Improving User Experience of Eye Tracking-based Interaction: Introspecting and Adapting Interfaces

RAPHAEL MENGES and CHANDAN KUMAR,
University of Koblenz-Landau, Germany
STEFFEN STAAB,
University of Koblenz-Landau, Germany, and University of Southampton, UK

Today's systems for gaze-based computer access through mouse and keyboard emulation incur unnecessary interaction and visual overhead for users, aggravating their user experience. We discuss how knowledge about interface semantics may help to reduce interaction and visual overhead, enhancing user experience. We propose an introspection of Web interfaces that determines the semantics of interaction elements on dynamic Web pages during browsing and adapts them for improved gaze control. We have developed GazeTheWeb, a Web browser that includes these capabilities to enhance the gaze interaction experience. In a summative lab study with 20 participants, GazeTheWeb allowed the participants to accomplish information search and browsing tasks significantly faster than an emulation approach. Additional feasibility tests of GazeTheWeb in lab and home environment showcase its effectiveness in accomplishing daily Web browsing activities and adapting large variety of modern Web pages to suffice the interaction for people with motor impairment.

CCS Concepts: • **Human-centered computing → Empirical studies in accessibility**; *Web-based interaction*; *Accessibility technologies*; • **Information systems** → Browsers.

Additional Key Words and Phrases: Eye tracking, gaze interaction experience, gaze-controlled interface, interface semantics, introspection, gaze-based emulation, Web accessibility, GazeTheWeb.

ACM Reference Format:
Raphael Menges, Chandan Kumar, and Steffen Staab. 2010. Improving User Experience of Eye Tracking-based Interaction: Introspecting and Adapting Interfaces. *ACM Trans. Comput.-Hum. Interact.* 9, 4, Article 39 (March 2010), 45 pages. https://doi.org/0000001.0000001

GazeTheWeb was first demonstrated at the 14th International Web for All conference 2017 (W4A' 17) [63], where it won the TPG challenge award for its usability and impact in the field of Web accessibility. Furthermore, we have received the honourable mention at The Web Conference 2017 (WWW' 17) [45] for an early prototype of the Web interface semantics extraction method used in GazeTheWeb. A detailed statement describing our previous activity relevant to this submission is available in Appendix C.

Authors' addresses: R. Menges and C. Kumar, University of Koblenz-Landau, Institute for Web Science and Technologies, Germany; emails: {raphaelmenges, kumar}@uni-koblenz.de; S. Staab, University of Koblenz-Landau, Institute for Web Science and Technologies, Germany; University of Southampton, Web and Internet Science Research Group, United Kingdom; email: staab@uni-koblenz.de.
Authors' addresses: Raphael Menges; Chandan Kumar,
University of Koblenz-Landau, Germany; Steffen Staab,
University of Koblenz-Landau, Germany, and University of Southampton, UK.

1073-0516/2010/3-ART39 $15.00
https://doi.org/0000001.0000001

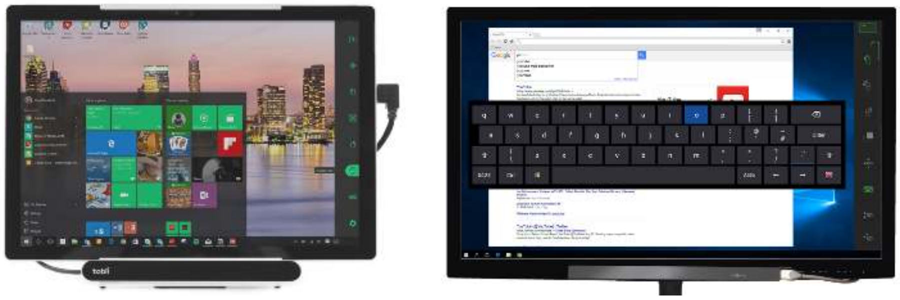
1 INTRODUCTION

The application of eye tracking can break interaction barriers and improve the quality of life for people with a disability that is limiting their interaction through traditional input devices in the digital world. Remote eye tracking systems establish a non-intrusive communication channel between a user and a computer. The systems employ a camera and near-infrared illumination to track the eyes of a user. A calibration process provides a mapping between the recorded image of the eyes and the fixated screen region, which is used to estimate the gaze x-y-position of a user on the screen. The gaze estimation can be interpreted as device control commands [39], so an application can be *gaze-controlled* by a user [65]. Thus, digital accessibility for users with reduced mobility can be improved, and gaze control may also assist users in daily life situations where they cannot use their hands [61]. However, most graphical user interfaces are not designed to be controlled with eye tracking devices, which provide gaze data with limited accuracy. Furthermore, gaze control requires unconventional selection techniques [51] to distinguish between perception and inspection behavior and a selection intention of a user.

A common method to employ eye gaze as a communication channel for controlling application interfaces is to emulate mouse and keyboard devices through an additional command-translation layer. The command-translation layer is implemented as a graphical panel, which is sensing for fixations on *virtual buttons* that exceed a certain *dwell time* to emit mouse and keyboard events. The gaze signals are filtered to improve precision, and pointing within the interface of the controlled application is based on magnification to compensate for accuracy limitations. This allows for an *indirect* control of application interfaces through eye gaze via a command-translation layer. The approach has existed for several years, e.g., the Eye-gaze Response Interface Computer Aid (ERICA) system [54] was incepted in 1983 and included gaze-control mechanisms, featuring mouse and keyboard emulation at operating system level. More recent work on gaze-based computer access uses the same approach with integrating mouse and keyboard functions [84]. Today's commercial systems like Tobii Dynavox Windows Control[1] or Visual Interaction myGaze Power[2] incorporate similar mechanisms. See Figure 1 for examples of commercial systems as sold by companies. Recently, Microsoft has even integrated gaze-controlled mouse and keyboard emulation into the Windows 10 operating system.[3] Moreover, there exist open source alternatives, like OptiKey [80], which work effectively with affordable eye tracking devices available in the market.

While the application agnostic construction of the emulation approach works with a broad variety of graphical interfaces, the command-translation layer is unaware of interaction element types in an application interface and data structures internal in an application. The users first must imagine how to control an interface with mouse and keyboard and then perform the emulation of these events via the gaze-controlled command-translation layer. Moreover, the users need to continuously switch attention between the emulation panel and the interface of the application to accomplish a task. This disrupts the user experience, violating most important aspects of user-centered design to provide users with natural means of interaction. To counter these issues, we argue that the adaptation of an application interface for gaze-based interaction may yield better performance, improved usability, and reduced workload for end users. We propose an adaptation of interfaces for gaze-based interaction that minimizes the count of input actions and the inspection overhead. We achieve the adaptation through knowledge about the *interface semantics*, such as the position, size, properties, state, and interaction means of the elements in an application interface.

[1]https://www.tobiidynavox.com/software/windows-software/windows-control
[2]http://www.mygaze.com/products/assistive-products/mygaze-power
[3]https://support.microsoft.com/en-us/help/4043921/windows-10-get-started-eye-control



(a) Tobii Windows Control [24] (b) Visual Interaction myGaze Power [36]

Fig. 1. Eye tracking systems for computer access as sold by Tobii and Visual Interaction. The remote eye tracking device is mounted below the computer screen and provides gaze signals to an emulation software on the computer. The softwares of both companies share the concept of indirect control over the common desktop and application interfaces via a command-translation layer. A user is presented an overlaying panel, which is placed in both systems on the right side of the screen space. By selecting virtual buttons in these panels through a fixation, a user can choose how their gaze is translated to mouse or keyboard events.

To retrieve the semantics, *introspection* of the interface allows us to track objects their properties such as type, location, and status within a known system.

We specifically propose the interface introspection and adaptation for eye gaze input in the Web environment. A Web browser provides a common platform for various Web-based applications like messaging, social media, entertainment, or even office suites. The underlying technologies of these applications are standardized to Hypertext Markup Language (HTML) for content structure, Cascading Style Sheets (CSS) for layout and style, and JavaScript for interaction and behavior. It is germane to these Web standards that the structure of the interface is explicit and accessible by third party software. Hence, the Web environment allows us to introspect the elements of a Web-based interface, to retrieve the interface semantics, and to adapt the interaction for the purposes of gaze control. Towards this goal, a primary challenge is to retrieve, classify, and track the interaction elements on Web page interfaces, like hyperlinks or text input, in a dynamic Web environment. We describe an effective method of utilizing the Web engine of a Web browser to retrieve the interface semantics of interaction elements and their dynamics during a user session and develop sophisticated interactions for gaze-adapted browsing operations. We demonstrate the applicability of our approach through GazeTheWeb,[4] a Web browser with *direct* gaze control, as the most prominent browsing functions are supported through interpretation of eye gaze with regard to the interface semantics. To assess the user experience, we have evaluated Web browsing in GazeTheWeb for its usability, workload, and overall task completion time, comparing it with the traditional emulation approach. Furthermore, we report about feasibility studies to demonstrate GazeTheWeb capability in handling complex and dynamic Web pages, and supporting everyday browsing tasks of users.

The following points summarize our contribution:

- **User experience perspective for eye gaze input.** Current research in eye tracking mostly focuses on how eye tracking can be a substitute for conventional input means like mouse and keyboard, and, hence, the research focuses on atomic interactions for better pointing and

[4]https://github.com/MAMEM/GazeTheWeb

typing operations. We argue that emulation of mouse and keyboard behavior does not cater to a wholesome experience for users and there is a need to move beyond the concept of eye tracking being a useful input method, to being usable in everyday interaction with interfaces. Thus, we call the user experience for gaze-based interaction the *gaze interaction experience*.

- **Introspection to adapt for gaze-based interaction.** Introspection methods have often been used to adapt interfaces for user context and input modality. In this work, we use introspection to retrieve the interface semantics and to adapt interaction for eye tracking as input method, which has not been explored before. We call such an interface then a *gaze-adapted* interface. Furthermore, we describe an interface introspection methodology to efficiently retrieve the interface semantics in a dynamic Web environment.
- **GazeTheWeb as a tool to enhance Web accessibility.** Access to the Web is a major issue for people with motor impairment, where eye tracking systems do already play an important role for digital accessibility. However, no prior system offers sophisticated Web access through direct gaze-based interaction. In that regard, GazeTheWeb browser contributes as a substantial tool to enhance Web accessibility, i.e., to support the common browsing functionalities by making them accessible with eye tracking as input method.
- **Evaluation methodology of gaze-controlled interfaces.** Most current approaches focus on evaluating the speed and accuracy for isolated interactions with other modalities like mouse, keyboard, or touch. We propose a comprehensive user experience evaluation by comparing gaze-controlled interfaces using task completion, workload, design assessment, and explicit feedback – and to assess the feasibility of the interface adaptations among people with motor impairment using task satisfaction and long term usage statistics.

2 RELATED WORK

In Section 2.1, we first discuss the state-of-the-art use of eye tracking as an input method for computer access. We explore the current focus of research to utilize eye tracking for conventional interactions of pointing and typing, and how these interactions are combined to provide mouse and keyboard emulation. In this work, we propose to use knowledge about interface semantics to improve the user experience with eye gaze input. Thus, we discuss the role of interface introspection to retrieve the interface semantics and how it is used to integrate input methods into existing applications in Section 2.2. Apart from our scientific contribution, we apply our propositions to improve the gaze interaction experience in a gaze-adapted Web environment, which enables people with motor impairment to access the Web. Hence, in Section 2.3, prominent Web browser applications are described that comparably contribute to the field of Web accessibility.

