described further on in this chapter.

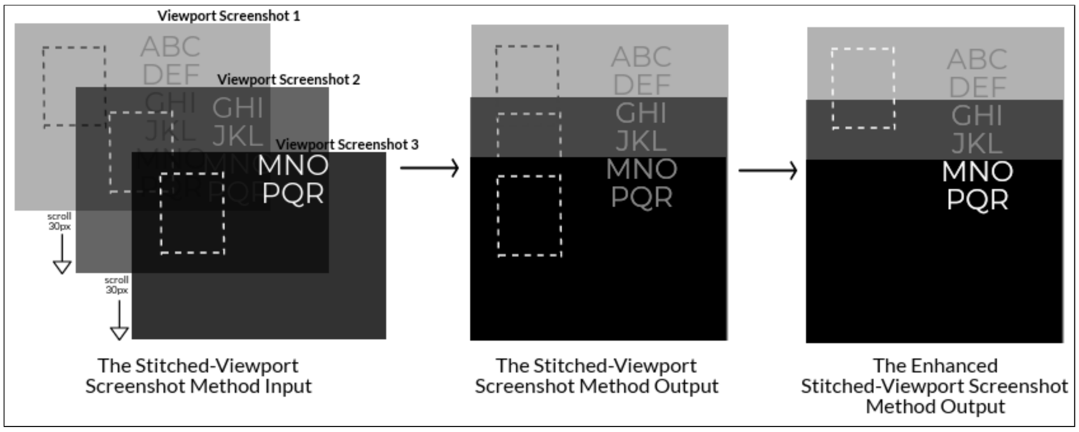

Viewport Screenshot 1
Viewport Screenshot 2
Viewport Screenshot 3
ABC
DEF
GHI
JKL
MNO
PQR
scroll 30px
scroll 30px
The Stitched-Viewport Screenshot Method Input
The Stitched-Viewport Screenshot Method Output
The Enhanced Stitched-Viewport Screenshot Method Output

Figure 6: Illustrates the mechanism of the Stitched-Viewport Screenshot method, as well as the Enhanced Stitched-Viewport Screenshot method. The first part on the left shows the input; three viewport screenshots on different scroll positions that include a dashed-line fixed element and a scrollable text, the middle part shows the output of the Stitched-Viewport Screenshot method, which has two duplicated fixed rectangles on two scroll positions, and the last part on the right presents the expected behavior which we have implemented.

4.1. Challenges

There are several challenges to overcome in order to reach our expected behavior. The following points conclude the difficulties in two different areas, which will be discussed with suggested solutions in the Approach concept 4.2:

4.1.1. Fixed Elements

Fixed Elements are designed and created in various ways. Therefore, identifying them is a hard task. The following are different cases of fixed element presentations:

1. Dynamic change in dimensions: Fixed elements are often transformed via scripts during the viewing time of the page. They are initially rendered with certain dimensions and during special events, their dimensions are modified by scripts.

   For example, in some Web pages, the header is initially designed with a (50px) height, and after scrolling vertically for some pixels, the header is magnified with a (20px) height. The dynamic change in the dimension of the header influences our used approach in identifying the overall-session dimension of the element to be cropped out of the viewport on each scroll position.

2. Dynamic change in visibility: Some fixed elements are created out of the viewport when the page is first loaded, and after a certain timeout or event, they appear in the viewport via a script. Other elements are designed to be viewed

with minus dimensions out of the viewport but still parts of them are visible in the viewport. Another scenario is when elements are designed in the viewport but with a hidden visibility or with an opacity of zero.

As an example, many news feed websites have a timeout newsletter subscription box. The subscription box is a fixed element that is either designed to be hidden or out of the viewport. After a specific timeout, a script makes the box visible. This issue impacts our method of recognizing the elements that are truly in the viewport during the viewing time.

3. Dynamic fixed elements: Web elements can also be designed to be fixed on the Web page without having the CSS property (position:fixed). In this case, a dynamic script manipulates the positioning of the element by setting its position to a fixed amount of pixels on each window scroll event. This technique produces a fixed element in the perspective of the viewer.

   For instance, a "contact us" *<div>* on the right side of the Web page that changes its position gradually when the user scrolls the Web page, which makes it visually as a fixed element. This point is challenging when the running query to identify the fixed elements is searching for elements with the CSS property (position:fixed) instead of checking the previous and current position of each element on each scroll event.

4. The overflow of fixed elements: The returned bounding rectangle of a Web element does not include the bounds of any child element that might happen to overflow. Some fixed elements have sub-elements (children) that are not fixed which leads to the overflow.

   For example, many header menus are fixed and contain sub menus that are not fixed; rather with an absolute position in relation to their fixed parent, which makes them appear to be fixed. In our implementation, we only capture the fixed element itself without being concerned with its children. The issue is only visible if an overflow happens.

4.1.2. Screenshots

Screenshots of the viewport are captured on each scroll position. In each captured image, the fixed elements are cropped, then all images are stitched together into one complete Web page image. The last part is to add the fixed elements on the Stitched-Viewport Screenshot with their gaze data and mouse moves in a suitable location. However, there are many difficulties in handling the screenshots capturing, stitching images, and cropping out fixed elements. The following are the possible cases of the challenges:

1. Gaps in screenshots: In some Web pages anchors are used, where users can jump from one location to another in the Web page without seeing the jumped-over areas. In these cases, it is impossible to fill the regions underneath the

fixed elements when the user has never seen them while scrolling. This issue is also produced when the user uses the scrolling slider of the Web browser to scroll the page, which resembles fast scrolling and the areas under the fixed elements are never visualized.

2. The final position of the gathered data (gaze data, mouse trails, and their fixated fixed elements): when the Web page is accessed with an anchor. In these cases, the first appearance of the fixed element will be the position of the target anchor. However, their final position will be computed as their position (top, bottom) in the viewport. Visually, the image will have black gaps for the unseen sections of the Web page, and the fixed elements will be positioned based on their calculated final position (top or bottom) depending on their appearance in the viewport.

   For example, a Web page which has a fixed header was reached with an anchor in the middle of the Web page. The final image of the Web page will have a black gap in the areas that were not seen by the user (over scrolled). The header will be cropped and added at the top of the final image.

3. Fixed elements overlay other page content: After determining the final position of the gaze data, mouse moves, and fixed elements in the final image, it might occur that their final position overlays other content of the image. This case is noticed when the element initially has a non-fixed position and is set to a fixed position after a certain amount of scrolling. Because in this scenario, the fixed element will be recognized once it is set to be fixed, its properties will be saved, and the final position will be determined based on its fixed position.

   For instance, when a Web page has a header that is initialized as a non-fixed element with these properties in relation to the viewport (0px top , 0px left, 1000px width, 400px height), and the header becomes fixed after scrolling (200px). The fixed header has the same properties (0px top, 0px left), and its final position will be the same values. In this case, the fixed header will overlay the initial header.

4. Replicated surrounding design: In some cases, fixed elements hold surrounding design effects on their CSS Box Model that consists of margins, borders, padding, and the content. The return value of the bounding rectangle of the fixed element does not take into account such effects, for example, the box-shadow, which creates a shadow of the desired color vertically or horizontally around the element. In this scenario, the cropped fixed elements will leave behind their surrounding effects in the image, which will lead to duplications of the design along the screenshot.

4.2. Approach concept

The aim of the approach is to create a connection between gaze data and their fixated elements to assemble them together and place them in one position in the final stitched screenshot. To achieve this goal, the following steps are followed:

Step One: Fixed elements identification and screenshot creation In the first step, an identification of fixed elements in the Web page is essential. The fixed elements are collected from the Document Object Model (DOM) of the HTML page. Their properties (position, dimension) are stored along with their XPath as a unique identifier in a data structure. While capturing the screenshots of the viewport (applying the Stitched-Viewport Screenshot method), the bounding boxes (rectangles) of the fixed elements are cropped out of the screenshots, and the cutout pixels of the fixed elements are saved along with their properties. The milestone of the first step is to achieve a final stitched screenshot of the Web page without its fixed elements. The following paragraphs note the challenges of the first step in both previously mentioned challenging areas (fixed elements and screenshots) and discuss the possible solutions for them.

Fixed Elements 4.1.1: Dynamic change in visibility 2 Elements with hidden visibility property or an opacity of 0 are excluded from the query results since they are not truly visible to the user. Nonetheless, whenever they are visible they will be recognized and added to the query results. Furthermore, if any fixed element is added or created in the DOM by a script, it will be recognized. The dynamic identification of fixed elements is possible since the query is fired on every DOM mutation event, which notifies about DOM changes, and on a constant timeout.

Fixed Elements 4.1.1: Dynamic change in dimensions 1 It is possible to identify an element to be newly or previously added by the use of XPath, which plays a role of a unique ID for each element based on their hierarchal location in the DOM. This characteristic is useful for checking the changes in old elements (their dimensions) in order to update them.

Screenshots 4.1.2: Replicated surrounding design 4 To handle the extra margins, the dimensions of the acquired rectangle of the corresponding fixed element could be manipulated in order to cover the area outside the bounding box, by expanding it to the size of the margin. In the same concept, we can cover other examples such as the box-shadow CSS attribute, which creates a shade for the element.

Fixed Elements 4.1.1: Dynamic fixed elements 3 In order to capture such behavior, there exists a mutation observer that could be attached to the page elements. The observer is able to notify about the changes made on elements and the type of the change. One can obtain the transformation occurring and check whether it is a shift in the position of an element while knowing the previous position of the element. The developer can then comprehend when the element is being positioned in the same location on each scroll event, hence, a fixed element behavior.

Fixed Elements 4.1.1: The overflow of fixed elements 4 We are able to capture the changes in the child nodes of the fixed element by the mutation observer and handle them as fixed elements once they are displayed in the viewport.

Step Two: Gathering gaze data and mouse moves on fixed elements The second step of reaching the desired outcome is to find the gaze data and mouse moves that were recorded on the fixed elements to link them together. Gaze data are stored in a data structure as coordinates that represent the positions of user fixations on the Web page. Therefore, the gaze points that intersect with the created fixed rectangles are considered to be associated with those fixed elements.

Step Three: Inserting the data in their final computed position Finally, after the recording session is over, the created stitched screenshot is completed and filled with the fixed elements in their computed final position with their gathered gaze data and mouse moves from different scroll positions laying on them.

The last positioning of the gaze data and mouse moves among their fixed elements is handled as follows: the elements that are visible on the second (bottom) half of the viewport are located in the bottom of the final stitched screenshot, for example, a footer. Whereas the elements that appear in the first (upper) half of the viewport are placed in the top of the screenshot such as a header. The following paragraphs note the challenges of the last step of the approach and discuss possible solutions for them.

Screenshots 4.1.2: The final position of the gathered data 2 and gaps in screenshots 1 Landing on an anchor area in a page will create an image with a height and width of the current position. If the user never scrolls to the top of the page then the area will remain black, to indicate no viewing. Moreover, the last positioning of the fixed elements remains the same. all fixed elements will be cropped and added to their computed last position, whether it was at the top of the image or bottom.

Screenshots 4.1.2: Fixed elements overlay other page content 3 It is possible to create separate images of each fixed element with their own gathered gaze data and mouse moves, with a transparent background, which could bring better visualization results by laying them on the original final stitched screenshot when needed.

4.3. Implementation

The implementation will carry on a modified open-source browser example based on Chromium from Qt. The used framework provides full control over the Web browser as well as the full access to the Web page renderer. In C++, we are able to acquire the pixels of the viewport to process them as required, while in JavaScript, we inject functions to collect data from the DOM and log the mouse moves. Integrating both technologies (C++ and JavaScript) helps us to run a Web browser that produces screenshots of the surfed Web pages with the mouse trails using our Enhanced Stitched-Viewport Screenshot. Furthermore, in order to add the support of eye tracking, we have contributed our implementation into the Recording Tool, which used the naïve Stitched-Viewport Screenshot method. The output of the Recording Tool are screenshots of the fixated Web pages with the gaze data and mouse moves using our method.

Implementation Challenges The following are the technical challenges that we faced in our implementation:

1. Capturing fixed elements: In order to identify Web elements with their last calculated properties, they have to be fully loaded in the Web page. The *onload* is a JavaScript event that can be attached to Web objects (Window, Document, Element) and is fired when a resource and its dependent resources have finished loading. For example, the *window.onload* event is fired when the entire Web page loads, including its content (images, CSS, scripts, etc.). However, in our initial experiment, the *window.onload* event was not profoundly robust since on some Web pages it was fired before the pixels are rendered or long after the page is visible to the user due to other running technologies like Ajax calls, for example. This issue makes it difficult to capture the fixed elements at the exact time when the user observes them.
2. Utilizing the runtime efficiency: of the algorithm when determining the data structure to store the properties of the fixed elements; positions, dimensions, image, gaze data, mouse moves, and XPath.
3. Memory optimization: is a big challenge since we are dealing with images, image processing methods, a large amount of gaze data as well as mouse data on frequent script calls.