2.1 Eye Tracking as an Input Method

Remote eye tracking technology determines the direction of the gaze of a user in relation to a screen. The motivation to investigate eye tracking as a hands-free input method is pertinent, as gaze control may be a significant addition to the lives of people with motor disabilities, which hinders their use of mouse and keyboard. With this motivation in mind, so far research in gaze-based interaction has focused on pointing and on typing.

2.1.1 Gaze-based Pointing and Typing. The characteristics of eye gaze as "what you look at is what you get" [39] had initiated the initial comparisons of eye gaze with other pointing mechanisms like mouse, pen, or touch. The performance of gaze-based pointing is assessed with Fitts' law [57], to evaluate throughput and accuracy. Furthermore, the usability of eye tracking as input method has been assessed as per device comfort ISO 9241-9 questionnaire [85], which is primarily used for traditional input devices like mouse, pen, or joystick. In general, the performance of eye gaze input is found to be lower than conventional pointing mechanisms [58]. Therefore, various methods of eye pointing, e.g., multi-step magnification [6], fish-eye lens magnification [5], or smooth pursuit of visual targets [71] have been evaluated and compared by means of time required for pointing and achieved accuracy. Similarly, eye gaze has also been considered as a substitute for a physical keyboard, to allow for text entry. There are several eye typing methods, e.g., AugKey [21], EyeSwipe [52], or GazeTheKey [74]. The performance of the eye typing methods has been evaluated with the metrics of words per minute, keystrokes, and error rate [75].

2.1.2 *Emulation of Mouse and Keyboard.*

Methods for better pointing and typing with eye gaze input are an imperative aspect of gaze interaction research. However, users do not use these methods in isolation and the acceptance of technology primarily depends on how these atomic interactions let users interact with applications through their interfaces as a whole. The current approaches for operating the computer with eye tracking follow a straightforward adoption of pointing and typing techniques, emulating mouse and keyboard devices to interact with the applications through their traditional graphical interfaces [54, 84]. Moreover, the usability of emulation approaches has only been assessed based on the accomplishment of atomic interactions or the subjective assessment of an eye tracking input method against other input means.

However, it is rather interesting to investigate how gaze-based interaction impacts the user experience in terms of both objective user performance in task execution and subjective usability and workload impression. There is a need to move beyond the concept of eye tracking being a useful input method, to being usable in daily use of applications. This would lead to eye tracking being more acceptable as an interaction technology for end users. Hence, in this work we go beyond the interactions to mimic mouse and keyboard and aim for seamless integration of eye gaze input in applications. We argue that introspection and adaptation of an interface plays a major role towards this goal, which is discussed in the following.

2.2 Interface Introspection and Adaptation

Introspection is a mean to observe, monitor, and reflect. It plays a key role in many different fields such as psychology [44], sociology [14], and computer science [4]. In computer science, the term *introspection* relates to the tracking of objects and their properties such as type, location, and status within a known system. In software engineering, introspection of programming components [56] allows for adapting modules to achieve more robust behavior, such as fault tolerance [13, 78]. In human-computer interaction, introspection of interface objects permits adaptation of interaction elements to enhance the experience of users [79]. There are various approaches to introspect interface elements for a personalized and context-aware adaptation of interfaces, based on the individual user profiles and interests [3, 28]. In this work we focus on how interface introspection can help adapting interaction for the specific input method of eye tracking. Therefore, we discuss the related approaches of adapting interaction using interface introspection.

2.2.1 *Adapting Interaction using Introspection.*

There are numerous approaches that adapt interfaces to account for varying means of input by utilizing knowledge about the interface semantics. Considering the ubiquity of mouse pointing in computer applications, researchers have developed techniques to improve pointing performance of users by considering the interface semantics. Many interfaces resize pointing targets to enhance spatial accuracy of pointing. The Bubble Cursor [26, 30] employs interface introspection to dynamically resize the activation area allowing the cursor to snap to the nearest target. Blanch et al. [10] decouple visual space and motor space of an interface to improve pointing performance. They adapt the pointer speed to the underlying interface elements to make it easier to select plausible options, make it more difficult to select less plausible options, and to scale the interface elements in visual space according to the information they convey.

Besides the improvement of pointing performance, there have been approaches to adapt general interface characteristics for input devices. Wang and Mankoff [81] provide support for arbitrary mappings between input methods. They describe an abstract layer between input and applications and define a mathematical model to map between different input methods according to their information throughput. The mappings are not inherently context-aware, as they do not integrate a retrieval of the underlying interface semantics. In contrast, the XWeb project [69] provides dynamic adaptation for a variety of applications and works across several different input devices including mouse and keyboard, pen, laser pointer, and speech. For this purpose, XWeb proposes an interface specification language and the adaptation for input devices therefore only works with interfaces written using the XWeb language. Analogously, Carter et al. [15] propose a tool that automatically modifies desktop applications whose interfaces are based on the Java Swing toolkit to accommodate a variety of input devices, like pointers, keyboards, switches, or speech. For example, for keyboard-only input, they replace a list with an interaction element that suggests available list items, based on the entered text. For switch input, they replace a text field with an interaction element that allows a user to select characters one at a time. They suggest a lookup table to augment or replace interaction elements according to the available input devices with interaction elements offering the more suitable interaction. Conceptually, they divide the user interaction into navigation actions – to select an interaction element among the available ones – and control actions – to select an option within of the interaction element – and then adapt both independently.

The discussed approaches expect that users can perceive the interface and inspect options with their eyes while using an input device (mouse, keyboard, touch, or switch) in parallel. The approaches assume that users can scan the available elements and options, and that users observe the effect of an input method while the interaction takes place. However, this assumption is not valid for gaze-based interaction, because the eyes of a user are performing a double duty [58]. The eyes of a user are overloaded with both perception or inspection and selection of an option. Improving the user experience for gaze interaction requires consideration of specific characteristics and challenges of eye tracking. However, none of the above-mentioned approaches investigates how introspection could help overcome these challenges. In this work, we aim to reduce the interaction and visual overhead for eye gaze input and propose how the interface semantics of existing application interfaces could help achieving this. The proposed gaze interaction improvements are applicable to all desktop and Web application interfaces. However, the realization depends on the effective retrieval of the interface semantics. The interface semantics could be retrieved using various introspection methods, and we discuss prominent methods in the next section.

2.2.2 Methods for Interface Introspection. For desktop applications, there are several ways to obtain element-related information and to interact with interaction elements from dedicated accessibility APIs [18, 79]. Apple has defined a universal access APIs for its Carbon and Cocoa toolkits.[5] Similarly, Microsoft provides the Active Accessibility[6] and System.Windows.Automation frameworks, and the X Window System offers an Assistive Technology Service Provider Interface (AT-SPI), which is supported by GTK+, Java/Swing, the Mozilla suite, StarOffice/OpenOffice.org, Qt, and provides a toolkit-neutral way for accessibility services. Most of these APIs can query attributes like position, size, type, and state of interaction elements in an interface and allow for interaction with the exposed elements. Furthermore, there are generic approaches to retrieve interface semantics from the visual appearance of an interface by interpreting the pixels on the screen [22, 23].

For the Web environment, retrieval, classification, and tracking of interaction elements is the basis that allows us in GazeTheWeb to adapt the interaction for eye gaze input. Static approaches

[5]https://developer.apple.com/documentation/applicationservices/carbon_accessibility
[6]https://www.microsoft.com/en-us/accessibility

of parsing the initial HTML structure of a Web page are not sufficient, as the adaptation needs to be revised as the Web page interface may change dynamically its content, layout, and visibility of elements. Thus, it is necessary to let the Web browser extract the interaction relevant elements of a Web page and observe changes in their properties. However, the introspection of elements in combination with the dynamics of Web pages are not yet reflected in Web page data extraction research, which is mostly focused on pure content extraction [16, 83]. Recent work describes methods that allow for observation of preselected, dynamic elements [12, 53]. In this work, we present a novel approach of utilizing the Web engine to retrieve, classify, and track elements on a Web page to gather the interface semantics and to propose the improvements of interaction in the Web environment for eye tracking as input method.

2.3 Enhancing Web Accessibility with Gaze-based Interaction

One of the most common and most useful applications that today's computer users access is the World Wide Web. Access to the Web is limited for people with various kinds of disabilities, which might be caused by disease, accident, or due to aging [34]. Although standards are a noteworthy component of the overall effort towards universal accessibility, for example the W3C Web Accessibility Initiative,[7] the reality is that the majority of Web pages are not developed with accessibility in mind. The current state of accessibility support on the World Wide Web requires tools that can handle Web interfaces, regardless of structure or adherence to accessibility standards.

2.3.1 Accessible Web Browsing. Various approaches and systems have been proposed to enhance Web accessibility. The concept of non-visual browsers has been studied for the benefit of people with visual impairment. CSurf [59] is a non-visual browser, which extracts relevant information from a Web page and outputs it using an audio screen reader. Jensen and Øvad [40] propose to embed a sign language dictionary in a browser to optimize Web accessibility for people with hearing impairment. Firefixia [20] is an accessibility Web browser toolbar that adapts the presentation of Web content to support people with dyslexia. The Web Trek [19] browser utilizes multimedia, like an image tile-based search engine, to provide Web access for people with cognitive disabilities.

One population of users for whom the Web is especially important is those with motor disabilities, because an accessible Web may enable them to do things that they might not otherwise be able to do: shopping, getting an education, or running a business. In that regard, speech-enabled browsing [2] or software like HandsFreeChrome[8] might provide people with motor impairment an option to interact with the Web using voice input commands. However, speech input suffers from recognition and privacy issues. Moreover, the majority of people with motor impairments, including cerebral palsy, Amyotrophic Lateral Sclerosis (ALS), brain stem strokes, and certain spinal-cord injuries, have also impaired speech, leaving fewer options for communication. In this regard, a single switch, albeit being a low bandwidth input device, is a widely used control device due to its simplicity and economy. There have been several browser extensions and approaches [32, 62, 77] to support Web navigation through single switch input. These approaches either simulate the "tabbing" action on a keyboard and scan through interaction elements at a fixed time interval, or they base on extraction of all hyperlinks to create a separated list of the hyperlinks to support linear or incremental search. There have also been Web browsers controlled by brain-computer interfaces for people who are completely paralyzed [8, 42], however, the interaction is limited, extremely slow and error prone.