There are three technical steps in this approach:

Step One: Fixed elements identification and screenshot creation In this step, fixed elements are extracted using a JavaScript function, that returns their properties (position, dimension, XPath). The properties are then passed to the C++ world in order to capture the pixel of the fixed elements and execute the Stitched-Viewport Screenshot method. The following points give further details on this step:

1. A JavaScript function (*getFixedElements()*) is injected to query the DOM for fixed positioned elements. The query returns the computed style properties of the fixed elements (bounding rectangles — the size of an element and its position) regardless of the CSS Modal Box effects such as the margins. All fixed elements are logged into the console of the Web page by JavaScript and captured in C++ by the *javaScriptConsoleMessage()* function provided by the *QWebEnginePage* class or are captured by a callback function in the *QWebEngineView*.

2. The JavaScript function (*getFixedElements()*) is called whenever the following events are triggered:

   a) *Window.onload*: is a JavaScript event that is fired when the entire page is loaded, including its content (images, CSS, scripts, etc.), which is crucial to get the latest computed style properties. However, this event was found inconsistent for our needs since sometimes the *window.onload* event is fired before the pixels are fully rendered or long after the page is viewed by the user. Figure 7 shows a column chart of five access trials that we made to check the consistency of the *window.onload* event. We have tested it on three different Web pages (`www.google.de`, `www.9gag.com`, `www.eyevido.de`) that uses different kinds of technologies. In the Google Web page, the window load was fired once on each access. Twice out of three accesses the pixels were fully rendered before the *window.onload* was fired. While in the 9gag Web page, the event was fired (53, 87, 53, 52, 42) times, respectively to access trials. Many elements were clearly visible before the first event was fired, and some were not rendered after the first event was noticed. However, after the last *window.onload* event, all elements were fully rendered on the Web page. In Eyevido Web page, the event was noticed three times. The page had fully rendered elements before the first *window.onload* event was recorded. Therefore, we have realized that the event is not consistent enough in our implementation since it profoundly relies on the technologies used in the tested Web page. Consequently, we have also included the next event trigger for more robustness.

   b) *Window.onscroll*: triggered when the scroll position is changed. On each scroll position that the user hits while viewing the Web page, the *getFixedElements()* function will be executed. However, it is possible that the user remains on a certain scroll position and a fixed element is created dynamically after a timeout in the viewport. In that case, the fixed element will not be recognized on that scroll position because the function to identify it was called before the creation of that element. Therefore, we create a timer that is triggered every 200 ms to check the current scroll position and execute the (*getFixedElements()*). Introducing the timer helps in capturing more cases of fixed element creations on timing bases instead of scroll event based. The timeout of the timer could be adjusted

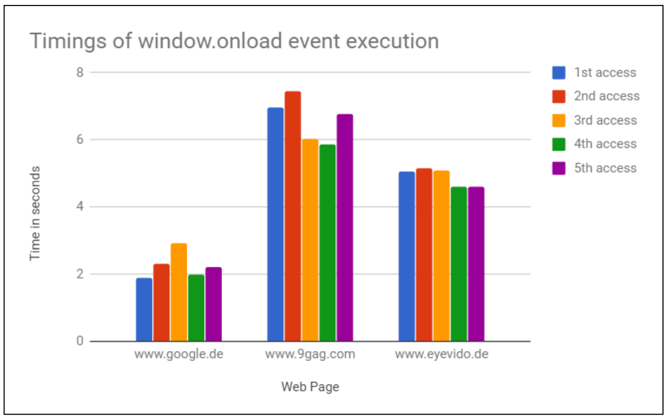


Figure 7: Time measurements of the *window.onload* event on 3 Web pages. Each column group shows the statistics of 5 access trials.

as wished. However, it was tested on many Web pages that the 200 ms timer produces higher quality of images than the slower timers such as 400 ms, which produces gaps in the screenshot on fast scrolling since the scrolling can be faster than the timeout.

3. After receiving the fixed elements' properties in C++, corresponding fixed element objects are created. Each object has a rectangle, an image filled with the pixels of the fixed element position in the Web page, an XPath for unique identification, mouse data vector, gaze data vector (explicit for the Recording Tool implementation), and a visibility boolean that tracks the current visibility of the element in the viewport.

4. Then the Stitched-Viewport Screenshot method is executed to create images of the viewport with an extra step of cutting off the previously created fixed elements. The created images are stitched together on the scroll position offset of each newly captured image.

Step Two: Gathering gaze data and mouse moves on fixed elements Gaze data and mouse moves that cross the bounding box of a fixed element throughout the Web page are linked together and added to their vectors in the fixed element object. Gaze data are collected from the eye tracker (explicit for the Recording Tool), while the mouse moves are logged in JavaScript and captured in C++, as were the properties of the fixed elements.

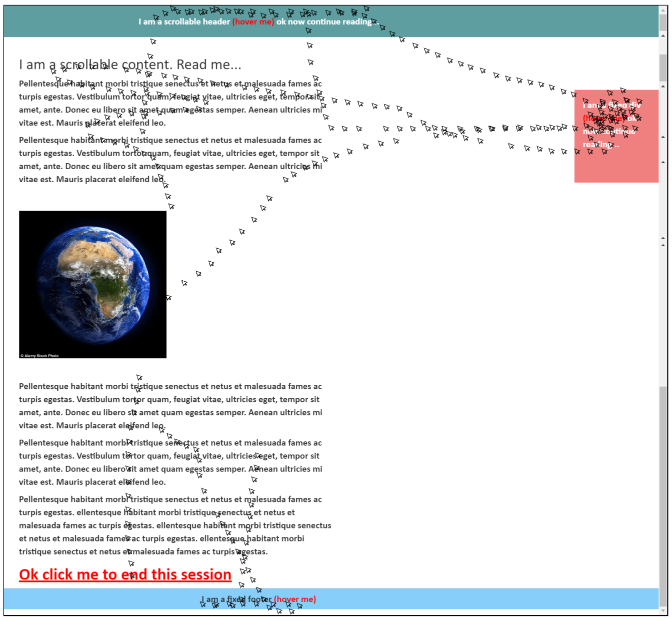
Step Three: Inserting the data in their final computed position The final step describes the reassembling of the fixed elements pixels on the final screenshot with their gaze data and mouse moves. Fixed elements with their gaze data and mouse moves are gathered from various scroll positions and placed on their computed last position. This step is executed on page leave, i.e. when a link is clicked. The final stitched screenshot is saved and the algorithm starts from the beginning with a new image.

Technical Contributions Our first contribution is an open source project based on a Qt Chromium example using the Qt WebEngine to prove our algorithm. Initially, the example had a lot of features that we have excluded for simplicity. We ended up with basic classes provided in the Qt WebEngine such as *WebView* and *WebPage*, which are inherited from *QWebEngineView* and *QWebEnginePage*, respectively. Our code is open sourced and pushed into a GitHub repository `https://github.com/hanadi92/demobrowser` under the project name of The Mousetracker Browser[3]. The browser handles the mouse moves and not the gaze data, however, it is possible to adapt the implementation with eye tracking technology, which we have achieved in the Recording Tool.

Figure 8 shows an example of the output image using our implemented open source program, where a test Web page was created (used in the Research Problem section) with two fixed elements (the red <*div*> and the footer). The user was hovering on (Hover me) occurrences then clicked on (Click me) link to end the session. The program exports two image files; the first image is Web page output of the Stitched-Viewport Screenshot method, and the second one shows the mouse trails with a transparent background. This feature allows users to merge both images into one frame using an image editor or use them individually.

Our second technical contribution was improving the Recording Tool. We integrated our implementation with the support of eye tracking into the Enhanced Recording Tool. The Enhanced Recording Tool was then used in the Evaluation 5 of this thesis. Figure 9 represents an example output of the test Web page, where the circles represent the fixations, which are generated by maintaining the visual gaze on a single location, and the lines represent the mouse trails.

Structure Diagram In order to assemble our implementation, we have created a JavaScript function to query the DOM for fixed elements (*getFixedElements()*), a JavaScript event listener for mouse moves, a C++ class called (*FixedElement*) to maintain the fixed element objects, and a class called (*ScreenShotter*) to handle the creation of the screenshots. We have also made some editions in the *WebView* class to handle computations of cropping fixed elements from the viewport images, and we have overridden the function responsible for catching console messages in the *WebPage* class. Furthermore, in the Recording Tool, we have utilized the implemented (*Eyetracker*) class, where a function is triggered with the gaze point coordinates whenever a fixation is captured. As shown in the descriptive structure diagram in Figure 10, where the relations between the five classes are illustrated, and each class is shown with its own operations and members. The following paragraphs explain the diagram in different areas.


I am a scrollable header (hover me) ok now continue reading...

I am a scrollable content. Read me...

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

© Alamy Stock Photo

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

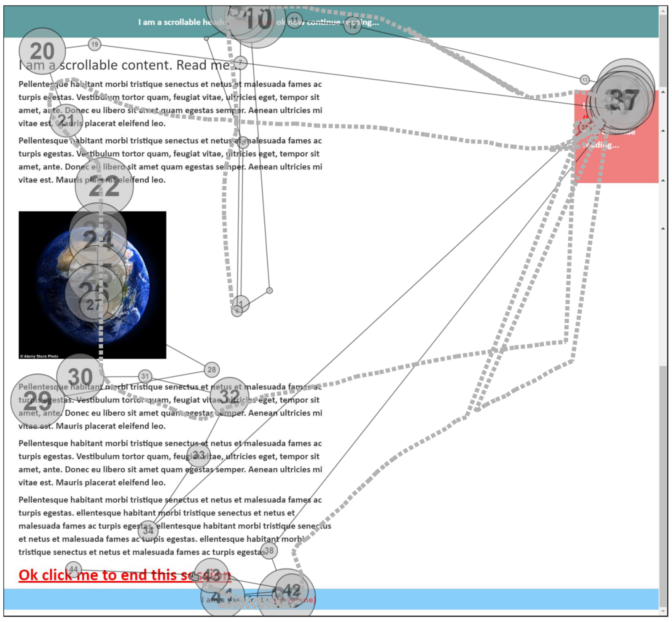
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Ok click me to end this session

I am a fixed footer (hover me)

Figure 8: Illustrates the combined image (of the two output images) of our implemented browser using a test Web page. The first image represents the Web page using the Enhanced Stitched-Viewport Screenshot method and the second one has the mouse trails with a transparent background.


I am a scrollable header (hover me) ok now continue reading...

I am a scrollable content. Read me...

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

© Alamy Stock Photo

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. ellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Ok click me to end this session

I am a fixed footer (hover me)

Figure 9: Displays the output of the Enhanced Recording Tool using a test Web page. The circles illustrate the fixations, and the dashed lines represent mouse trails.

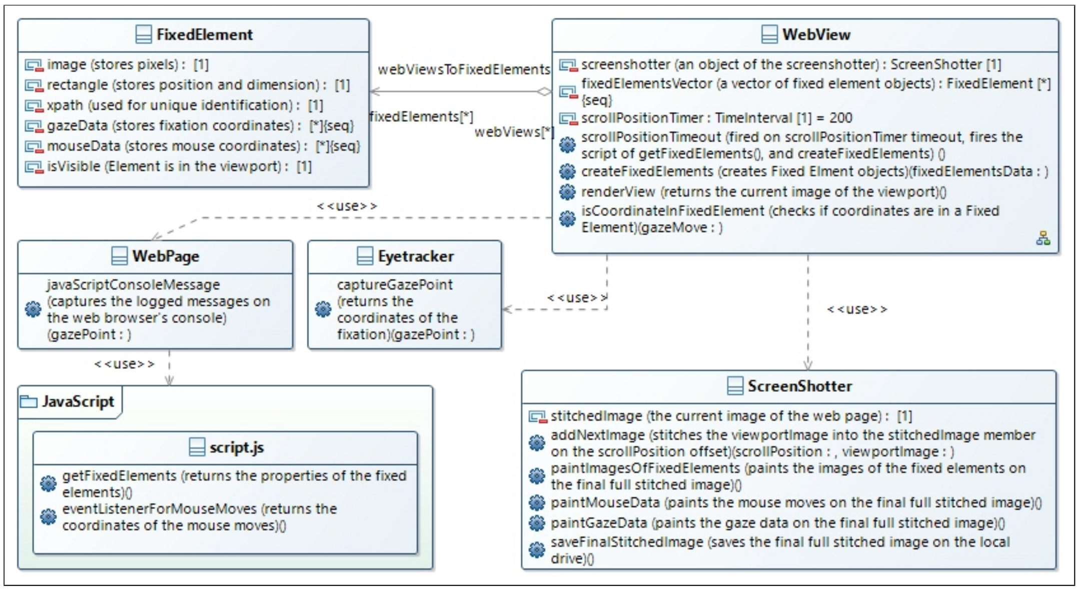

FixedElement
image (stores pixels) : [1]
rectangle (stores position and dimension) : [1]
xpath (used for unique identification) : [1]
gazeData (stores fixation coordinates) : [*]{seq}
mouseData (stores mouse coordinates) : [*]{seq}
isVisible (Element is in the viewport) : [1]
webViewsToFixedElements
fixedElements[*]
webViews[*]
WebView
screenshotter (an object of the screenshotter) : ScreenShotter [1]
fixedElementsVector (a vector of fixed element objects) : FixedElement [*] {seq}
scrollPositionTimer : TimeInterval [1] = 200
scrollPositionTimeout (fired on scrollPositionTimer timeout, fires the script of getFixedElements(), and createFixedElements) ()
createFixedElements (creates Fixed Elment objects)(fixedElementsData :)
renderView (returns the current image of the viewport)()
isCoordinateInFixedElement (checks if coordinates are in a Fixed Element)(gazeMove :)
<<use>>
WebPage
javaScriptConsoleMessage (captures the logged messages on the web browser's console)(gazePoint :)
Eyetracker
captureGazePoint (returns the coordinates of the fixation)(gazePoint :)
<<use>>
<<use>>
<<use>>
JavaScript
script.js
getFixedElements (returns the properties of the fixed elements)()
eventListenerForMouseMoves (returns the coordinates of the mouse moves)()
ScreenShotter
stitchedImage (the current image of the web page) : [1]
addNextImage (stitches the viewportImage into the stitchedImage member on the scrollPosition offset)(scrollPosition : , viewportImage :)
paintImagesOfFixedElements (paints the images of the fixed elements on the final full stitched image)()
paintMouseData (paints the mouse moves on the final full stitched image)()
paintGazeData (paints the gaze data on the final full stitched image)()
saveFinalStitchedImage (saves the final full stitched image on the local drive)()