2.3.2 Gaze-controlled Web Browsing. Eye tracking technology has emerged as a non-intrusive way of interaction [39] and can provide an efficient modality to enhance Web accessibility for people

[7] http://www.w3.org/WAI
[8] https://www.handsfreechrome.com

with motor impairment. For some users, eye gaze might be the most suitable channel for interaction. For example, for people at the advanced stages of ALS, eye muscles are often among the last to deteriorate. We identify two directions to integrate eye tracking and the Web environment:

Bringing Eye Tracking into the Web Environment. One prominent approach is to make eye gaze available in the Web environment to enrich the interaction means and augment the experience of users. The Text 2.0 framework [9] by Biedert et al. describes a browser plugin that allows gaze-responsive Web pages, enhancing the traditional mouse-based interaction. The gaze-based interaction can be implemented by Web developers through a novel set of event handlers for Web page elements, e.g., *onFixation*, *onGazeOver*, and *onRead*. Wassermann et al. [82] build upon this concept and propose a browser-independent framework to bring eye tracking into the Web environment. They propose a native client to communicate with the proprietary eye tracker interfaces. The Web environment receives generic eye gaze data from the client via the Web socket standard and spawns corresponding events on elements of the Web page, utilizing the widespread jQuery library. The system allows using various eye tracking hardware to capture eye gaze and the interpretation on a Web page within any Web browser that implements modern Web standards. Wassermann et al. have shown the practicality of their approach with the integration of eye gaze in an online eLearning platform. The inclusion of gaze data allows the eLearning platform to provide a user meaningful feedback on the relationships between different elements, to provide hints about information a user may have missed and to allow for integrated psychological experiments. Although, it is a pertinent guideline for the Web developers to include gaze interactions in their application, the proposed mechanisms do not resolve the problem of browsing the current Web with gaze-based interactions. Hence, there is a need of introspection methodology to identify existing interaction elements automatically, so that the interaction can be adapted for eye gaze input.

Bringing the Web into the Eye Tracking Environment. To enhance Web accessibility with eye tracking, there have been very few basic approaches to allow gaze-based interaction for certain actions like hyperlink navigation and scrolling. Abe et al. [1] demonstrated a custom eye tracking environment to control a limited number of Web browser functionalities. WeyeB [70] was a browser prototype that implemented basic hyperlink navigation via eye gestures and scrolling through virtual buttons. Lutteroth et al. [55] proposed a system for hyperlink navigation based on the knowledge about the position and size of hyperlinks on Web pages. They augmented each hyperlink with a color, which is unique within a certain screen region. A user could first look at the hyperlinks on a Web page and then confirm selection of a certain hyperlink by fixating on a corresponding color indicator at the side of the Web page viewport. All these systems did not investigate the introspection of complex interaction elements in the Web environment like text input, select fields, or videos, and their adaptation for effective gaze-based interaction. Moreover, their limited adaptation for eye gaze input works only for static Web pages, as they initially parse the page for elements of interest. Hence, the systems lack the design and functionality to work for the modern Web environment with dynamic Web pages.

The related work shows that eye gaze as digital communication channel is primarily used to emulate the behavior of mouse and keyboard input devices. However, it is imperative to explore how gaze-based interaction can follow the principles of user-centered design to provide better user experience. Thus, we argue for the need to retrieve the semantics of an interface to be controlled with eye gaze to go beyond the scope of the context-unaware emulation approach. There are parallel approaches of introspection to use interface semantics for input adaptation, however, to the best of our knowledge, sophisticated adaptation for eye tracking as input method has not yet been explored. We have discussed different ways to perform introspection of interfaces and argue that the Web environment offers a promising opportunity for the retrieval of interface semantics on many Web page interfaces, to offer user-centered gaze control for a wide range of applications.

3 IMPROVING GAZE INTERACTION EXPERIENCE

In this section, we discuss the challenges of using eye tracking as input method, describe the emulation approach by example, analyze the shortcomings of the emulation approach, and put up our propositions for improving the gaze interaction experience. More specifically, the challenges in Section 3.1 describe the error in estimation of eye gaze and in which way filtering is applied to compensate for the error in estimation of eye gaze. Furthermore, the dual role of the eyes as being used for both inspection and selection in gaze-based interaction is discussed. Then, we show by example how gaze-based interaction is implemented for the emulation approach in Section 3.2. Finally, we describe how the emulation approach deviates from user-centered design principles, and we discuss our propositions for a better gaze interaction experience in Section 3.3.

3.1 Challenges of Eye Gaze Input

In gaze-based interaction, the eye tracking device captures a user's eye gaze positions and movements, and interprets them as control commands (as discussed in Section 2.1.1). Hence, challenges arise from estimating eye gaze signals precisely and accurately, as well as from disambiguating eye gaze recordings into intended user interface control commands.

3.1.1 Precision and Accuracy of Eye Gaze Estimation. Several factors influence eye gaze estimation of remote eye tracking systems. Users may move their head, change their angle of gaze in relation to the tracking device, or wear visual aids that distort the recorded geometry of the eyes. Further parameters could influence the capture of the eye gaze, such as ambient lighting, the sensor resolution of the utilized camera, and the calibration algorithm [33]. During eye gaze tracking, these factors result in an error that is modeled as *precision* (also: variance, gaze jitter) and *accuracy* (also: bias, gaze drift) [25]. The higher the precision, the less spatially distributed are the gaze samples of a fixation. Filtering aggregates multiple successive samples to an estimated center of fixation. The accuracy describes the distance of the estimated from the true center of fixation by a user. The body of work in *signal processing* includes techniques to filter and smooth the eye gaze input for improved precision [25, 33, 49]. Research in the field of *design and interaction* has proposed adaptive designs like enlarged and screen-centered interface elements or visual feedback [7, 46], to compensate for accuracy issues when it is crucial to determine the attention on a certain element of an interface that is sized below the accuracy limit of an eye tracking device.

3.1.2 Dual Role of Eyes. Eyes play a major role in our daily lives by letting us inspect the surrounding environment to perceive and understand the objects in the environment and guide actions we want to perform. In human-computer interaction, eyes are used in their natural role to inspect the interface, i.e., to perceive the content information (e.g., reading text), or to guide the interaction when performing *input actions*, i.e., interact with a specific interface element to achieve the associated functionality. In conventional interaction, input actions are effectively achieved with selections using motor responses through an explicit interaction channel (e.g., mouse, keyboard, or touch), and eyes are used in parallel to guide the control commands, by inspecting the interface (e.g., examine which element to select and observe the selection response).

However, in gaze-based interaction, eyes are the only modality of interaction, i.e., they are additionally used to provide motor responses for desired input actions [58] and, hence, the parallelism between inspection and selection cannot be achieved. Therefore, distinguishing between the user intentions of inspection and selection, termed the Midas Touch problem [39], becomes the major challenge for gaze-controlled interaction. Furthermore, the dual role overloads the visual channel, causing high cognitive load on the users. More specifically, to perform any input action, users sequentially need to inspect the interface to choose among the interactive elements (*pre-select inspection*), perform gaze-based confirmation (e.g., through dwell time, blink, or eye gesture) [38] onto the desired element (*selection*), and inspect the effect of the selection (*post-select inspection*).

3.2 Limitations of the Emulation Approach

Methods for better pointing and typing have been an imperative aspect of gaze interaction research (as discussed in Section 2.1.2). Today's systems for gaze-based computer access incorporate these methods via an emulation approach that emits mouse and keyboard events upon gaze-based selections. The emulation approach employs the discussed filtering and magnification methods to compensate for precision and accuracy limitation of the eye gaze estimation of an eye tracking device, and utilizes dwell time to distinguish between inspection and selection. In the following, we describe the emulation approach by example and discuss the shortcomings in the user experience.

3.2.1 Emulation Approach. We describe the emulation approach using the example of controlling a Web browser application, as visualized in Figure 2: The remote eye tracking device captures the gaze direction of the user *(1)* and transfers the observations as gaze data to the emulation software. Real-time filtering is applied on the gaze data to compensate for issues with precision *(2)*. An emulation panel is rendered to an extra window on the screen *(3)*. The emulation panel is presented to the user *(4)* besides the window of the Web browser application interface. The virtual buttons in the panel are grouped by the input device they are able to emulate, as for instance OptiKey [80] offers one panel for mouse emulation and another panel for keyboard emulation. Because the screen space is limited, it is not feasible to put all available emulation functionalities associated to individual virtual buttons, which are sized to account for the limited accuracy, into a single emulation panel or to display all emulation panels at once. Switching between the emulation panels is achieved through a separate virtual button selection. Selections of virtual buttons in the panels are translated to mouse events in the mouse emulation panel or to keyboard events in the keyboard emulation panel *(5)*. The Web browser receives these events as if they were originating from a physical hardware device and the application acts like controlled

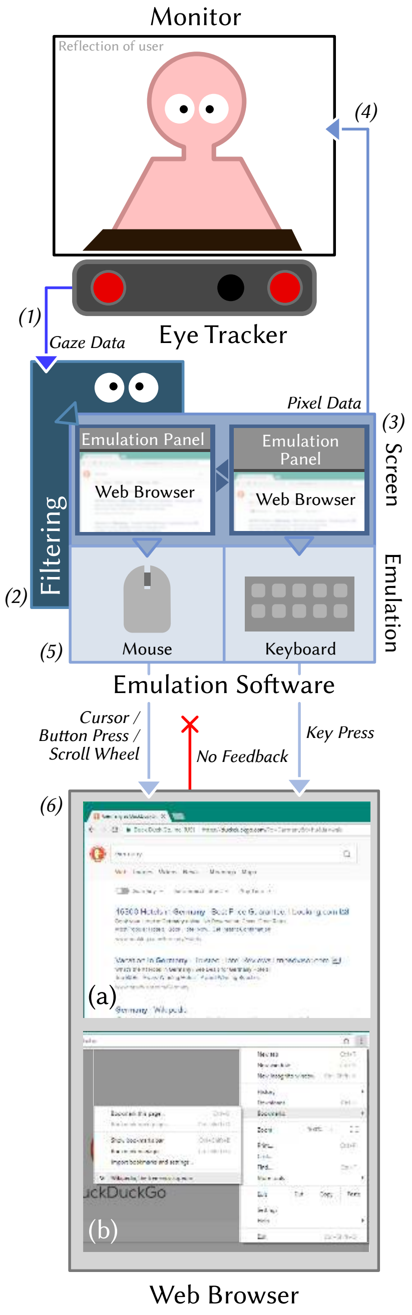


Fig. 2. Emulation of mouse and keyboard to browse the Web with a desktop Web browser through indirect gaze-control.

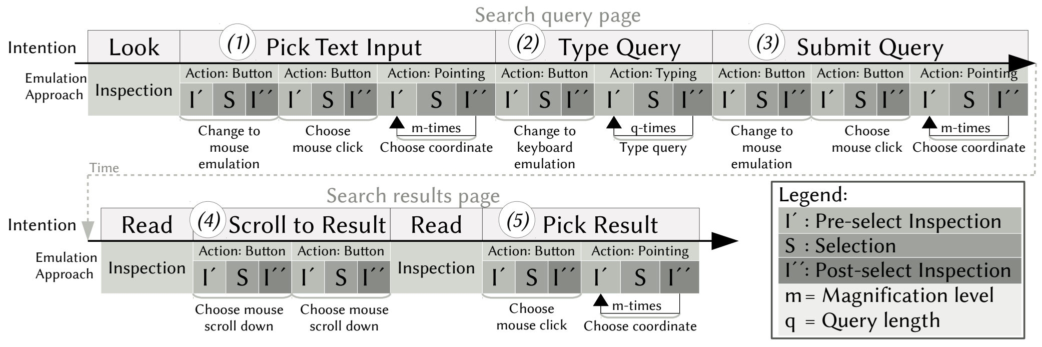


Fig. 3. Input actions to place a query on a search engine Web site and select a result with the emulation approach. We assume a gaze-based pointing method with multiple magnification steps and a dwell time-based virtual keyboard. The user opens a search engine Web page, picks the text input to set the focus on the keyboard *(1)*, types the search query *(2)*, and clicks on the submit button of the search engine *(3)*. On the Web page with search results the users scrolls down two times *(4)* and finally picks one result *(5)*.

by the conventional input devices *(6)* for both Web page (a) and Web browser (b) access through the application interface.