Figure 10: Presents a structure diagram of the implemented algorithm that contains the *WebView* and *WebPage* classes among two newly introduced classes (*FixedElement* and *ScreenShotter*). The *Eyetracker* class is only used in the Recording Tool integration.

Fixed Elements Identification Initially, the JavaScript file is injected into the loaded Web page through a higher class than the *WebView* called the *QTabWidget*, which holds into a stack of tabbed widgets, where the widgets are the user interface objects of the application. There are two approaches of retrieving fixed elements' properties: a JavaScript event initializer (*window.onload*), and a C++ callback function (*scrollPositionTimeout()*).

JavaScript Event In this method, the *getFixedElements()* function is fired when the event listener *window.onload* is fired when the page is loaded with all of its content. The function logs the properties through a console message, which are captured by the *WebPage* function *javaScriptConsoleMessage()*. The passed string is filtered and the *WebView* function *createFixedElements()* is called to create fixed element objects and push them into the member vector of fixed elements. Hence, this way is fired only once, on *window.onload* event.

C++ callback Function The *WebView* loads the Web content in the Web browser, and holds the timer parameter (*scrollPositionTimer*). On each timeout, the *ScrollPositionTimeout()* runs a script call to execute the *getFixedElements()* function, that returns the fixed element properties. The callback of the *ScrollPositionTimeout()* captures the properties and runs the *createFixedElements()* function to create fixed element objects corresponding to the passed properties.

Elements Visibility The visibility of an element is determined in JavaScript by calculating the appearance of its rectangle (position, dimension, opacity) in the viewport. However, the determination in C++ is achieved by checking the console logged *XPath* property of the existing fixed elements against the returned fixed elements in the callback function *scrollPositionTimeout*. If the *XPath* of the existing element matches any of the *XPath*s of the returned elements, then they are marked as visible using the Boolean *isVisible*. Otherwise, if the *XPath* of the existing element does not occur in the returned elements, then they are marked as invisible.

Stitched-Viewport Screenshot Additionally, the *ScrollPositionTimeout()* function fires a series of activities in order to achieve the Stitched-Viewport Screenshot method: *renderView()*: to capture an image of the viewport, crop out the visible fixed element regions, and send it with the *scrollPosition* to the *ScreenShotter* object, which holds a member of the so-far stitched Web image. The *ScreenShotter* is created with each *WebView* and it is responsible for maintaining the image stitching process. It receives the current viewport in *addNextImage()* function, and stitches it to the member *stitchedImage* on the offset of the passed *scrollPosition* parameter.

Mouse Data The mouse moves are logged through console messages through the injected JavaScript file and captured in the *WebPage* function *javaScriptConsoleMessage()*. They are then passed to the *WebView* class in order to check if they are in relation to the fixed elements or not. In *isCoordinateInFixedElement()* function, we access the vector of fixed element objects and check on each received mouse move if the coordinates are reported to be inside the rectangle of the object. Mouse moves that cross the rectangles are stored in the *mouseData* series member of the intended object. In the Recording Tool, gaze data are captured through the *Eyetracker* class function *captureGaze()* and then passed to the *WebView* class in order to check if the coordinates lay in the fixed rectangle or not using the same function *isCoordinateInFixedElement()*. When the fixation lays in the fixed object, it is stored in the *gazeData* series member of that object.

Final Position and Gathering of Mouse data Before saving the final Stitched-Viewport Screenshot, in the *ScreenShotter* function *saveImage()*, we access the fixed element objects and check their position in relation to the viewport size in order to calculate their final position on the final image. Furthermore, we manipulate the mouse data coordinates (*paintMouseData()*) in order to collect them over their intended fixed elements by checking the position of the fixed element that owns the data, when the element is in the top, its data coordinates stays the same since the reported coordinates have an (x) and (y) relatively to the viewport. However, when the fixed element is in the bottom, the coordinates are manipulated by adding the last recorded maximum scroll position in the Web page in order to show the data in the bottom of the final image. The same logic is adapted to the gaze data in the Recording Tool (*paintGazeData()*).

Image Saving The *ScreenShotter saveImage()* function is fired on three conditions: when the session is over i.e. if the user hits the (x) button to close the tab, or on an accepted navigation request i.e. on success hyperlink click, or if the user requests to save an image by clicking on the save image icon in the toolbar (an old floppy disk icon).

Algorithm Workflow There are four activities in the workflow of the algorithm. The following points explain each activity with a reference diagram figure. The order of this list is not important for the implementation, however, they were listed in this order for better understanding:

- Fixed elements Identification and Creation: In Figure 11, the workflow of capturing fixed elements methods is illustrated. Where it states the JavaScript Event method starting with the loading event that calls the function to query for fixed elements and logs the data as a console message. The *WebPage* captures the console message and send it to the *WebView*, where the creation of fixed element objects is executed.

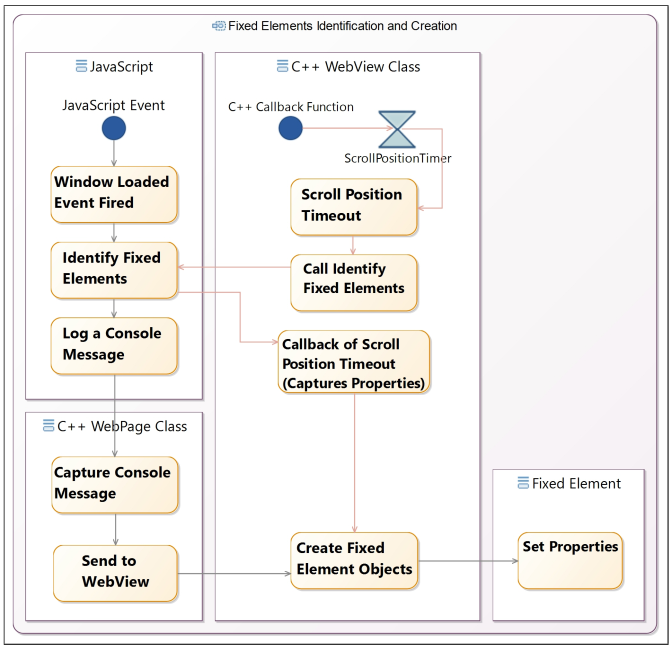


Figure 11: The workflow of fixed elements identification and creation in both implemented methods; through the JavaScript event *window.onload* and the C++ callback function *ScrollPositionTimeout()*.

- Capturing Mouse Data: Figure 12 shows the workflow of the previous activity in gray color (inactive) and the mouse data capturing activity in yellow color (active). Both activities are independent from each other. In mouse detection, the initial trigger starts with a JavaScript event listener for mouse moves which logs the coordinates into the console log. The *WebPage* captures the log messages and forwards it to the *WebView* in order to check if the coordinates are inside a fixed element, which will be added to the intended fixed element object, otherwise, the move is stored regardless of fixed elements.

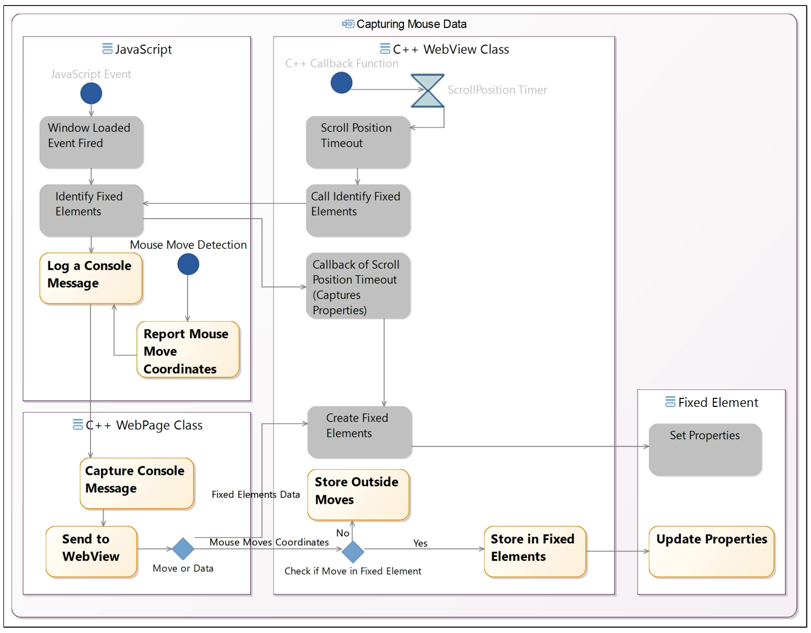


Figure 12: The workflow of mouse data capturing in JavaScript and storing in C++. The active workflow is illustrated in yellow color and blue initial node with black title, while the gray color shows the previous inactive activity of fixed elements identification and creation.

- Capturing Gaze Data (only in the Recording Tool): This activity deals with fixation detection, which is reported to the *WebView* by the *Eyetracker* class. The gaze coordinates are then handled in the same way the mouse data is. Figure 13 shows the workflow of the gaze data reporting in active colors.

- Applying the Enhanced Stitched-Viewport Screenshot Method and Image Saving: Figure 14 reveals the workflow of the stitching method and the image saving in active colors, which are of the *ScreenShotter* responsibility. The stitching starts on each timeout of the *scrollPositionTimer* to capture an image of the viewport and crop out fixed elements, if available. Then it sends the image to the *ScreenShotter*, which stitches the received image into its own member image of the Web page on the passed *scrollPosition* parameter offset. Image saving is triggered by session ending, navigation, or on request. The trigger is sent to the *ScreenShotter*, which first adjusts the data for final positioning, saves the image on local disk, and the stitching method resets to an empty image.

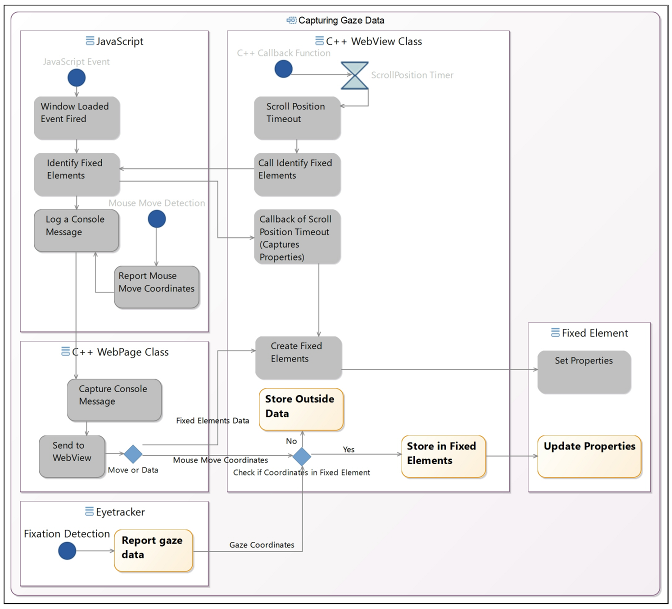


Figure 13: The workflow of gaze data where the *Eyetracker* class captures the fixation coordinates and report them to the *WebView*. The active workflow is shown in yellow colors and a blue initial node with black title.

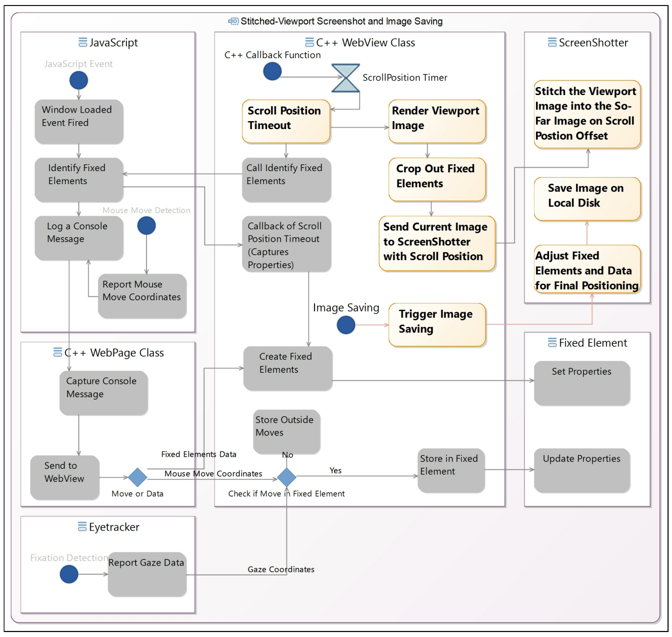


Figure 14: The Stitched-Viewport Screenshot and image saving workflow is illustrated in the active color. Starting from the blue node with black title.

In summary, we were able to achieve our expected behavior by applying the previously described methodology in practice. We have faced and overcame the challenges that were mentioned in Challenges 4.1. Furthermore, there are some improvements that can be contributed to our approach for a better sense of visualization, that were described in theory (in Approach concept 4.2) but not yet implemented: the dynamic fixed elements, the overlaying of fixed elements on other page content, the overflow of fixed elements, and the replicated surrounding design.

5. Evaluation

In this chapter, we are going to introduce our used methods in the evaluation, the outcome of the evaluation, and the interpretations of our results and discussions on further explanations. Since we were able to overcome the research problem using our Enhanced Stitched-Viewport Screenshot, we want to compare it against another visualization method that also overcomes the problem. Therefore, we chose the Video visualization method, where the videos display the viewport of the user session that has the fixed elements on their all-session position with their gaze data. It was inconvenient to choose the naïve Stitched-Viewport Screenshot because the results would have been too obvious and more biased towards the enhanced method. Also, the Expanded-Viewport Screenshot method would not have made a good candidate since it still has the issue of sparse gaze data on several scroll positions that are not connected to their fixed elements. The evaluation studied experts evaluating both visualizations and recorded their response timings to some tasks. The results of the evaluation are shaped as quantitative and qualitative forms, which are then interpreted and analyzed to confirm the following suggested hypothesis: **our Enhanced Stitched-Viewport Screenshot allows faster interpretations for the investigators than the Video visualization approach**. Furthermore, we explain the results and note down important remarks on further research.

5.1. Methods

The plan is to conduct an eye tracking study using the Recording Tool, which was extended by our implemented Enhanced Stitched-Viewport Screenshot method. The output of the study forms a dataset of interactive data (gaze and mouse data) on Web pages, which will be analyzed by experts in both studied visualization methods using the Analytical Tool. The Analytical Tool presents both screenshots and videos in the same design in terms of time controls, scan-path, statistical parameters such as fixation duration, and the ability to create areas of interest (AOIs).

The study proceeds through two phases in order to complete the assessment and be able to accept or reject the hypothesis. There are three roles contributing throughout the two phases; the first phase revolves around the participants, who join the eye tracking study to provide interactive data on several Web pages while completing different tasks. The second phase is covered by the experts, who analyze the gathered dataset while accomplishing tasks on data analysis, and by the evaluator, who engages with the experts to present an introductory workshop and report the responses of the given tasks.

5.1.1. Phase one

The first phase includes conducting a study using the Recording Tool in order to collect the eye tracking dataset of several Web pages. The ground truth is also determined in this step; by the participants, through answering a questionnaire, and by the evaluator, through observing the dataset.

Creating the eye tracking dataset The aim of the first phase is to collect the dataset of interactive data on four different Web pages. Two participants are asked to join the study using the Recording Tool while a screen recording application is running in the background to capture the user session on each Web page. The Recording Tool uses an Eyetracker to capture the fixations of the participants on the Web pages and produces screenshots using our Enhanced Stitched-Viewport Screenshot, while the screen recorder produces videos of user sessions on the Web pages. Since static screenshots have the advantage to combine all participants of the same Web page in one screenshot, we end up with a total of 4 screenshots of the 4 Web pages. Each Web page has the interaction data of the two participants. However, the combination is not possible in the Video approach, because of different user behavior on the Web page, scrolling behavior, for example. Therefore, there are 8 videos in the dataset, each video represents each user session on each Web page. Figure 15 illustrates an infographic of phase one.

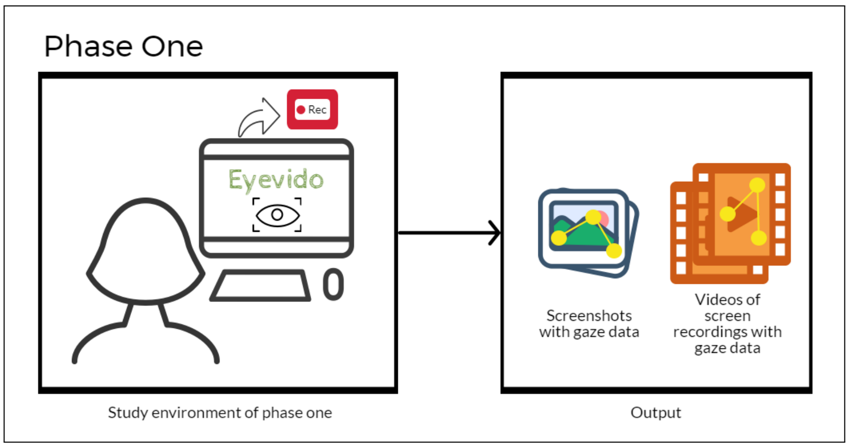

Phase One
Rec
Eyevido
Screenshots with gaze data
Videos of screen recordings with gaze data
Study environment of phase one
Output

Figure 15: The study environment [left]: Participants join the study using the Recording Tool while a screen recording tool is recording their session in the background. The output [right]: The outcome of this phase is the dataset of screenshots and videos.

Choosing two participants is sufficient for our dataset formulation since it will produce an adequate amount of interactive data on a satisfactory number of screenshots and videos of each Web page. It is crucial to understand that the focus of this evaluation is not the gathered dataset; rather it is the interpretation of the interactive data and visualizations in the dataset, which is discussed in more details later on in the second phase, where we distribute the dataset among the experts to analyze them.

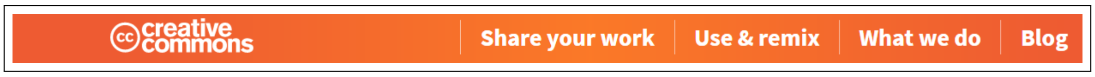
Web pages We have chosen to run the study on four different Web pages: Creative Commons `https://creativecommons.org/licenses/by/2.0` which allows people to easily and legally share knowledge and creativity, Yelp `https://www.yelp.com/search?find_loc=koblenz` to connect people with great local businesses, Jimdo `https://www.jimdo.com/` which is a Web hosting service, and Digg `http://digg.com/` that delivers news and most interesting stories on the Internet. These Web pages were chosen for their visiting ranks noted on `https://moz.com/top500` as well as the appearance of different types of fixed elements in their design which we want to evaluate.

The fixed elements that appear in the aforementioned Web pages are shown in Figure 16. Creative Commons Web page can be considered as a complex page in terms of having two fixed elements that appear in the viewport after a timeout by a script, see sub-figures 16a and 16b, Yelp can be considered as less complicated since it has a fixed column on the right that is set to (position: fixed) by a script on scroll, see sub-figure 16c, Jimdo can be considered as a less complex design in regards to fixed elements since it has a fixed header on scroll, see sub-figure 16d, and finally, Digg can be considered as the least complicated one since it has a fixed header, shown in sub-figure 16e.

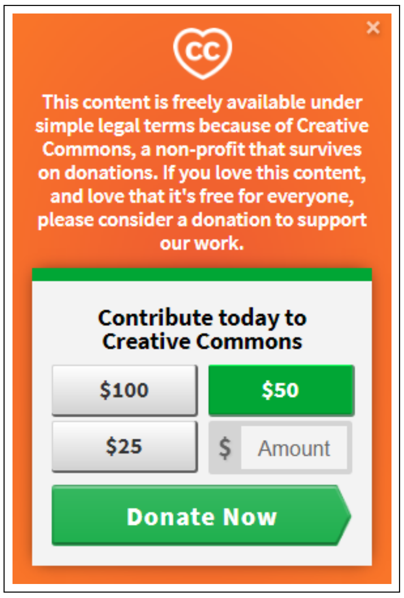
Our focus is centered on analyzing fixed elements because it is hard to investigate an AOI on a scrollable content in the Video method. The expert has to draw the AOI on the first appearance of the scrollable content, enclose the time where that element has appeared, and keep a pen and paper to note down the statistical values of the area at that time. Whenever the viewport is scrolled, the expert has to drag the AOI to cover the new position and do the same steps again. However, when investigating a fixed element in a video, it will be always visible in the viewport on the same position. On the other hand, maintaining scrollable content in the Enhanced Stitched-Viewport Screenshot method requires no more effort than creating the AOI on the static screenshot, which is the same case for fixed elements.

Tasks for the participants The tasks are designed to cover the whole page vertically, in order to assure the correctness of the used stitching method that we have implemented. The participants are asked to freely navigate through the page to accomplish the following tasks:

- Creative Commons: The participants are asked to find the Frequently Asked Questions (FAQs) link and click on it.



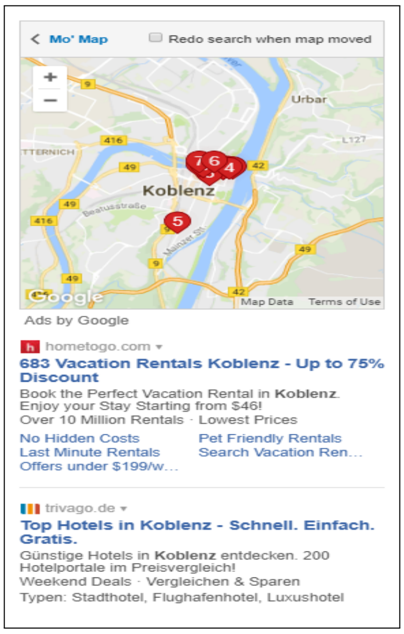
(a) A fixed header on `https://creativecommons.org/licenses/by/2.0` Web page that appears on scroll up.



This content is freely available under simple legal terms because of Creative Commons, a non-profit that survives on donations. If you love this content, and love that it's free for everyone, please consider a donation to support our work.

Contribute today to Creative Commons

(b) A fixed element on `https://creativecommons.org/licenses/by/2.0` Web page that appears on the left side of the viewport after a timeout by a script.

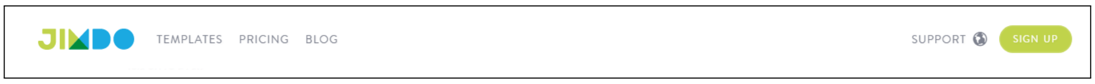


Ads by Google

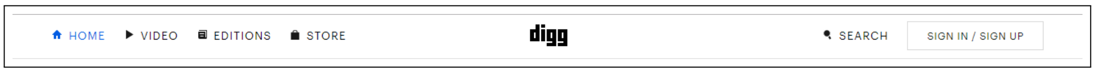
hometogo.com ▾
683 Vacation Rentals Koblenz - Up to 75% Discount
Book the Perfect Vacation Rental in Koblenz. Enjoy your Stay Starting from $46! Over 10 Million Rentals · Lowest Prices
No Hidden Costs · Pet Friendly Rentals · Last Minute Rentals · Search Vacation Ren... · Offers under $199/w...

trivago.de ▾
Top Hotels in Koblenz - Schnell. Einfach. Gratis.
Günstige Hotels in Koblenz entdecken. 200 Hotelportale im Preisvergleich! Weekend Deals · Vergleichen & Sparen
Typen: Stadthotel, Flughafenhotel, Luxushotel

(c) A fixed element on the right side of the viewport on `https://www.yelp.com/search?find_loc=koblenz` Web page.



(d) A fixed header on `https://www.jimdo.com/` Web page that appears after a certain scroll position.



(e) A fixed header on `http://digg.com/` Web page.

Figure 16: The fixed elements that appear on the Web pages in the first phase.