3.2.2 Issues in User Experience. Figure 3 shows a Web search task in detail that requires several clicking, scrolling, and typing efforts from a user (see steps *(1)* to *(5)*) as executed with the emulation approach. We identify two issues with the gaze interaction experience of the emulation approach.

Input Action Overhead. The emulation approach does not serve automatically an appropriate interaction method. A user needs to map mentally each interaction element in an application interface onto the most appropriate emulation panel. For example, the user first needs to click on the text input to type into *(1)* and then switch to the panel for emulation of the keyboard *(2)*. This is a cognitively demanding task, especially for a person with motor impairment, who may have never used a mouse or a keyboard. In general, requiring the user to decide between two modes that emulate two different devices leads to high memory load [35] and negatively impacts the user experience. Furthermore, switching between the emulation panels introduces additional steps of input actions, extending both inspection and selection efforts.

Inspection Overhead. Both, application interface and emulation panel, are placed on separate screen areas and there is no feedback from the application towards the emulation. For an input action, the users must shift her attention towards the window with the emulation panel to inspect which input event to perform. Once a user has selected a virtual button in the emulation panel to cause an event, she has to switch the visual attention back onto the application interface to inspect the effect on the application and back again to the emulation panel, when further input action are required. For example, in a Web browser controlled through the emulation approach, each time a user initiates a scroll down or up command, she needs to check the effect on the Web page within the Web browser viewport *(4)*. The additional perceptual and cognitive load caused by shifting the focus and repeated scanning of the environment [37, 60] detriments the user experience.

3.3 Propositions for Improved Gaze-based Interaction Experience

The following two propositions suggest exploiting interface semantics to adapt the interface of an application for a lower time demand, less workload, and higher usability; for a better gaze interaction experience.

3.3.1 Reduce the Input Action Overhead. The common design principle of "minimal input actions" [76] states that a function should be reachable with as few input actions as possible. However, the emulation approach imposes an overhead of input actions, for example by demanding the user to switch between emulation panels, according to the required events. The switching does not only contribute to higher task completion time, but also makes the interaction more tedious for a user as a selection is usually performed via dwell time or eye gestures. We argue that the input action overhead could be reduced by lessening the interpretation gap between the user intention and the interaction elements. In this regard, we propose to utilize the knowledge about the interface semantics to associate the functionality of an interaction element with suitable gaze interaction. For example, interaction with a hyperlink on a Web page could be associated with a left mouse click event, whereas the interaction with a text input would be associated with a virtual keyboard for eye typing. More sophisticated optimizations could be achieved, e.g., if an text input is designed for password entry, a gaze-based text entry interface for secure password entry can be instantiated [48].

3.3.2 Reduce the Inspection Overhead. For the required input actions, an inherent part of interaction is to visually inspect the interface. The user needs to assess which interaction element to select prior to a selection and inspect the effect of the selection on the application afterwards.

Pre-select Inspection. The design guideline "recognition than recall" [67] argues that a user should not be expected to remember the functionality of an interface, but to recognize the functionality through an intuitive design. This issue is even more relevant in gaze-based interaction due to the dual roles of the eyes for inspection and selection. In dwell time-based selection, slow inspection might cause unintentional selection and discomfort a user. Thus, it is imperative to minimize the time needed for visual search within a gaze-controlled interface. Understanding of semantics of interaction elements, we may reduce the time needed for reducing visual search during inspection prior a selection. For example, we may augment interaction elements with visual indicators to provide a user a cue about the kind of interaction that is to be expected and to allow in-place interaction with the interaction element.

Post-select Inspection. The design principle "visibility of system status" [76] advises to make the status of a computer application easy to perceive by a user. The emulation approach divides the screen space between the emulation panel and the application interface. After issuing an event in the emulation panel trough a selection, a user must shift her attention to the application interface and inspect the effect of the event there. Knowing the interface semantics allows for providing in situ feedback about selections and may thus help to curb the number of attention shifts, while reducing cognitive load in controlling an interface with eye gaze input. For example, we can retrieve the scrolling status of a document and provide feedback about the scrolling in a relevant position where a user may proceed or finish with scrolling.

4 IMPROVING GAZE INTERACTION EXPERIENCE BY INTERFACE INTROSPECTION

In the previous section we have discussed how the knowledge about interface semantics may help to improve the gaze interaction experience. Introspection can provide us with the interface semantics of interaction elements of an interface and further attributes of an interface that we want to adapt. A Web browser is an application to access rich-featured Web-based information and services like messaging, social media, entertainment, or even office suites. Hence, introspection within the Web browser may enable us to adapt all these Web page interfaces, enhancing Web accessibility. In Section 4.1 we describe a methodology to retrieve, classify, and track interaction elements and their properties in an efficient and robust manner from the Web engine of a Web browser. Then, we show in Section 4.2 how the Web browser and Web page interface may be adapted for eye tracking as input method, e.g., by augmenting Web page interaction elements with *gaze-sensitive icons*.

4.1 Retrieval of Interface Semantics

A Web browser receives a requested Web page as a HTML document. This document consists of XML elements of different types (container, hyperlink, image, etc.) with various attributes (address of hyperlink, assigned style class, etc.). Container elements can have child elements of arbitrary type, allowing for nested XML structures. The structure is parsed into a *Document Object Model* (DOM), which is a tree that consists of DOM nodes. Each DOM node corresponds to a single element from the HTML document and stores standard-conforming attributes as properties. The `<body>` element is parsed into the `document.body` node, which acts as root node of the tree that models the actual Web page content. The visual layout and design of nodes is defined by assigned CSS style classes and values (e.g., color, width, height, or visibility).

4.1.1 Dynamics in the Modern Web. Past approaches of gaze-controlled Web browsers by Abe et al. [1] and the WeyeB [70] prototype parse the HTML document after initial loading of a page and identify hyperlinks for adaptation of gaze-based hyperlink navigation. However, most modern Web pages are dynamic and their structure and layout often change after initial loading. Dynamics is achieved through JavaScript functions, which have read and write access to the DOM tree and the CSS style classes. The JavaScript function calls can be triggered by local events (e.g., infinite page scrolling) and remote events (e.g., news update from live events such as game scores). According to *httparchive.org*,[9] about 97% of the crawled Web pages make use of dynamic JavaScript requests (statistics from 11th September 2017). The JavaScript requests are especially utilized in the context of Asynchronous JavaScript and XML (AJAX), which allows dynamic addition and change of Web page content via client-server communication when encountering local or remote events [66]. Hence, the visibility, position, and associated functionality of the interaction elements on the Web page can change, affecting the gaze-controlled interaction with these interaction elements. Document level properties like page height can be kept up-to-date through polling the corresponding value from the root of the DOM tree. However, an observation of the Web page structure needs to work efficiently at run-time on the complete DOM tree to track changes in appearance and functionality of the interface of a Web page.

4.1.2 Web Page Interface Semantics: Retrieval, Classification, and Tracking of Interaction Elements. The DOM standard features the Mutation Observer [29], which is a mechanism to track changes to the DOM tree and DOM node properties. The Mutation Observer is available in the Web engines of current browsers[10] and allows for efficiently tracking changes in dynamic Web pages independently of the operating system and browser. We inject a JavaScript snippet including a Mutation Observer instance into the `document.body` node of the DOM tree every time a Web page is loaded. The Mutation Observer receives records about creation, update, or removal of nodes and properties within the DOM tree. An example from our implementation in GazeTheWeb is depicted in Figure 4, where a Mutation Observer retrieves, classifies, and tracks a newly attached text input and stores information like visibility, size, position, id, and textual content in a list dedicated to capturing such information about all interaction elements.

We perform a rule-based classification process based on the tag, the type, and the role property assigned to a DOM node by the attributes of the HTML element. Figure 5 summarizes the classification process implemented in GazeTheWeb. For example, an element `<input type="search"/>`

[9]http://httparchive.org/interesting.php
[10]Check compatibility at the bottom: https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver

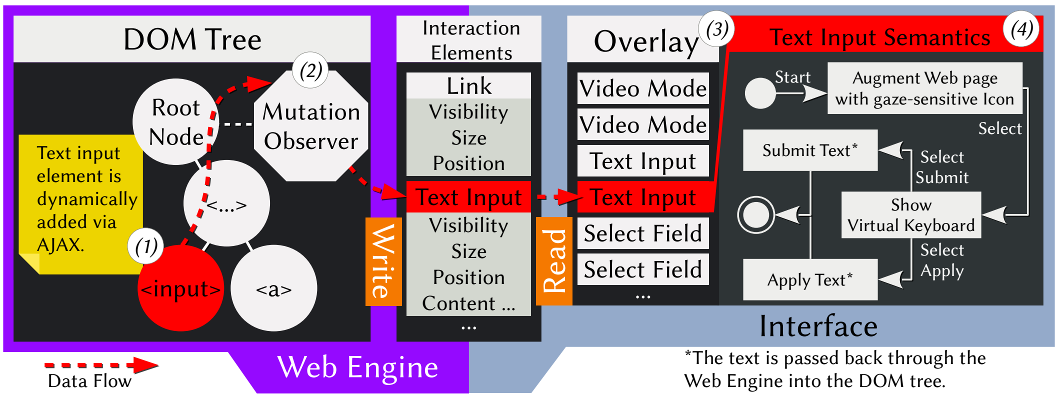


Fig. 4. Data flow for real-time interaction element classification and tracking. In the example, a text input is added dynamically to the DOM tree of the Web page via an AJAX execution. The node with tag <input> is appended to the DOM tree *(1)* below an existing structure. The Mutation Observer on the root node *(2)* is notified about the change and our approach classifies the node as interaction element of type text input (see Figure 5 for details about the classification process). The interface-defining code is called to perform a lookup of the found interaction elements and augments the Web page interface with an associated gaze-sensitive icon *(3)*. When the user selects the gaze-sensitive icon associated with the text input, a virtual keyboard for eye typing to enter and submit the text is shown *(4)*.

in the HTML document is mapped onto a DOM node that expects text input from the user, thus the element is classified as text input interaction element. In contrast, an element defined as `<input type="submit"/>` is treated as link interaction element. The classification of DOM nodes in interaction elements and their adaptation for eye gaze input is extensible for further combinations of properties and DOM nodes, like audio tags, radio buttons, or text fields for password entries.