- Yelp: The participants are asked to choose the 10th result of the search results.
- Digg: The participants are asked to find the Frequently Asked Questions (FAQs) link and click on it.
- Jimdo: The participants are asked to find the (work with us) link and click on.

The participants are well informed about the possibility to terminate the study at any point if they felt unpleasant or unable to complete the requested task by pressing the (ESC) key on the keyboard or closing the tab window simply by selecting the (x) button on the browser tab.

Ground Truth The ground truth is assembled in two strategies:

- By the participants: reporting the answers of the questionnaire "How challenging was the task for you?" after viewing each Web page.
- By the evaluator: documenting the results from the participants and the true statistical values in the dataset, for example, the total number of fixations on the fixed elements.

Technical Details The video produced of each participant session by the screen recording software has all four Web pages. Therefore, we need to manually crop the video in order to get a recording of each Web page interaction. The gaze data for the videos will be collected from the connected Eyetracker through the Recording Tool which is running in parallel to the screen recorder. This will guarantee to have the exact same gaze data of the same Web page in the video and on the Enhanced Stitched-Viewport Screenshot.

5.1.2. Phase two

The gathered dataset in the first phase is used as an input in the second phase for the experts investigations. The experts are asked to attend a workshop, that will bring them to better understanding of eye tracking metrics and introduce them to the Analytical tool that will be used for interpreting both data visualizations. They will learn how to interpret the gaze data on screenshots and videos to be able to participate in our evaluation. After the workshop, the dataset is distributed among the experts, and tasks are assigned to them. Meanwhile, the evaluator observes their response time of the tasks and communicates with them for answers. The output of this phase is a full report on the results of the experts' investigations on the dataset.

The Environment Set-Up in The Analytical Tool Both evaluated data visualizations are represented in the Analytical Tool with equal designs, such as: the gaze scan-path, the mouse dashed-lines, a timeline to play the interactive data, zooming tools, and AOI drawing. At the first sight of the visualization, the overall interactive data is displayed on the screenshot or video, and the timeline is used to play the scan-path or the dashed-lined trails. Also, the visualization is initially fitted into a frame, and can be zoomed in or out, where the video has the visual of the viewport with the scrolling behavior and the screenshot remains static. Furthermore, The screenshot has a combination of all participants interactive data, which can be switched between to show or hide the data of the participant. Figure 17 and Figure 18 show examples of both visualizations in the Analytical Tool. The first visualization is a screenshot of the Creative Commons Web page that has the overall interactive data of two participants displayed. The AOI was selected on the fixed element in that Web page. The second visualization is a video of one participant session on the Jimdo Web page that has the overall of their interactive data. The video starts with the loading of the Web page, and ends at the end of the session. The AOI was selected on the header of the Web page that is visible once the video is played. In both visualizations, the AOI calculates the fixations that were made in the defined timespan on the timeline.

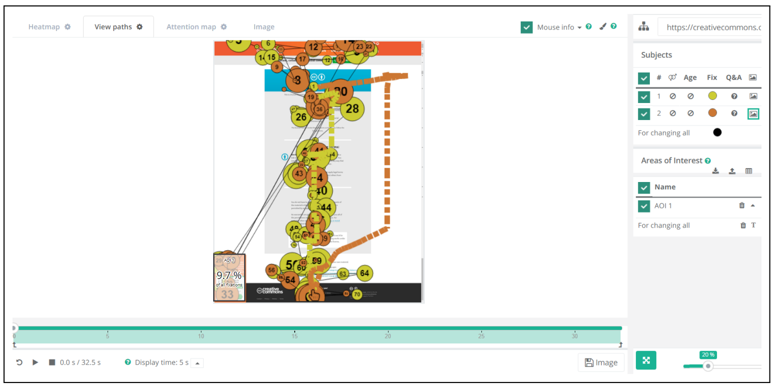


Figure 17: Shows the screenshot visualization in the Analytical Tool, where the overall interactive data of both participants are displayed on it. The scan-path of the gaze data and the dashed-lines of mouse data are played on the screenshot by pressing the play button of the timeline.

Workshop The second phase starts with collecting 10 experts, whom we are going to study while they are examining our dataset. The experts are defined by their knowledge in the topics which we present in a planned workshop. The topics include: the analysis of gaze data on screenshots and videos, the Analytical Tool, and important eye tracking concepts like Area of Interest (AOI) and fixation-derived metrics such as Time to First Fixation, Total Fixations, and Fixation Duration as described in table 1, which are concluded from various references [18][13][12]. During the workshop, experts are asked to use the Analytical Tool freely and perform some tasks in order to assure their understanding and their capabilities of using the tool in the proposed evaluation.

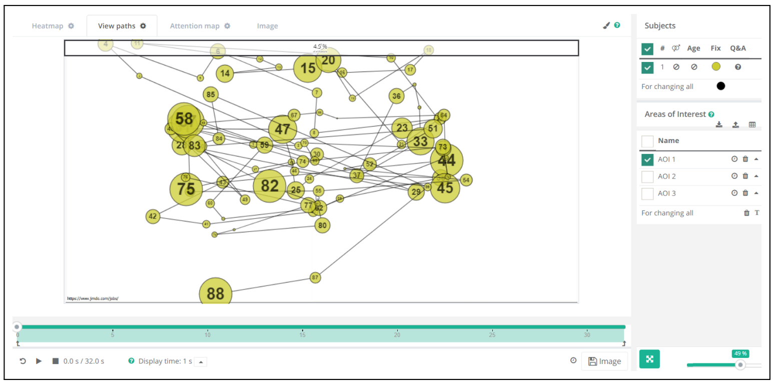


Figure 18: Shows the video visualization in the Analytical Tool, where the overall interactive data of one participant are initially displayed on it. The video starts from the loading time of the Web page till the end of the session. The video and the scan-path of the gaze data is played by pressing the play button of the timeline. The mouse data is also shown in the video frame.

Metric	Definition
Time to First Fixation (TTFF)	indicates the amount of time that a respondent takes to look at a particular Area Of Interest (AOI).
Total Fixations (TF)	indicates the sum of fixations on a single area of interest.
Fixation Duration (FD)	indicates the duration of maintaining the gaze on an area of interest.
Average Duration of Fixations	indicates the average sum of all selected fixation durations within an Area of Interest

Table 1: Eye tracking fixation-derived metrics and their definitions.

Dataset Distribution After the introductory presentation, the experts are asked to analyze the dataset while the evaluator measures their timings in solving the required tasks, see Figure 19 for a visual demonstration. The dataset is distributed among the experts in the following manner: The first group of 5 experts is responsible for studying 2 videos (of each participant on websites X, Y) and 2 screenshots (of all participants on websites Z, W). The second group of 5 experts is responsible for analyzing 2 videos (of each participant on websites Z, W) and 2 screenshots (of all participants on websites X, Y). For more clarification, the distribution is shown in Table 2.

Utilizing this combination allows us to compare the results of each Web page in both groups in the two visualization methods (Enhanced Stitched-Viewport Screenshot and Video) in a fair way without repetition since each expert will observe at least one element (either S or V) of each Web page. Additionally, it eliminates the

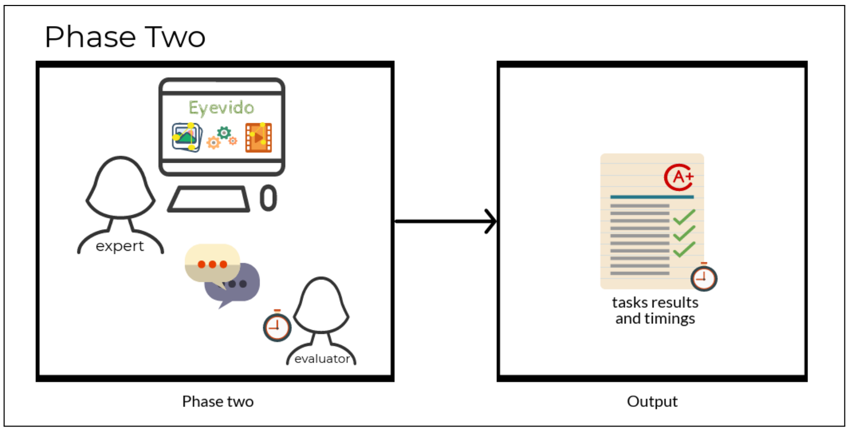


Figure 19: Phase two set-up [left]: Experts analyzing the dataset using the Analytical Tool while the evaluator observes their response timing using a stopwatch. The output [right]: The outcomes of this phase are the statistical answers and reported timings.

Web Page	Group A	Group B
X	Videos(V_{x1}, V_{x2})	Screenshot(S_x)
Y	Screenshot(S_y)	Videos(V_{y1}, V_{y2})
Z	Videos(V_{z1}, V_{z2})	Screenshot(S_z)
W	Screenshot(S_w)	Videos(V_{w1}, V_{w2})

Table 2: The distribution of dataset between the two groups of experts, where each group is responsible to analyze a different kind of element on a Web page than the other group.

possibility of already knowing the solutions of the tasks, which could occur when the expert views a video of a Web page to complete a task and then views the screenshot of the same Web page to run the same task again.

Tasks For The Experts In order to examine the effectiveness, efficiency, and satisfaction of each method, two main types of measures are designed; quantitative and qualitative measures. The ground truth for all quantitative measures was sat in the initial trial. Figure 20 illustrates an infographic of the measures, the tasks, and the leading purpose of each measure.

The quantitative measures include tasks of objective and subjective sorts. The objective ones measure numerical values in relation to the fixation-derived metrics that were mentioned in table 1, whereas the subjective tasks measure the impression of the experts towards the studied participant behavior. These assignments are distributed to identify the effectiveness and efficiency of the tested methods. Furthermore, the fixed element to be analyzed is displayed to the experts before viewing

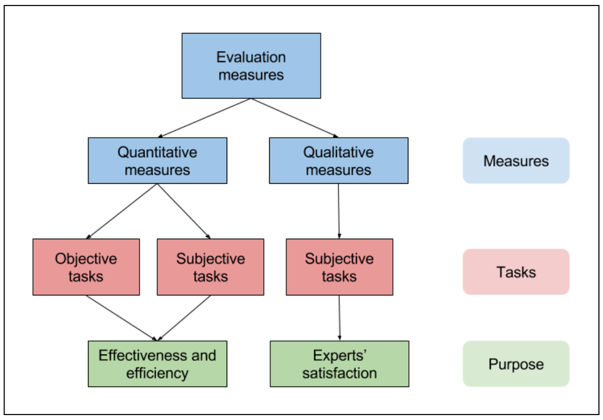


Figure 20: Types of evaluation measures and their assigned tasks each reaching a different value of determination. Quantitative measures justify the efficiency and effectiveness, while qualitative measures verify the satisfaction.

the visualization. When the experts view either of the visualizations, their task is to find the assigned fixed element and draw an AOI to cover it, that will measure the answers of the following questions:

- How long was the time until the subject made their first fixation on the fixed element? (an objective task)
- How many fixations are on the fixed element in total? (an objective task)

Afterwards, the experts are asked to interpret the difficulty level that the participant experienced while completing their task by playing the scan-path of gaze data and mouse trails on the screenshot or the video in order to answer the following question:

- Rate the following sentence: It was challenging for the participants to solve the task. (a subjective task on a scale of "Absolute agreement" to "Absolute disagreement")

The "fixed element" in each question represents one of the fixed elements that were shown in Figure 16 based on the analyzed Web page. For example, on the Creative Commons Web page screenshot or video, the questions are as follows: (for a detailed list of tasks on each Web page please visit Appendix A)

- How long was the time until the subject made their first fixation on the fixed element shown in sub-figure 16b? (an objective task)
- How many fixations are on the fixed element, that is shown in sub-figure 16b, in total? (an objective task)

- Rate the following sentence: It was challenging for the participants to solve the task. (a subjective task on a scale of "Absolute agreement" to "Absolute disagreement")

In the moment when the experts are answering the questions, the evaluator starts recording their response time using the stop-watch for each of the previous objective tasks. The timing starts when the expert displays a visualization and holds when the expert has finished selecting the AOI and found the first statistical result (first fixation), then it continues for the next objective result (total fixation). The timer resets when the expert investigates the second participant. Therefore, the timing includes: noting the fixed element, selecting it with an AOI, and reporting the statistical value.

The qualitative measures are applied using the NASA Task Load Index (TLX) [6] technique. The NASA TLX is utilized to assess the subjective tasks concerning the experts' feedback on the analyzed methods. It provides an overall workload average of ratings based on six sub-scales: mental demand, physical demand, temporal demand, performance, effort, and frustration level. The following scales are presented to the experts for each analyzed method:

Rating Scale Tasks Based on NASA TLX

The following tasks are in regards to the analyzed visualization.
Please choose an accurate scale for each question based on your own experience.

Please answer the following questions:

1. How mentally demanding was the task? (Mental Demand)
   very low □—□—□—□—□—□—□ very high
2. How physically demanding was the task? (Physical Demand)
   very low □—□—□—□—□—□—□ very high
3. How hurried or rushed was the pace of the task? (Temporal Demand)
   very low □—□—□—□—□—□—□ very high
4. How successful were you in accomplishing what you were asked to do? (Performance)
   perfect □—□—□—□—□—□—□ failure
5. How hard did you have to work to accomplish your level of performance? (Effort)
   very low □—□—□—□—□—□—□ very high
6. How insecure, discouraged, irritated, stressed, and annoyed were you? (Frustration)
   very low □—□—□—□—□—□—□ very high

5.1.3. Results analysis

After both evaluation phases are completed, we, as the evaluator, collect and interpret the results of phase two, which represent the answers and response timings to the given tasks. The analysis includes comparisons between the results of each visualization method on each Web page, taking into consideration the ground truth values. Furthermore, we use the T-Test to find the level of significance in the results by analyzing the two (experts groups) populations mean. Our aim is to accept or reject our hypothesis, as well as finding other trend-lines in the results.

5.1.4. Equipments

The following tools are required in order to complete the evaluation:

- An eye tracker (myGaze n [4]): an eye tracking device using a sampling rate of 30Hz and producing the following data: Time-stamp, and gaze data (x/y screen coordinate). (in Phase One 5.1.1).
- A screen recording tool: an open-source third-party screen recording tool is used to record participants' sessions called the Open Broadcaster Software [4] (in Phase One 5.1.1).
- A video editor software: the open-source third-party software OpenShot [5] is used to crop the produced videos in order to get a video clip for each Web page of each user session separately (in Phase One 5.1.1).
- The Recording tool: a gaze recording software provided by the Eyevido Lab, used to collect the dataset from the study participants (in Phase One 5.1.1).
- The Analytical tool: a portal for data visualization provided by the Eyevido Lab, used by the experts to examine the dataset (in Phase Two 5.1.2).
- A stopwatch: used to measure the time that the experts require to solve the tasks (in Phase Two 5.1.2).

5.2. Results

The research problem of this master thesis is the lacking of intelligent linking between the interactive data and their intended fixed elements in static screenshot visualization method, which leaves duplicate images of fixed elements on the screenshot and sparse interactive data along the length of the screenshot. We have achieved an approach to overcome the problem by connecting the interactive data and their fixed elements to gather them in one location on the static screenshot using our Enhanced Stitched-Viewport Screenshot. In the Video approach, the problem does

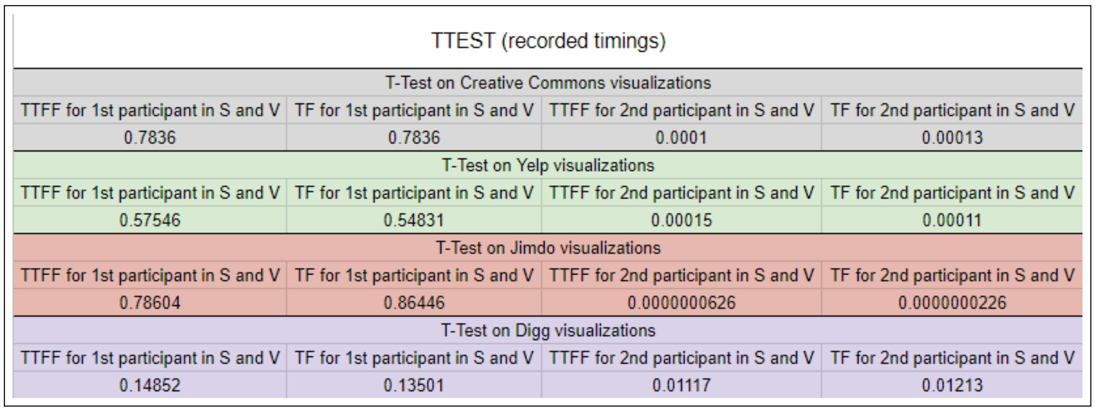
[4]myGaze n `http://www.mygaze.com/products/mygaze-n/`

not occur since the video has the visual of the viewport and the fixed elements are always together with their interactive data.

In the evaluation, we hypothesized that **our Enhanced Stitched-Viewport Screenshot allows faster gaze interpretations for the investigators than the Video visualization approach**. Therefore, we compared both visualization methods (Enhanced Stitched-Viewport Screenshot and Video) by collecting a dataset of interactive data on 4 different Web pages and assigning 10 experts to analyze the dataset in both visualizations. The dataset was gathered from two participants (2 Females) with a mean age of (32) and a standard deviation of (8.48). The 10 studied experts were of (2 Females and 8 Males) with a mean age of (25.4) and a standard deviation of (1.77). The results of the experts' interpretations emerged in quantitative and qualitative forms.

Significance Tests First of all, we have run an independent two-paired T-Test on the response time of the answers of the two experts groups using a threshold for p-value of (0.05), where each group of experts had interpreted one of the visualizations of each Web page. See Figure 21 for detailed statistics. The significance test showed a high difference level among the data of interpreting the second study participant in the dataset, which confirms that the results did not originate by chance and the use of a certain visualization method affects the experts' interpretation speed. For example, the results of the T-Test on the Creative Commons that was visualized as a screenshot in experts group A and as a video in group B show a non-significant p-value of (0.7836) in the response time of finding the Time to First Fixation (TTFF) between the first participant interpretation in group A and the first participant interpretation in group B. However, the results show a very high significant p-value of (0.0001) for the TTFF response time between interpreting the second participant in group A and interpreting the second participant in group B. The significant difference is also realized in the interpretation of Yelp visualization, where it is higher for the second participant interpretation than the first for the TTFF. Also, on the Jimdo visualization, where the TTFF for the second participant is higher than the first and the same for the Digg visualization.

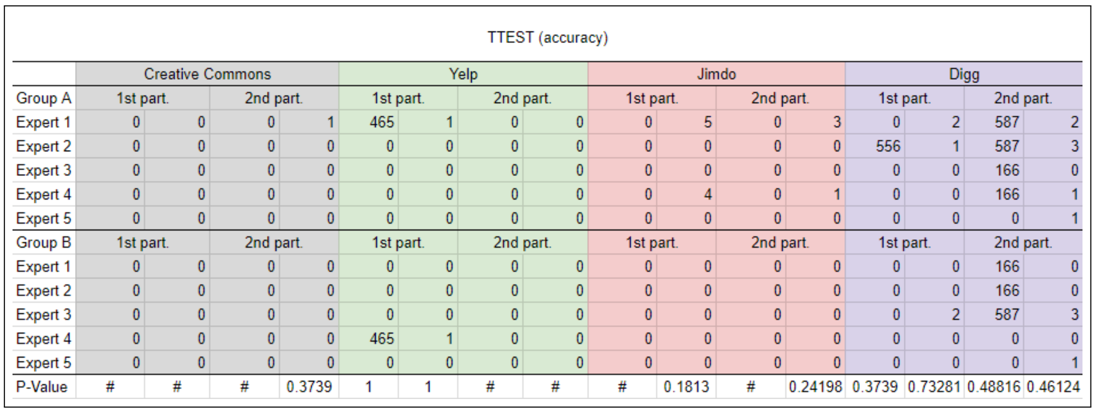
Additionally, we ran a second independent two-paired T-Test on the accuracy of the reported statistical answers from the experts in both groups. We calculated the accuracy by finding the difference between the reported answers and the ground truth values. We also used (0.05) as a threshold for the p-values. The test has revealed that there were no significant difference between the two ranges, which implies that there were no affects on the experts' interpretations using either of the visualizations and that both visualizations recorded the same level of accuracy. Figure 22 shows the calculated differences between the reported values and the ground truth. The last row presents the p-values of each interpretation in group A and group B. For example, the first interpretation of the Creative Commons is exactly the same in both groups, which is noted as a # symbol, and the second interpretation recorded a high p-value of (0.3739).



TTEST (recorded timings)

T-Test on Creative Commons visualizations			
TTFF for 1st participant in S and V	TF for 1st participant in S and V	TTFF for 2nd participant in S and V	TF for 2nd participant in S and V
0.7836	0.7836	0.0001	0.00013
T-Test on Yelp visualizations			
TTFF for 1st participant in S and V	TF for 1st participant in S and V	TTFF for 2nd participant in S and V	TF for 2nd participant in S and V
0.57546	0.54831	0.00015	0.00011
T-Test on Jimdo visualizations			
TTFF for 1st participant in S and V	TF for 1st participant in S and V	TTFF for 2nd participant in S and V	TF for 2nd participant in S and V
0.78604	0.86446	0.0000000626	0.0000000226
T-Test on Digg visualizations			
TTFF for 1st participant in S and V	TF for 1st participant in S and V	TTFF for 2nd participant in S and V	TF for 2nd participant in S and V
0.14852	0.13501	0.01117	0.01213

Figure 21: T-Test results (p-values) of the recorded timings of answering the assigned tasks in both experts groups. Each group had to analyze two participants on both evaluated visualizations on four Web pages. TTFF denotes Time to First Fixation, TF denotes Total Fixations, S denotes screenshot visualization, and V denotes video visualization. The T-Test has the first range of the recorded timings on a screenshot of a Web page and the second range was the recorded timing on the video of the same Web page for each participant.



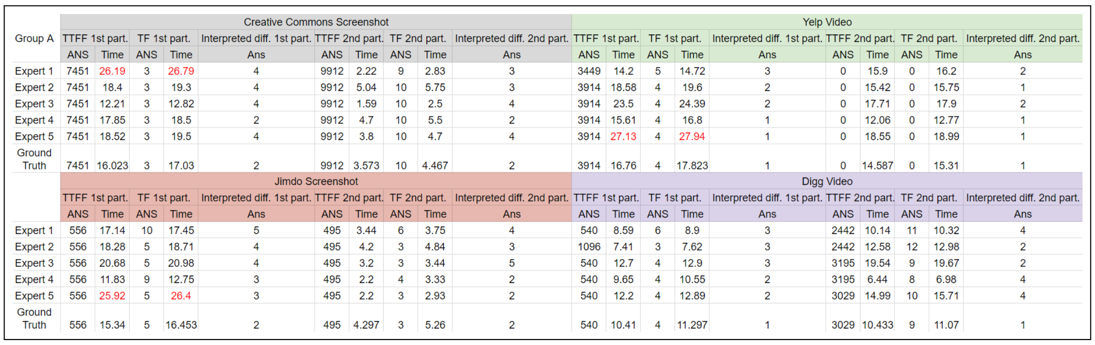
TTEST (accuracy)

	Creative Commons				Yelp				Jimdo				Digg			
Group A	1st part.		2nd part.		1st part.		2nd part.		1st part.		2nd part.		1st part.		2nd part.	
Expert 1	0	0	0	1	465	1	0	0	0	5	0	3	0	2	587	2
Expert 2	0	0	0	0	0	0	0	0	0	0	0	0	556	1	587	3
Expert 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	166	0
Expert 4	0	0	0	0	0	0	0	0	0	4	0	1	0	0	166	1
Expert 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Group B	1st part.		2nd part.		1st part.		2nd part.		1st part.		2nd part.		1st part.		2nd part.	
Expert 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	166	0
Expert 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	166	0
Expert 3	0	0	0	0	0	0	0	0	0	0	0	0	0	2	587	3
Expert 4	0	0	0	0	465	1	0	0	0	0	0	0	0	0	0	0
Expert 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
P-Value	#	#	#	0.3739	1	1	#	#	#	0.1813	#	0.24198	0.3739	0.73281	0.48816	0.46124

Figure 22: Presents the calculated differences between the reported values and the ground truth. The last row presents the p-values of each interpretation in group A and group B. The # symbol implies that both sets were exactly the same.

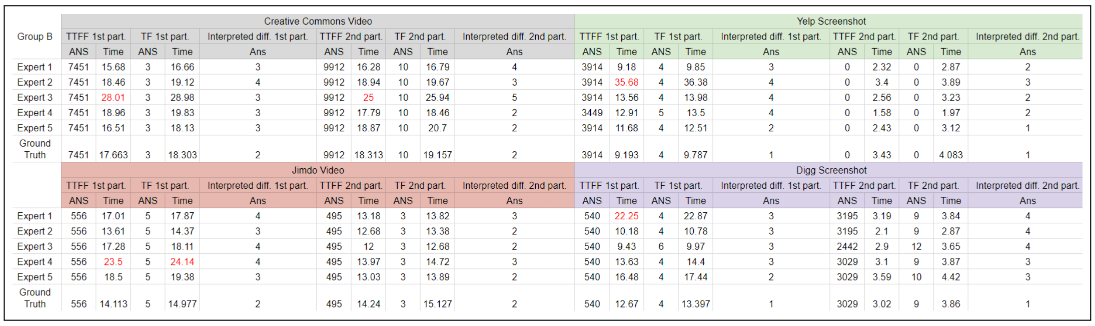
Quantitative Results The quantitative results consist of the experts' statistical answers of the assigned tasks, the recorded timings of response, and the interpretation of the participant difficulty level in completing their tasks on a Likert-scale.