The current standard comprising the Mutation Observer comes with the limitation that the Mutation Observer cannot directly track changes of computed styles. Computed styles are style properties of a node that are not defined by the node itself but adopted from a property defined by parent nodes. For example, a container element, which is not classified to be an interaction element and, hence, is not tracked, changes its visibility property. The change in visibility may also affect the observed descendant nodes, e.g., an interaction element that is a text input. The current standard does not foresee that the Mutation Observer is notified about such change of visibility. One might attempt to observe all elements of the DOM tree and estimate the impact of changes on descendant nodes. However, observing the complete DOM tree is slow on complex Web pages and replicating the behavior of the styling mechanism is a hard task. Therefore, we perform a frequent

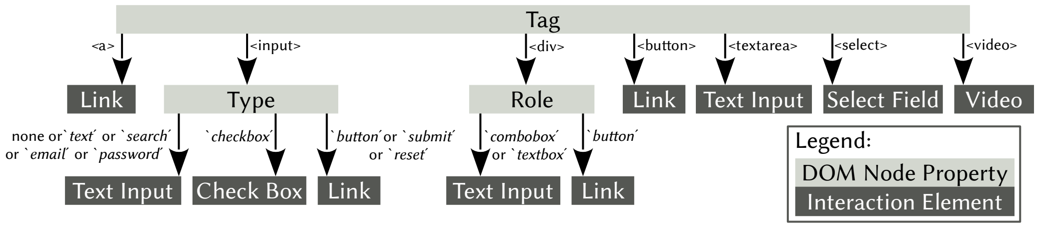


Fig. 5. Classification of DOM nodes to interaction elements as implemented in GazeTheWeb. This classification can be easily extended in the future to support further kind of interaction elements on Web page.

poll of the properties of interaction elements during run time and regularly check for changes. Polling can be limited to interaction elements deemed suitable for adaptation to gaze interaction, because creation and removal of nodes is fully observable by the Mutation Observer itself.

4.1.3 *Web Browser Interface Semantics: Page Meta-Information and Browser Control.*

Besides the Web page content from the `<body>` element, we also extract meta-information from the `<head>` element of a Web page document and retrieve status information through the utilized Web engine. The acquired data, e.g., the title and URL, the favicon, the area extent of a Web page, and the current scrolling status, is imperative to adapt not only the interaction within the space of the Web page but with the Web browser interface in general. We retrieve the browser control functionalities from the Web engine to load a page from a certain URL, to scroll a Web page in the viewport, and to adjust the zoom level. In GazeTheWeb, we make the general Web browser functionalities available through the gaze-controlled interface. Thus, we use the page meta-information for a better user experience in operating those general Web browsing functionalities.

The proposed methodology for introspection can be implemented on Web engines that follow the DOM standard, however, the gaze adaptation itself requires custom interface components, e.g., for the augmentation by gaze-sensitive icons and for the integration of the eye tracking environment into the Web browser, like a virtual keyboard for eye typing. The adaptation of the Web environment through custom interface components within GazeTheWeb is described in the next section.

4.2 Adapting the Interface for Improved Gaze Interaction Experience

The Web standard offers various types of elements on a Web page and the means of interaction varies for these elements. Thus, we categorize the adaptation of Web browsing interactions as following: First, we consider complex elements that offer multiple options of interaction, e.g., text inputs, select fields, or videos. Second, we consider simple elements that offer a single option for interaction and are more frequently found in Web pages compared to complex elements, e.g., hyperlinks or check boxes. Third, we consider the interaction with the Web browser to scroll a page within the viewport, to change the URL, to mark and to choose a bookmark, or to manage tabs.

4.2.1 *Adapting the Complex Web Page Interaction Elements.*

Figure 6 shows an example of complex interaction elements (marked as (a), (b), and (c)) in a Web page.[11] We augment these complex elements with gaze-sensitive icons to adapt the interaction based on their attribute properties such as size, position, and type. For a gaze-sensitive icon, the positioning is determined by the position and size of its associated element on a Web page; the appearance and interaction mode is determined by the type of element as classified according to Figure 5. In the following, we describe how these gaze-sensitive icons enhance the gaze interaction experience. First, the optimized size of gaze-sensitive icons allows for an accurate selection via eye tracking. Second, each gaze-sensitive icon features a symbol to provide the user a visual cue about the type of the associated interaction element. Both, the spatial relation of the gaze-sensitive icons to the associated Web page content and the symbol itself, potentially reduce the pre-select inspection time. Third, a selection of a gaze-sensitive icon summons the most appropriate interaction mode with respect to the specific type of interaction element, as depicted in Figure 7. This reduces overhead in input actions in comparison to the emulation approach, because individual type recognition and switching between different emulation panels is not required. We describe the adapted complex interaction elements

[11] A video entry on the MSN Web portal has been used as running examples in this section. The Web page can be reached under the following URL: https://www.msn.com/en-us/news/video/roaring-crowds-greet-new-royal-baby/vi-AAweZlk.

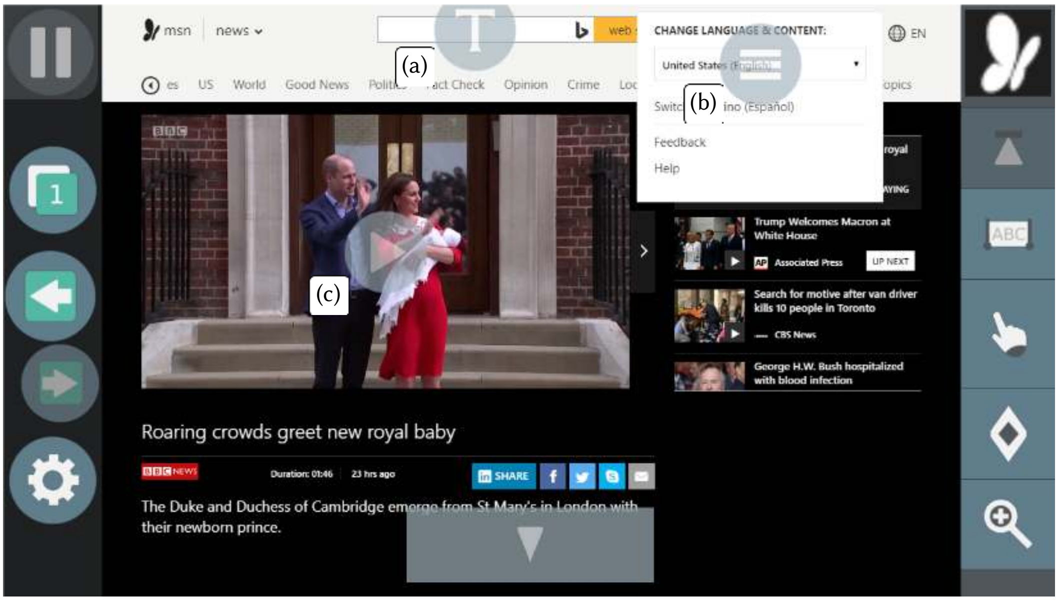
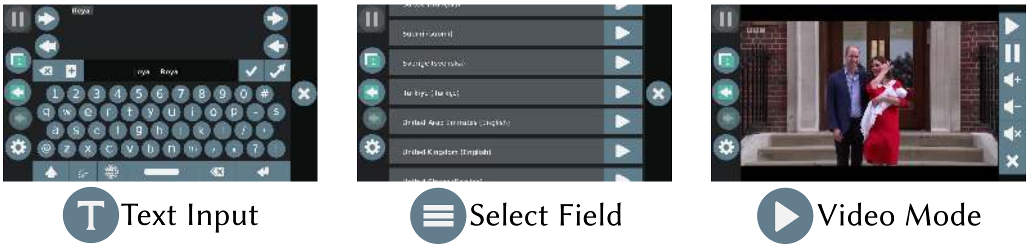

(a)
(b)
(c)
msn
news
CHANGE LANGUAGE & CONTENT:
United States (English)
Feedback
Help
Roaring crowds greet new royal baby
Trump Welcomes Macron at White House
Search for motive after van driver kills 10 people in Toronto
George H.W. Bush hospitalized with blood infection
The Duke and Duchess of Cambridge emerge from St Mary's in London with their newborn prince.

Fig. 6. Complex interaction elements on a Web page are augmented with gaze-sensitive icons.

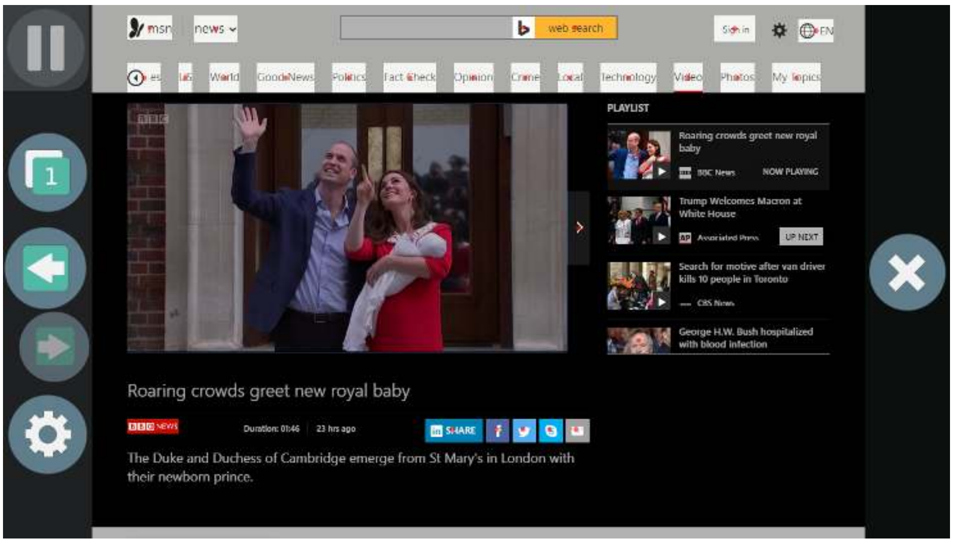


(a) Virtual keyboard for text input. (b) List of options from select field. (c) Mode with video controls.

Fig. 7. Gaze-sensitive icons on a Web page are associated with gaze-controlled interface modes for sophisticated gaze-based interaction.

and their associated gaze-controlled interface modes of GazeTheWeb in the following:

(a) A text input is a Web page interaction element that allows a user to insert text for a search query, information transactions like filling a form, or sending a chat message. In conventional interaction, text insertion is directly performed via a physical keyboard. However, in gaze-based interaction, sophisticated eye typing techniques are required to translate gaze signals into keystrokes. Most of the eye typing approaches display each character of the alphabet as virtual button. Therefore, each keystroke can be interpreted as an interaction option, which makes text input a complex interaction. We provide the user with an interface mode featuring a virtual keyboard for eye typing, including the option to directly submit a query without any additional selection on the Web page. Additionally, other semantics of the text input can be queried for further optimizations of the virtual keyboard, e.g., to automatically provide a discrete and secure way to insert passwords or to



Roaring crowds greet new royal baby

Duration: 01:46 23 hrs ago

The Duke and Duchess of Cambridge emerge from St Mary's in London with their newborn prince.

Roaring crowds greet new royal baby — BBC News

Trump Welcomes Macron at White House — Associated Press

Search for motive after van driver kills 10 people in Toronto — CBS News

George H.W. Bush hospitalized with blood infection

Fig. 8. Click mode for interaction with simple Web page interaction elements. The simple Web page interaction elements are highlighted and a red focus point is displayed over their center.

determine the expected language to support the user in the typing task.