The reported statistical answers, the recorded timings, the difficulty interpretations, and their ground truth values can be seen in sub-figures 23a and 23b. For example, in Group A, the first expert had the Creative Common Screenshot as the first visualization, and as they investigated the first participant, they gave the answer for the Time to First Fixation: (7451ms), recorded (26.19s) before answering, and interpreted that the analyzed study participant had a difficulty of scale 4 (from 1-5, 5 being the highest) in solving their task. The red marked values might be classified as outliers, however, they were taken into account in all comparisons and statistical tests.



	Creative Commons Screenshot										Yelp Video									
Group A	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.
	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans
Expert 1	7451	26.19	3	26.79	4	9912	2.22	9	2.83	3	3449	14.2	5	14.72	3	0	15.9	0	16.2	2
Expert 2	7451	18.4	3	19.3	4	9912	5.04	10	5.75	3	3914	18.58	4	19.6	2	0	15.42	0	15.75	1
Expert 3	7451	12.21	3	12.82	4	9912	1.59	10	2.5	4	3914	23.5	4	24.39	2	0	17.71	0	17.9	2
Expert 4	7451	17.85	3	18.5	2	9912	4.7	10	5.5	2	3914	15.61	4	16.8	1	0	12.06	0	12.77	1
Expert 5	7451	18.52	3	19.5	4	9912	3.8	10	4.7	4	3914	27.13	4	27.94	1	0	18.55	0	18.99	1
Ground Truth	7451	16.023	3	17.03	2	9912	3.573	10	4.467	2	3914	16.76	4	17.823	1	0	14.587	0	15.31	1
	Jimdo Screenshot										Digg Video									
	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.
	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans
Expert 1	556	17.14	10	17.45	5	495	3.44	6	3.75	4	540	8.59	6	8.9	3	2442	10.14	11	10.32	4
Expert 2	556	18.28	5	18.71	4	495	4.2	3	4.84	3	1096	7.41	3	7.62	3	2442	12.58	12	12.98	2
Expert 3	556	20.68	5	20.98	4	495	3.2	3	3.44	5	540	12.7	4	12.9	3	3195	19.54	9	19.67	2
Expert 4	556	11.83	9	12.75	3	495	2.2	4	3.33	2	540	9.65	4	10.55	2	3195	6.44	8	6.98	4
Expert 5	556	25.92	5	26.4	3	495	2.2	3	2.93	2	540	12.2	4	12.89	2	3029	14.99	10	15.71	4
Ground Truth	556	15.34	5	16.453	2	495	4.297	3	5.26	2	540	10.41	4	11.297	1	3029	10.433	9	11.07	1

(a) The quantitative results of group A.



	Creative Commons Video										Yelp Screenshot									
Group B	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.
	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans
Expert 1	7451	15.68	3	16.66	3	9912	16.28	10	16.79	4	3914	9.18	4	9.85	3	0	2.32	0	2.87	2
Expert 2	7451	18.46	3	19.12	4	9912	18.94	10	19.67	3	3914	35.68	4	36.38	4	0	3.4	0	3.89	3
Expert 3	7451	28.01	3	28.98	3	9912	25	10	25.94	5	3914	13.56	4	13.98	4	0	2.56	0	3.23	2
Expert 4	7451	18.96	3	19.83	3	9912	17.79	10	18.46	2	3449	12.91	5	13.5	4	0	1.58	0	1.97	2
Expert 5	7451	16.51	3	18.13	3	9912	18.87	10	20.7	2	3914	11.68	4	12.51	2	0	2.43	0	3.12	1
Ground Truth	7451	17.663	3	18.303	2	9912	18.313	10	19.157	2	3914	9.193	4	9.787	1	0	3.43	0	4.083	1
	Jimdo Video										Digg Screenshot									
	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.	TTFF 1st part.		TF 1st part.		Interpreted diff. 1st part.	TTFF 2nd part.		TF 2nd part.		Interpreted diff. 2nd part.
	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans	ANS	Time	ANS	Time	Ans
Expert 1	556	17.01	5	17.87	4	495	13.18	3	13.82	3	540	22.25	4	22.87	3	3195	3.19	9	3.84	4
Expert 2	556	13.61	5	14.37	3	495	12.68	3	13.38	2	540	10.18	4	10.78	3	3195	2.1	9	2.87	4
Expert 3	556	17.28	5	18.11	4	495	12	3	12.68	2	540	9.43	6	9.97	3	2442	2.9	12	3.65	4
Expert 4	556	23.5	5	24.14	4	495	13.97	3	14.72	3	540	13.63	4	14.4	3	3029	3.1	9	3.87	3
Expert 5	556	18.5	5	19.38	3	495	13.03	3	13.89	2	540	16.48	4	17.44	2	3029	3.59	10	4.42	3
Ground Truth	556	14.113	5	14.977	2	495	14.24	3	15.127	2	540	12.67	4	13.397	1	3029	3.02	9	3.86	1

(b) The quantitative results of group B.

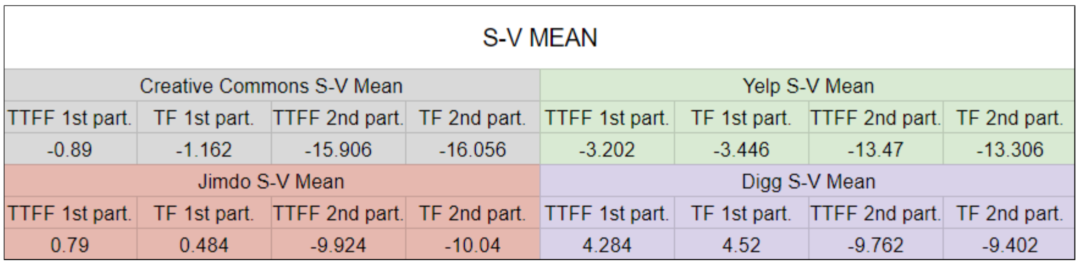
Figure 23: For each visualization in both groups there are answers for the TTFF (Time to first fixation) and TF (Total Fixations) for each participant data, a recorded response time in seconds, and a difficulty interpretation answer on scale from 1-5 (denoted as Interpreted diff. n part.). Every value is presented with its ground truth at the end of the column.

The findings of the quantitative results are threefold:

1. The accuracy of the statistical answers of the experts (Time to First Fixation, Total Fixations) did not show any significant difference in the interpretations of both visualizations since they have the exact same interactive data.

2. Experts were faster in interpreting the second study participant using the Enhanced Stitched-Viewport Screenshot than when using the Video approach. The results have shown an average of (12s) difference between interpreting the second participant using the screenshot and the second participant using

the video visualization. As shown in Figure 24, where the average timings of each visualization (V - Video) were deducted from the other visualization (S - Screenshot) to show the faster method. One may notice that the Enhanced Stitched-Viewport Screenshot is not always faster in the first participant interpretation as it is in the second participant interpretation. For example, in the Digg visualization interpretation, the screenshot took on average 4.284 seconds longer than the video in interpreting the first participant. While the screenshot took on average 0.89 seconds less in interpreting the first participant on the Creative Commons Web page than the video.



S-V MEAN

Creative Commons S-V Mean				Yelp S-V Mean			
TTFF 1st part.	TF 1st part.	TTFF 2nd part.	TF 2nd part.	TTFF 1st part.	TF 1st part.	TTFF 2nd part.	TF 2nd part.
-0.89	-1.162	-15.906	-16.056	-3.202	-3.446	-13.47	-13.306
Jimdo S-V Mean				**Digg S-V Mean**			
TTFF 1st part.	TF 1st part.	TTFF 2nd part.	TF 2nd part.	TTFF 1st part.	TF 1st part.	TTFF 2nd part.	TF 2nd part.
0.79	0.484	-9.924	-10.04	4.284	4.52	-9.762	-9.402

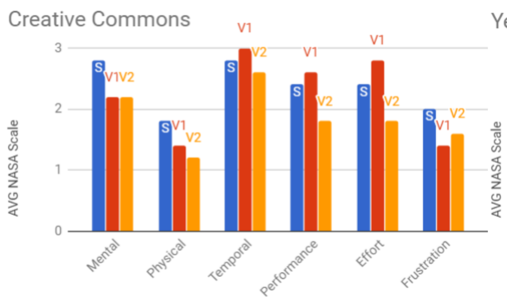
Figure 24: The difference of the mean values of the recorded response timings in both visualizations for the four Web pages. The mean of the recoded timings in the video visualizations is subtracted from the mean of the recoded timings in the screenshot visualizations. The minus mean values reveal how fast were the screenshot interpretations than the videos.

3. Experts were closer to the ground truth values using the Video approach in interpreting the difficulty in the participant behavior when accomplishing their tasks.

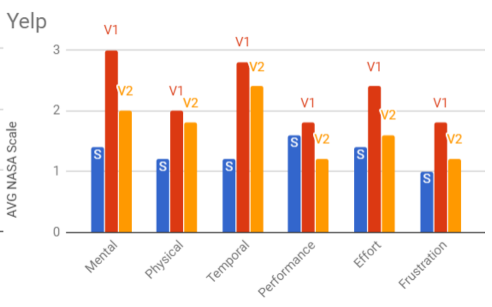
Qualitative Results The qualitative results consist of the NASA TLX survey answers of the experts, that represent their Mental Demand, Physical Demand, Temporal Demand, Performance, Effort, and Frustration of each interpreted method. The results are visualized as column charts in Figure 25 to present the possible trend-lines. The first chart [top-left] shows the experts' survey answers on the Creative Commons visualizations, the second one [top-right] is for the Yelp visualizations, the third [bottom-left] is for Jimdo visualizations, and the fourth [bottom-right] is for Digg visualizations. In every chart, the blue columns represent the experts' responses on the Enhanced Stitched-Viewport Screenshot visualization, the red columns for the first participant Video visualization, and the yellow columns for the second Video visualization. Our findings on the qualitative results are outlined as follows:

- The results have shown a higher temporal demand for each first Video visualization of each Web page than the Enhanced Stitched-Viewport Screenshot. Experts felt more hurried or rushed while viewing the first video element of each Web page than when viewing an Enhanced Stitched-Viewport Screenshot of that same participant. Closer rates were shown on the Creative Commons Web page, however, all other Web pages show a high difference between the temporal rate of the first video and the screenshot. Note the third column set (Temporal) of each sub-figure in Figure 25.

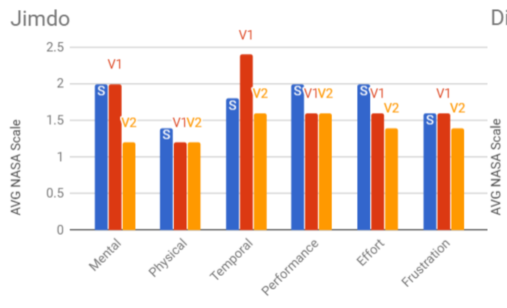
- The second trend-line of the NASA TLX results is that the second-viewed video of each Web page requires lower or equal rates of the workload compared to the first-viewed video. The results show that the Video method requires less physical, mental, temporal demands among with performance and effort from the experts once they have viewed at least the first-participant video.



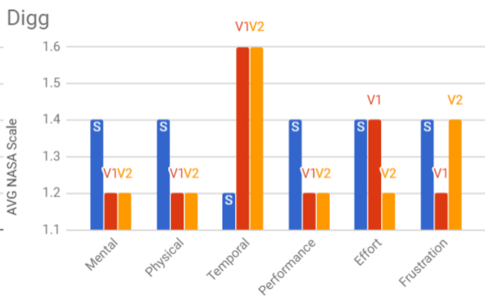
(a) The qualitative NASA TLX results on Creative Commons visualizations.



(b) The qualitative NASA TLX results on Yelp visualizations.



(c) The qualitative NASA TLX results on Jimdo visualizations.



(d) The qualitative NASA TLX results on Digg visualizations.

Figure 25: NASA TLX results of experts' responses on four Web pages in screenshot visualization (blue column), first participant video visualization (red column), and on second participant video visualization (yellow column). The x-axis shows the workload type of NASA TLX, y-axis shows the recorded average scale of the workload by the experts.

The results have shown that both visualizations recorded same levels of accuracy in the reported statistical values. Additionally, the results revealed that the Enhanced Stitched-Viewport Screenshot is more time-efficient than the Video approach since it provided faster interpretations of the second observed participant than the Video approach. However, the Enhanced Stitched-Viewport Screenshot was found to be less effective in interpreting the difficulty of the participant's behavior since it provided less accurate rates of interpretations than the Video approach. The Enhanced Stitched-Viewport Screenshot gave higher interpretation satisfaction and usability rates since it recorded lower temporal demands than the Video approach for all analyses. Nevertheless, the results reported that viewing the second participant video recorded stable or lower rates of the workload after the first participant video. After acquiring these results, we confirm our initial hypothesis and attach a more accurate detail to it: **our Enhanced Stitched-Viewport Screenshot allows faster interpretations for the investigators in analyzing non-initial participants than the Video approach**.