(b) A select field is a Web page interaction element that allows the user to select among predefined options, like date of birth, filters of search results, or the localization. The select field offers multiple options for the user to choose from, and, hence, we categorize the select field as a complex interaction element. We present the available options in an interface mode with virtual button associated to each option, allowing for a convenient gaze-based interaction. If the number of options extends beyond the vertical screen space, the currently inspected option is vertically moved to the center of the screen and the interface mode automatically reveals further options in the direction of movement through scrolling.

(c) A video is a Web page interaction element that embeds video content into a Web page. Videos offer multiple options for control, e.g., play, pause, or skip, which makes the video element a complex interaction element according to our definition. Videos are especially cumbersome to interact with in the gaze-based emulation approach, as the controls of video on a Web page usually disappear when no mouse movement is detected over the video by the Web browser. However, mouse movement in emulation is only inherently performed through positioning the mouse cursor to emulate a mouse click event. Instead, we suggest a gaze-controlled interface mode that shows the video instead of the Web page and features associated virtual buttons to control the video.

The interaction with the above listed complex interaction elements has been gaze-adapted for the scope of our experiments, however, adaptation is not limited to text inputs, select fields, and videos. In similar fashion, interface semantics about less common but standardized interaction elements like sliders or audio players might be defined and their interaction adapted with dedicated gaze-controlled interface modes.

4.2.2 Adapting the Simple Web Page Interaction Elements. In the Web browsing scenario, users face a high number of interaction elements with a single mean of interaction, like hyperlinks or check boxes. Contrary to the complex interaction elements that might have to be handled in various ways depending on their exact type, these elements are generalizable to the same type of interaction, e.g., a click to navigate to a hyperlink or a click to toggle a check box. Hence, a dedicated gaze-sensitive icon is not required for each of these simple elements. Furthermore, gaze-sensitive icons for gaze-control need to occupy a minimum amount of screen space to account for accuracy limitations in eye tracking. A gaze-sensitive icon for each hyperlink or check box would lead to overlapping gaze-sensitive icons and therefore to ambiguous interpretation of gaze signals. We inherently associate all simple interaction elements with a click mode of gaze-based selection. The user can activate the click mode by selecting the virtual button with the "finger pointing" symbol in the right panel of GazeTheWeb, see Figure 6. The click mode is implemented as a multi-step magnification [6] of the Web page, to allow for an accurate selection among all candidate simple interaction elements. Knowing the position of these elements provides the possibility of dynamic adjustments to deal with the limited accuracy of eye gaze estimation, i.e., a hyperlink can be selected in the near vicinity of a user's gaze coordinates.

Moreover, minimalistic user interface design as of today, e.g., Google Material design,[12] often does not visually hint simple interaction events like hyperlinks or buttons through visual indicators like embossing or shadows. Users who use a mouse as pointing device may observe the change from "mouse pointer" to "finger pointer" when hovering with the cursor over simple interaction elements at inspection. Our approach removes this prerequisite of hovering by highlighting the simple interaction elements in the click mode, see Figure 8. Furthermore, we overlay each simple interaction element with a *focus point* [49], rendered as red dot in the center of the simple interaction element as displayed. Focus points help in the gaze-based selection process, as otherwise users tend to roam their gaze within a highlighted interaction element and the eye tracking system is not able to detect a stable fixation, as it is required for the selection. Both, highlighting and focus point, reduce the pre-select inspection overhead in comparison to the emulation approach. In addition, we provide local feedback about the issued selection by displaying a circular wave that collapses at the location of the selected simple interaction element, allowing for fast post-select inspection of the selection, without a shift in attention.

We argue that the described adaptation of complex and simple elements can enhance the gaze interaction experience compared to an emulation approach. To showcase this, we revisit the sequence of input actions for a Web search scenario in the emulation approach (Figure 3) with a corresponding optimized sequence of input actions through the proposed adaptations in Figure 9.

4.2.3 Adapting the Web Browser. We do not only consider the Web page proper, but we have also improve the gaze interaction experience with a Web browser in general, to provide a wholesome user experience. Kellar et al. [43] have examined user behavior in Web browsing and classify Web browsing functionality into two categories: "within-task-session" and "new-task-session". They define "within-task-session" as the Web browser functionality used while a user works on a task, e.g., a Web search or shopping. In contrast, "new-task-session" functionality is primarily used by a user at the beginning of a task. This categorization has been incorporated into the two main interfaces of GazeTheWeb, as depicted in Figure 10.

(a) The primary interface mode, as shown in Figure 10a, contains the "within-task-session" functionality. The interface presents the Web page with the augmented complex interaction elements inside a viewport in the center of the screen. There are two panels on both sides of the viewport to support interactions within a browsing session through selection of virtual buttons, e.g., to

[12] https://material.io/design

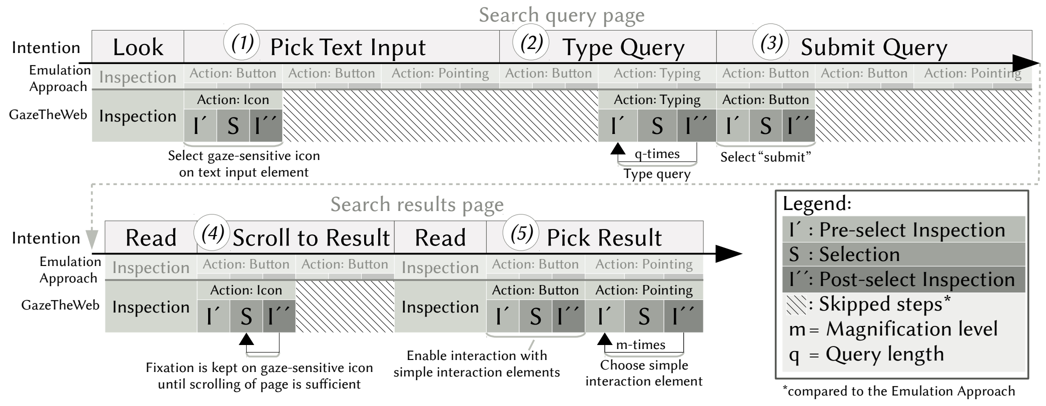


Fig. 9. Input actions to place a query on a search engine Web site and selection of a result with GazeTheWeb, analogously to Figure 3. Instead of input actions to switch the emulation panels, a user can just select the gaze-sensitive icon associated with the text input *(1)*. The virtual keyboard for eye typing is automatically presented after selection of the gaze-sensitive icon and a user can type a query *(2)*. The interface mode of the keyboard allows for instant submission of the query, which saves another selection on the Web page *(3)*. Scrolling down on the Web page is sped up, too, as the user can stay with her focus on the gaze-sensitive icons for scrolling *(4)*. Additionally, a user's selection is facilitated by the highlight of the search results during the final pointing effort *(5)*. This is not reflected in the figure, as the figure does only account for the reduction of input actions. In addition, reduction of inspection efforts can be expected at interaction, e.g., the gaze-sensitive icon associated with the text input is placed in situ on the Web page interface.

activate click mode in order to select simple interaction elements, to go back to the previous page and forward to the next page, or to scroll directly to the top of the current page. Additional gaze-sensitive icons are overlaid on the top and the bottom of the Web page viewport to support scrolling of the Web page within the viewport. The placement of the scrolling functionality within the Web page viewport reduces the pre-select inspection overhead for the user. A vertical filling of the gaze-sensitive icons provide indication about the current scrolling status of the Web page. This in situ feedback about the scrolling status additionally reduces post-select overhead, as a user can solely focus on the gaze-sensitive icon to scroll until a desired scroll position is reached.

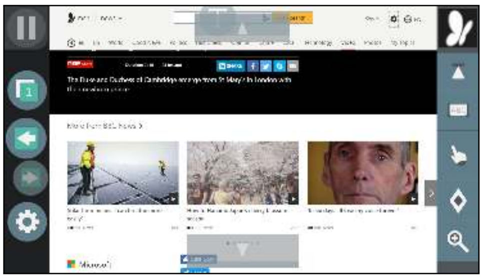
(b) The secondary interface mode, as shown in Figure 10b, contains "new-task-session" functionality. This includes functionality to enter a the URL, to mark or to choose bookmarks, and to manage tabs. We enrich the secondary interface with information about the current Web page, like URL, title, and a preview of the viewport content. The presented information supports the user in selecting whether to load another page or to change to another tab. The explicit exposure of functionality through virtual buttons in the interface of the application does reduce the required input action count and the inspection overhead in comparison to the emulation approach.

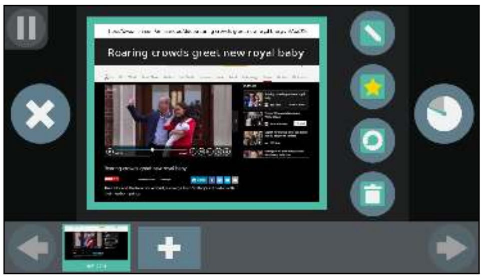
We have implemented the GazeTheWeb with the Chromium Embedded Framework (CEF)[13] as Web engine and the eyeGUI library [64] for composing the interface.

5 EVALUATION I: COMPARISON WITH EMULATION APPROACH

The goal of our first evaluation is to assess whether the implementation of the propositions for improved gaze interaction experience in GazeTheWeb enhances the user experience in comparison

[13] https://bitbucket.org/chromiumembedded/cef


(a) Primary interface mode. The second virtual button from top on the left panel allows loading of secondary interface mode.



(b) Secondary interface mode. Selection of the virtual button with the cross symbol loads the primary interface mode.

Fig. 10. Adaptation of general Web browser functionality.

to the emulation approach. Therefore, we have conducted a lab study to evaluate GazeTheWeb against a state-of-the-art emulation implementation. We used OptiKey [80] as an instance of the emulation approach to control the popular Google Chrome browser.[14] OptiKey has received high praise in recent years as a tool to assist gaze-based computer access.[15,16] OptiKey has also been used at MIT[17] for supporting people with motor disabilities in computer access. You can find a detailed description of the gaze-based interaction with OptiKey in the Appendix A.

The focus of our lab study is to analyze whether the gaze interaction experience can be improved for users interacting with Web using our approach in comparison to the emulation approach. For a reasonable comparison of OptiKey and GazeTheWeb, we used the simplistic information search and browsing scenario. This is because more complex interaction scenarios, like a video, are not accessible through the emulation approach. For example, on popular video platforms like YouTube, the videos embed their controls into the video view and only reveal them upon cursor movement. However, the emulation implements cursor placement rather than cursor movement, and, hence, the controls of videos are not visible and the user is not able to select them through emulation approach. Hereby, we choose an interaction scenario where both approaches are functional for the user, such that they give us relevant feedback on user experience. We assess the widespread feasibility of GazeTheWeb in handling dynamic Web pages with complex interaction elements, and supporting everyday browsing tasks of end users, in Section 6.

5.1 Methodology

To assess user experience, we propose to cover an information search and browsing task. We quantify the task completion time, perceived workload, and usability. The main hypothesis is that GazeTheWeb allows for less time demand upon completing the search and browsing task, while users give higher ratings in usability and lower for workload than for OptiKey. For this purpose, we conducted lab study at our university with 20 participants (9 female and 11 male) in the age range from 23 to 31 years (average = 25.55, sd = 2.16). All these participants were students at our university with no prior experience in operating GazeTheWeb or OptiKey. Seven participants wore corrective lenses (3 female and 4 male). The participants were paid each an amount of 10 euro for their effort after the trial. The eye tracking device was attached to the bottom of a 24 inch monitor

[14]https://www.google.com/chrome
[15]http://www.businessinsider.com/an-eye-tracking-interface-helps-als-patients-use-computers-2015-9
[16]https://alsnewstoday.com/2016/03/08/article-for-als
[17]https://vimeo.com/148316508

which resolution had been set to 1600x900 pixels. The used eye tracking device was an independent variable between subjects, i.e., SMI REDn scientific eye tracker[18] (sampling frequency 60 Hz) was used for the study with 10 participants. The other 10 participants operated the systems with a Tobii EyeX controller[19] (sampling frequency 30-70 Hz), which is an affordable eye tracking device that is designed for consumer-use. Artificial illumination and blocking of sunlight provided a similar lighting condition for all participants.

For both test conditions (GazeTheWeb versus OptiKey and Google Chrome), dwell time has been used as gaze-based selection method. The time threshold for dwelling of both systems was set to one second, as appropriate for untrained users and chosen in related evaluations [70]. Depending on the system setup, we analyze the required time to succeed in the tasks and the subjective analysis from the System Usability Scale (SUS) [11], NASA Task Load Index (NASA-TLX) [31], design heuristics [73], and qualitative feedback. For the experiments, counter-balancing for the order of the system was used so as to eliminate any bias of one system over the other. In the rest of the section, the system setups with *GazeTheWeb* and *OptiKey and Google Chrome* are referred as GTW and OK, respectively.

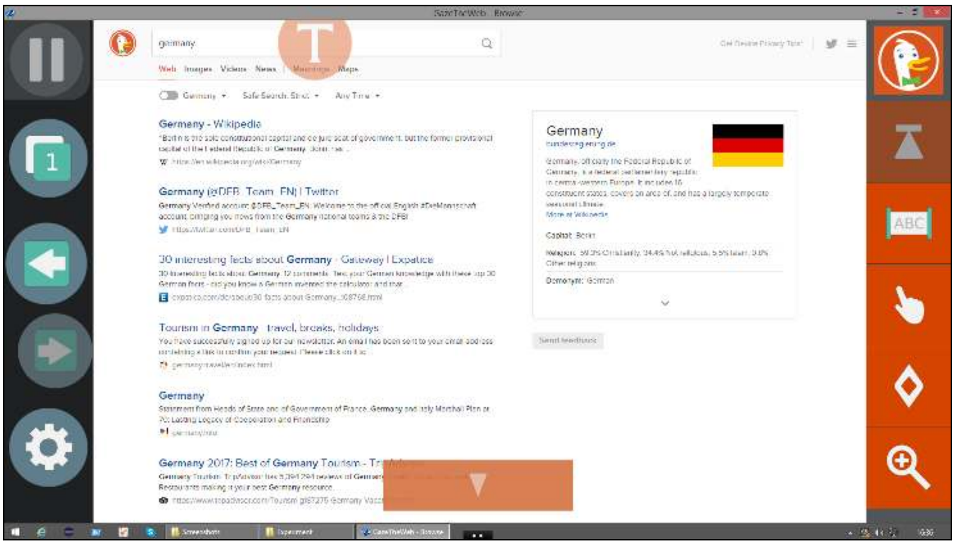
5.2 Procedure

Prior to each execution, we gave the participants an explanation about the general nature of the tasks. After the oral explanation, we performed an initial eye tracking device calibration and provided the participants with a training phase that helped them to understand the functionality of the systems. Upon completion of the training, we gave them a short period of time to get accustomed to the environment on their own, so that they gathered a certain confidence with a system before the actual experimental session started. After a break of a few minutes, the experimental session started, including a second calibration of the eye tracking device.

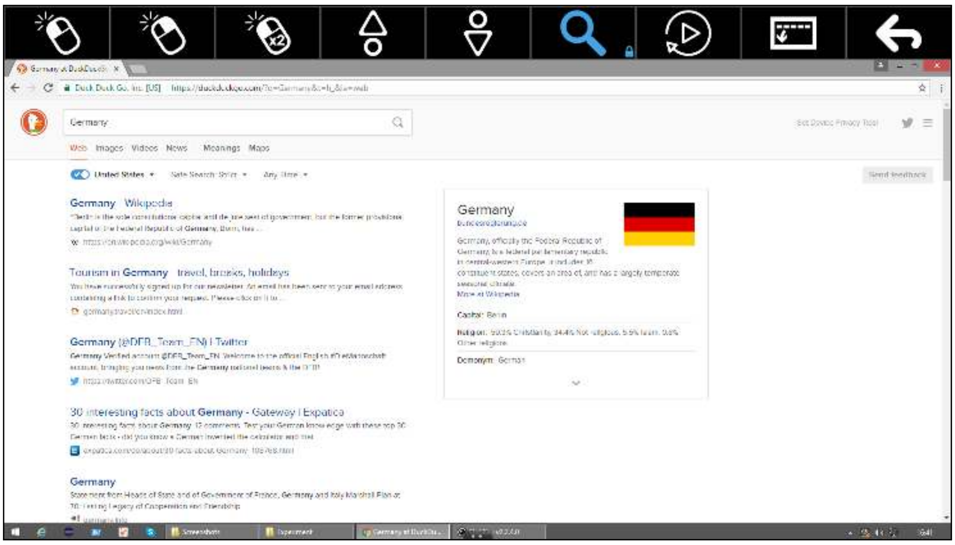
Tasks. The tasks involved common user activities to search and browse the Web to find specific information. Each participant was instructed to enter the search string "Germany" on the Duck-DuckGo search engine page. They were asked to go to the English Wikipedia page about Germany from the search results. See Figure 11 for screenshots of both systems displaying the Web page with search results. Upon reaching the page about Germany, the participants had to find the section about constituent states, and from there the first task was to select the particular state North Rhine Westphalia (NRW). Upon reaching the page, they had to scroll to the cities of NRW and select the particular city Bonn from there. Once on the city page, they had to bookmark it and go back to the constituent states section of the NRW entry. The same process was executed for the constituent State of Baden-Württemberg and the city Mannheim, respectively the state Lower Saxony and the city Oldenburg Land. Finally, the participants were instructed to access a city from the list of bookmarks.

Survey. After the participants finished the tasks using one system, they were handed a SUS and a NASA-TLX questionnaire. The SUS questionnaire is used to measure the overall usability of applications. The SUS contains 10 questions, which are answered on a five point Likert scale from *strongly disagree* to *strongly agree*. The NASA-TLX questionnaire contains six components: mental demand, physical demand, temporal demand, performance, effort, and frustration. For each component, the participant specified the most applicable scores on a scale from 1 (low) to 7 (high). In addition, a custom gaze interaction design heuristics evaluation for gaze-controlled Web interfaces has been used [67, 73], to analyze **#1** How was the *visibility* of the main interaction elements?

[18]SMI REDn scientific https://www.smivision.com/eye-tracking/products/remote-eye-tracking
[19]Tobii EyeX controller https://tobiigaming.com



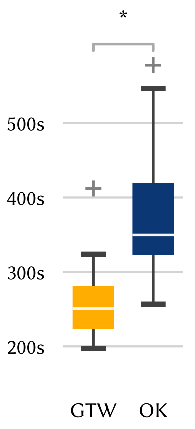
(a) GazeTheWeb



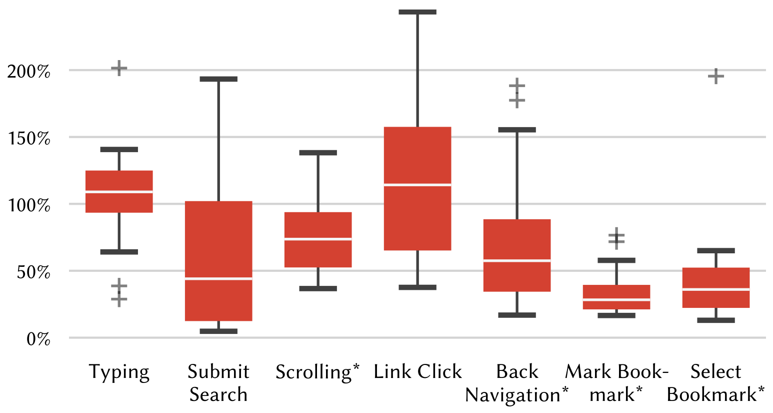
(b) OptiKey and Google Chrome

Fig. 11. Screenshots of both systems from the first evaluation. The screenshots have been taken after the search query "Germany" had been submitted to the Duckduckgo search engine. Both systems display the Web page with the search results in the screenshots.

#2 How comfortable was the *size* of the interaction elements? **#3** How *intuitive* was the reading and scrolling experience? **#4** How *easy* was handling the link navigation in the browser? **#5** How easy was it to recover from *errors* made? **#6** How *close* do you feel is this browser to conventional browser environment? The questionnaire can be answered on a scale from 1 *strongly disagree* to 10



(a) Completion time



(b) Times in GazeTheWeb in comparison to times in OptiKey as baseline. 100% means a participant needed same time in GazeTheWeb as in OptiKey, 50% means a participant needed only half the time with GazeTheWeb than with OptiKey, for the same task.

Fig. 12. Box plot of the times required by the participants for the execution of a browsing task. The star symbol marks a significant difference between both systems in the reported dependent variable.

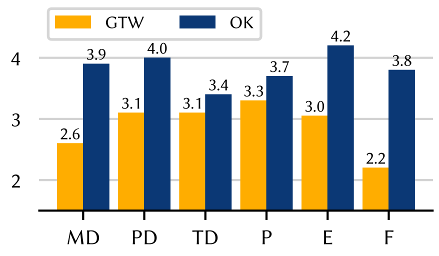
strongly agree. At the end, participants were asked if they have any general feedback on what they liked, disliked, and comments for improvement.

5.3 Results

We tested the significance of the influence of the two eye tracking devices. For this, we have performed unpaired significance tests on completion time, SUS, and NASA-TLX average raw value between the participant groups using a different eye tracking device but the same system. A Mann-Whitney U test of completion time between the groups using different eye tracking devices results in p = 0.43 with GTW as system and p = 0.85 with OK as system. Analogously, we performed a two-sample t-test for the SUS scores and report two-tailed p values of p = 0.14 for GTW and p = 0.43 for OK. Furthermore, a two-sample t-test two-tailed of the average NASA-TLX raw scores result in p = 0.27 for GTW and p = 1.0 for OK. Hence, the effect of the eye tracking devices between participants were non-significant, and we report the objective and subjective outcomes for the complete set of 20 participants from the first evaluation in the following.