5.3. Discussion

In this section, we will reveal our interpretations of the results and explain possible causes for them. Also, we will theoretically expand our results on bigger datasets, and select suitable visualizations for different scenarios.

The Accuracy The accuracy of the statistical answers was found to be the same in both visualization methods. Each Web page representations (as a video and as a screenshot) had the same interactive data. However, it was noticeable in the reported answers that some experts did not report the exact value but there were no trend-lines in the reported mistakes. The explanation for such mistakes is that some experts failed to accurately draw an AOI on the fixed element. Therefore, the AOI calculated more or fewer fixations and that ended up with a wrong statistical value.

The Timing The Enhanced Stitched-Viewport Screenshot recorded faster timings in interpreting the second participant than the Video visualization. We discuss this result in the following paragraphs.

The visualizations are initially shown in the Analytical Tool fitted in a frame and can be zoomed in or out based on the desire. All analyzed screenshot visualizations were long enough for the participant to require zooming in to locate the analyzed fixed element, which consumed some of the time. For instance, the screenshot interpretation took more time in the Jimdo Web page, which has a header that becomes fixed when the scroll position is more than zero, and Digg Web page, which has an apparent fixed header. In the fitted Video of both Web pages, the headers are obvious enough for the expert to select them with an AOI. However, in the large screenshot of the Web page, the expert had to zoom in carefully to be able to see the headers and select them accurately.

Furthermore, the difference in the presentations of the fixed elements in the four Web pages is also a factor of the recorded time. The interpretation of the first participant in the screenshot took less time in the Creative Commons, which has a fixed element that appears after a certain timeout, and the Yelp, which has an element that appears to be fixed on a certain scroll position. Therefore, it took longer time for these elements to appear in the video session for the experts to select an AOI on them, however, they were visible from the very first sight on the screenshot, and clear enough for the experts to select an AOI on them.

While the interactive data in the Enhanced Stitched-Viewport Screenshot method of each participant can be viewed on a combined screenshot, it is not the case in the Video approach. There is a screen recorded video of each participant session with the interactive data because the videos cannot be combined due to different participants interaction. Therefore, the interpretation of the second participant takes only the time to switch between the interactive data of the participants on the screenshot. However, for the video approach, it requires the expert to view the video of the second participant whilst the expert who is interpreting the screenshot of the same Web page in that time has finished interpreting both participants.

In the Enhanced Stitched-Viewport Screenshot approach, the expert has to mark the AOI on the fixed element only once to analyze all the participant on that Web page. However, in the Video scenario, the expert has to draw an AOI on each video of each participant that they are viewing. If we cloned the AOI from the first video to all other videos, it would not give accurate results since the visibility of the Web elements could change based on the participant interaction.

The Participants' Difficulty Level Interpretation The possible effective difference between the two visualization approaches which could produce the results of the Enhanced Stitched-Viewport Screenshot being less efficient in interpreting the difficulty in the user behavior is that in the Enhanced Stitched-Viewport Screenshot approach, experts see the Web page with all interactive data on it, which creates an overview of the behavior, then they play the gaze scan-path and mouse moves to interpret the user behavior. However, in the Video approach, experts initially observe the video of the Web page with the overall interactive data, they play the video to view the viewport with its dynamics (ads, transitions, etc) with the interactive data, and the scrolling experience. Our understanding of the results reveals that experts were closer to interpret the difficulty in the user behavior observing their scrolling experience along with the interactive data.

Based on our results that we have gathered from 10 experts on a dataset of 2 participants, we discuss the possible results of an expanded dataset with 10 participants. Our evaluation has shown a significant time difference between interpreting the second participant using the Enhanced Stitched-Viewport Screenshot and using the Video approach. Analyzing the 10th participant will not take much time using the Enhanced Stitched-Viewport Screenshot as it will take using the Video approach.

As we have concluded, the Enhanced Stitched-Viewport Screenshot is the faster choice since for only two participants it showed a high timing significance, which means that it will save a lot of time that is consumed while observing videos of 10 participants. Moreover, the Enhanced Stitched-Viewport Screenshot gives higher user satisfaction for the analyst as it showed lower temporal demands in our evaluation. Therefore, if the analysts are looking for a fast method without feeling hurried or rushed, they should choose the Enhanced Stitched-Viewport Screenshot method.

On the other hand, the Video approach has recorded higher rates in understanding the difficulty in the participant behavior. In addition, even though Videos have shown higher temporal rates in the NASA TLX analysis, our results reveal that after observing the first video, the workload for the next video is either stable or lower. Which means that the results could converge to a normal workload rate that would not disturb the expert anymore. Therefore, if the analysts need to focus on the user behavior interpretation and can invest more time and workload in analyzing the visualizations, they should choose the Video approach.

6. Conclusion and Future Work

In this thesis, we addressed the problem of lacking the intelligent linking between gaze data and their intended fixed elements in static screenshot data visualizations. We have combined different kinds of research (eye tracking, Web, and image processing) in order to solve the problem. Initially, we have built up our proposed solution on the naïve Stitched-Viewport Screenshot method, which was first introduced by the Eyevido Lab. With our enhancements, we were able to achieve the expected behavior that puts together gaze data and their intended fixed elements in one position on the screenshot without duplications or sparse data. Later on, we hypothesized that our Enhanced Stitched-Viewport Screenshot allows faster interpretations for the investigators than the Video visualization approach. Using our Enhanced Stitched-Viewport Screenshot in the Recording Tool of the Eyevido Lab, we conducted an evaluation that compares the visualization of our approach against the Video visualization method.

The evaluation was based on experts interpretations of a dataset that consists of interactive data on both visualization methods on four Web pages. The experts were asked to accomplish analysis tasks and report statistical values, meanwhile, their response time was being recorded. The results were of quantitative and qualitative forms. The quantitative statistical results were used to interpret the accuracy of the visualization methods by the experts, which have appeared to be equal in both methods. Furthermore, the response timings have shown that the Enhanced Stitched-Viewport Screenshot recorded faster rates in interpreting the second participant than the Video method. In addition, we have also examined the expert's interpretation of the level of difficulty in the participant behavior in completing their tasks, which has revealed that the Video approach brings higher accuracy in interpreting the difficulty level than the Enhanced Stitched-Viewport Screenshot.

The qualitative results were produced from the NASA TLX survey that was distributed to the experts during the evaluation. The interpretations of the results showed that interpreting video visualization produced higher temporal rates than interpreting screenshot visualization. However, the results pointed out that every second video has a stable or lower temporal rate than the previous one. Which could eventually yield to a normal temporal rate after an n number of videos. Nevertheless, it could happen that by the time the expert is interpreting the nth video, the expert who is interpreting the screenshot of the same Web page is already done by viewing the one and only screenshot with the participants' interactive data.

Further research could be established to gain better insights on how screenshot visualizations were not as close to the ground truth as videos in interpreting the level of difficulty by studying the user behavior. Our research adds better insights on Web structure which in its way improves Web usability. The implementation covers the specific case of fixed elements on scrollable Web pages, however, the theory can be adapted to fit further research on scrollable content in the Web page such as a *<div>* with the CSS property *overflow:scroll* since it shares the same problem that we had on the unlinked gaze data with content. Furthermore, this research is the first step to identify and extract Web elements, which can be used as a lead to further research on detecting other types of technologies that run in the Web page like the dynamic changes of Web elements via asynchronous calls.

In conclusion, we were able to overcome the research problem by introducing our Enhanced Stitched-Viewport Screenshot as a solution. We compared our work with the Video approach. Our results have shown that our Enhanced Stitched-Viewport Screenshot has recorded faster interpretations than the Video approach for the second user analysis. The future work of this research focuses on further research in the path of enhancing the outcome screenshots of our approach by including possible dynamic changes in the Web page, and collecting further explanations on user behavior interpretation in screenshots for higher improvements.

7. Bibliography

All infographics were designed using Piktochart `https://piktochart.com/`.

References

[1] Fireshot. `https://getfireshot.com`. Accessed: 24-08-2017.

[2] Full page screen capture chrome extension. `http://mrcoles.com/full-page-screen-capture-chrome-extension/`. Accessed: 24-08-2017.

[3] The mousetracker browser. `https://github.com/hanadi92/demobrowser`. Accessed: 26-11-2017.

[4] The open broadcaster software. `https://obsproject.com`. Accessed: 02-09-2017.

[5] Openshot video editor. `http://www.openshot.org/`. Accessed: 02-09-2017.

[6] NASA Task Load Index (TLX). Volume 1.0; Paper and Pencil Package. Technical Report, NASA Ames Research Center, Jan 1986.

[7] Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. ISO 9241-11, ISO, March 1998.

[8] *What Do You See When You're Surfing? Using Eye Tracking to Predict Salient Regions of Web Pages*. Association for Computing Machinery, Inc., April 2009.

[9] Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems. ISO 9241-210, ISO, March 2010.

[10] Eyevido crowd eyetracking. `http://eyevido.de`, 2015. Accessed: 02-09-2017.

[11] A. M. Barreto. Do users look at banner ads on facebook? *Journal of Research in Interactive Marketing*, 7(2):119–139, 2013.

[12] Z. Bylinskii. Eye fixation metrics for large scale analysis of information visualizations. 2015.

[13] Z. Bylinskii, M. A. Borkin, N. W. Kim, H. Pfister, and A. Oliva. *Eye Fixation Metrics for Large Scale Evaluation and Comparison of Information Visualizations*, pages 235–255. Springer International Publishing, Cham, 2017.

[14] E. Cutrell and Z. Guan. What are you looking for?: An eye-tracking study of information usage in web search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 407–416, New York, NY, USA, 2007. ACM.

[15] C. Ehmke and S. Wilson. Identifying web usability problems from eye-tracking data. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 1*, BCS-HCI '07, pages 119–128, Swinton, UK, UK, 2007. British Computer Society.

[16] R. Menges, C. Kumar, D. Müller, and K. Sengupta. Gazetheweb: A gaze-controlled web browser. In *Proceedings of the 14th Web for All Conference on The Future of Accessible Work*, W4A '17, pages 25:1–25:2, New York, NY, USA, 2017. ACM.

[17] A. Poole and L. J. Ball. Eye tracking in human-computer interaction and usability research: Current status and future. In *Prospects", Chapter in C. Ghaoui (Ed.): Encyclopedia of Human-Computer Interaction. Pennsylvania: Idea Group, Inc*, 2005.

[18] A. Poole and L. J. Ball. Eye tracking in human-computer interaction and usability research: Current status and future. In *Prospects", Chapter in C. Ghaoui (Ed.): Encyclopedia of Human-Computer Interaction. Pennsylvania: Idea Group, Inc*, 2005.

[19] H. Sano, S. Shiramatsu, T. Ozono, and T. Shintani. Web block extraction system based on client-side imaging for clickable image map. *Journal of Communication and Computer 10*, pages 1–8, 2013.

[20] Y. Tang, J. Shin, and H.-C. Liao. De-ghosting method for image stitching. *International Journal of Digital Content Technology and its Applications*, pages 1–8, 2012.

Appendices

A. Detailed Experts' Tasks

1. Creative Commons Web page tasks:
    - How long was the time until the subject made their first fixation on the fixed element shown in sub-figure 16b? (an objective task)
    - How many fixations are on the fixed element, that is shown in sub-figure 16b, in total? (an objective task)
    - Rate the following sentence: It was challenging for the participants to solve the task. (a subjective task on a scale of "Absolute agreement" to "Absolute disagreement")
2. Yelp Web page tasks:
    - How long was the time until the subject made their first fixation on the fixed element shown in sub-figure 16c? (an objective task)
    - How many fixations are on the fixed element, that is shown in sub-figure 16c, in total? (an objective task)
    - Rate the following sentence: It was challenging for the participants to solve the task. (a subjective task on a scale of "Absolute agreement" to "Absolute disagreement")
3. Digg Web page tasks:
    - How long was the time until the subject made their first fixation on the fixed element shown in sub-figure 16d? (an objective task)
    - How many fixations are on the fixed element, that is shown in sub-figure 16d, in total? (an objective task)
    - Rate the following sentence: It was challenging for the participants to solve the task. (a subjective task on a scale of "Absolute agreement" to "Absolute disagreement")
4. Jimdo Web page tasks:
    - How long was the time until the subject made their first fixation on the fixed element shown in sub-figure 16e? (an objective task)
    - How many fixations are on the fixed element, that is shown in sub-figure 16e, in total? (an objective task)
    - Rate the following sentence: It was challenging for the participants to solve the task. (a subjective task on a scale of "Absolute agreement" to "Absolute disagreement")