Objective Results. All participants succeeded in the given tasks with both systems. The participants were in average over 135 seconds faster with GTW than with OK to complete the overall tasks. Furthermore, the time differences between both systems are normally distributed, according to a Shapiro-Wilk test with p = 0.05 threshold. This allows us to assess the significance of the differences by a paired t-test calculating the two-tailed p-value. We report a significant difference in the completion time for GTW (average = 261.91s, sd = 49.25) and OK (average = 397.43s, sd = 130.76s), with t(19) = -5.23, p= 4.78E-5, and a high effect size of Cohen's d= 1.17. See Figure 12a for a box plot showing the consistent times the participant required to fulfill all tasks in GTW, whereas the times for OK showed much higher variety.

In a gaze-based interaction task on dynamic stimuli it is difficult to record exact timings of highly granular activities. In contrast to usual eye tracking experiments (attention analysis on static



(a) Raw NASA-TLX scores. Lower scores signify lower perceived workload. MD = Mental Demand, PD = Physical Demand, TD = Temporal Demand, P = Performance, E = Effort, F = Frustration.

Heuristics	GTW	OK
#1 Visibility	8.45	7.65
#2 Size	7.95	6.95
#3 Reading	8.0	6.05
#4 Link Navigation	7.25	5.35
#5 Error Recovering	7.95	6.55
#6 Conventional	7.9	6.4

(b) Heuristics Scores. Higher scores are better.

Fig. 13. Feedback by the participants of the lab study at our university.

stimuli), there is a lot of information on the screen with a full set of Web browser controls and the Web page itself. Especially, it is not feasible to determine a hard threshold between inspection to *read* or inspection to *interact*, i.e., pre-select or post-select inspection.

Thus, we report more details about the user performance by separating the task into atomic Web browsing activities, e.g., scrolling from the top of the page towards desired hyperlinks. We compare the timings within subject in GTW with the timings in OK as baseline and provide a box plot in Figure 12b. The timings for typing and hyperlink clicking appear to be similar, whereas submission of search, back navigation, and bookmark management have been achieved faster with GTW than using OK. We report a significant improvement in times for GTW over OK for scrolling (W = 20, Z = -3.17, $p < 0.05$, r = 0.71), back navigation (W = 41, Z = -2.39, $p < 0.05$, r = 0.53), marking (W = 0, Z = -3.92, $p < 0.05$, r = 0.88), and selection (W = 14, Z = -3.4, $p < 0.05$, r = 0.76) of bookmarks.

Subjective Results. The subjective evaluation of the experimental session was performed using the SUS for general usability assessment, NASA-TLX to measure the workload, and custom heuristics criteria to evaluate the Web browsing experience.

The survey shows average SUS usability scores of 77.13 for GTW in contrast to 55.0 for OK, indicating an over-average acceptability rate of the gaze-adapted system of GTW among the participants. The difference between the SUS scores by the participants are normally distributed, according to a Shapiro-Wilk test with p=0.05 threshold, and we have performed a paired t-test to assess the significance of the higher rating for GTW. There is a significant difference in the overall SUS scores for GTW (average = 77.13, sd = 16.08) and OK (average = 55.0, sd = 19.36), t(19) = 3.6, p = 0.0019.

The consistently better ratings of GTW over OK in terms of NASA-TLX mental workload (MD), physical demand (PD), level of effort (E), and sense of stress and irritation (F) are presented in Figure 13a. The feeling of success (P) in accomplishment of the task was also slightly better for GTW in comparison to OK, since a lower value means a higher feeling of success in NASA-TLX questionnaire design.

The results of the gaze interaction design heuristics questionnaire is shown in Table 13b, where we can see that actions like the ease of recovering from errors (#5) in GTW receive better scores than in OK. The intuitive factor (#3) and ease of hyperlink navigation (#4) is rated better for GTW.

5.4 Discussion

Objective results from the study indicate significantly lower overall task completion times for GazeTheWeb compared to the emulation approach. More specifically, we observe significant improvements in the activities like scrolling, back navigation, and bookmarking, which typically involve multiple selections and inspections. Typing and link clicking in GTW are similar to OK. This validates our experimental setup, as both systems employ similar mechanisms and selection methods. Both keyboards, in GTW and OK, work with dwell-time sensitive keys and are set to the same dwell time. Link clicking first needs the selection of the specific mode – in GTW by selection of the virtual button with the "finger pointing" symbol on the right panel and in OK by selection of the "left mouse click" virtual button – and then a multi-step magnification. More adapted Web browsing activities show the potential of gaze-adapted interfaces. The median of submitting the typed search query, scrolling, back navigation, and especially menu navigation like marking and selecting bookmarks, are with GTW below the baseline of OK, see Figure 12b.

This strengthens our propositions to focus on more *usable* gaze interaction experience and implies that the user interface adaptation is an indispensable aspect of gaze-based interaction, irrespective of conventional issue like eye tracking quality. Hence, we argue that in addition to the technological advancements for more precise and accurate gaze signals, the research on interface user experience aspects need to be evolved to make eye tracking more usable and acceptable interaction technology for end users.

The subjective results from the study indicated similar trend, i.e., superiority of GazeTheWeb compared to the emulation approach, as the rating by the participants positions it far above the general median of systems. The average SUS score of 77.13 for GTW is considered to be close to the high order as per SUS guidelines.[20] A score equal or greater than 80.3 indicates a positioning within the 10% of applications with very good usability and the tendency of users to recommend the system to friends. The scores achieved by OptiKey are not satisfying, because the estimated score of 55.0 is close to the score of 51 and below, which would sort the usability of the application into the category among 15% of the worst evaluated applications. For the NASA-TLX scores, all major points of workload are rated better for GazeTheWeb. The participants felt especially less mental demand, less effort, and less frustration, but a higher sense of success. Our interpretation is that the additional command-translation layer of the emulation approach causes higher overall workload, as the users had first to imagine how to achieve a task with mouse and keyboard and then to execute it via the gaze-controlled emulation tools. We obtained similar results in a separate preliminary study comparing the interaction with Twitter using a custom gaze-controlled interface and an emulation [47]. The heuristics questionnaire results further validated the hypothesis of GazeTheWeb possessing a more suitable design for gaze-controlled Web interaction, as the feeling of controlling a conventional Web environment was even higher for GazeTheWeb than for the combination of OptiKey and the well-known Google Chrome Web browser.

The explicit comments and feedback from participants are aligned with our hypothesis and the presented results. Most of the participants liked the intuitive interaction aspect of GTW, as comments by participants like ***"I****t was easy to navigate and also very easy to get used to start working with it****"*** emphasize the good usability of GTW, whereas about OK a participant reported ***"T****oo cumbersome and physically exhausting. Requires many clicks which are tiring for the eyes****"***. More specifically, participants explicitly reported about OK on their bad experience with regard to required input actions, e.g., ***"M****ultiple click for every action****"*** or ***"R****educe left click mechanism using any other intuitive means. For example, let's say I am accessing the bookmarked URLs it should give easy access to the links by reducing the number of intermediate left clicks****"***. The improved aspect of

[20]http://www.measuringu.com/sus.php

inspection overhead in GTW was also evident, as the participants appreciated the visual indicators for optimizing pre-select inspection by comments such as *"Interaction elements are self-explainable as it was not at all in OptiKey"* and the in situ feedback for optimizing post-select inspection in GTW *"I liked button feedback (if clicked)"*. With regard to high workload in the emulation approach, as indicated in NASA-TLX results, we have already discussed the issue that users might need to imagine how to achieve a task with mouse and keyboard and then to execute it via the gaze-controlled emulation tools. The explicit comments on OK usability substantiates the argument, e.g., *"Interaction elements should be easy to use, for a non-technical person it can be hard to use for example selecting button (like left click) every time after the [beginning of the] task"*, *"It seems too quick and [I] need to remember exact process I am following"* or *"Too complicated, too many interactions, too many clicks"*.

Additionally, there were some general comments about head movement and fatigue as eye tracking limitations that apply for both systems, e.g., comments such as *"The biggest problem is holding the head in the same position for a longer time period"* or *"Maybe [use] some eye tracking glasses, to make it possible to change the position of the head frequently"*.

6 EVALUATION II: FEASIBILITY OF GAZE THE WEB

In the evaluation of the previous Section 5, our focus was to quantify how effectively GazeTheWeb performs in comparison to an emulation approach. Hence, we reported a task-focused lab study on a simplistic search and Wikipedia browsing scenario where both GazeTheWeb and emulation approach would be functional. In this section, we aim to assess the feasibility of GazeTheWeb in handling of dynamic Web pages with complex interaction elements, and supporting everyday browsing tasks of end users. For a realistic assessment, we primarily involved target users of gaze assistive technology to test the feasibility of GazeTheWeb, i.e., people with motor impairment who would benefit from improved eye tracking-based interaction. We first conducted a controlled lab study to investigate if the users are able to accomplish the daily browsing activities such as communication (writing an email, posting on social media), content creation (editing a photo), and entertainment (watching videos) using GazeTheWeb (Section 6.1). We evaluated the overall feasibility of GazeTheWeb in a field study by installing GazeTheWeb at the home of participants for a one month period (Section 6.2).

6.1 Lab Study

The study was conducted as part of the MAMEM project first phase trial [68]. The MAMEM project's goal has been to integrate people with motor disabilities back into society by enabling them to interact with computers using multimodal interaction channels. The complete trial for multimodal interaction was executed using different device configurations, i.e., a SMI REDn scientific eye tracking device to estimate gaze, a BePlus LTM Bioelectric Signal Amplifier[21] to record electroencephalography (EEG) signals, and a Shimmer3 GSR+[22] to capture galvanic skin response (GSR) and heart rate (HR) signals. The trial consisted of four experimental sessions including *(1)* training with eye gaze and EEG in a persuasive tutorial with gamification elements, *(2)* recording of event-related potentials (ERP), *(3)* sensorimotor rhythms (SMR) execution, and *(4)* dictated everyday-Web-browsing-tasks using GazeTheWeb. In this paper, we discuss the outcomes of experimental session *(4)*, as the dictated everyday-Web-browsing-tasks were performed solely with GazeTheWeb and eye tracking as input method.

[21]http://www.ebneuro.biz/en/neurology/ebneuro/galileo-suite/be-plus-ltm
[22]http://www.shimmersensing.com/products/gsr-optical-pulse-development-kit

Fig. 14. A photo from the MAMEM trials.[24]A participant with Parkinson's Disease selects an E-Mail.

6.1.1 Methodology. The trials were executed at three clinical sites, each responsible for one group of six participants with motor impairment. In total 18 participants with motor impairment took part in the lab study. Six participants (1 female, 5 male) with Parkinson's Disease (average age = 64, sd = 6.69) at AUTH - School of Medicine, Greece; six participants (2 female, 4 male) with a neuro-muscular disease (average age = 34.2, sd = 6.18) at MDA Hellas, Greece; and another six participants (1 female, 5 male) with a spinal-cord injury (average age = 45, sd = 16.4) at SHEBA - Academic Medical Center Hospital, Israel, participated in the trial. Additionally, 18 participants without motor impairment (5 female, 13 male; average age = 45, sd = 13) attended the experiments as the control group, with the same experimental setup as the target group. All 36 participants had no prior experience with eye tracking.

Before the trials, the clinical protocol was followed as per the ICH-GCP65 guidelines,[23] and an ethical approval (Helsinki Approval) was obtained. The guidelines provide "an international ethical and scientific quality standard for designing, conducting, recording and reporting trials that involve the participation of human subjects".

6.1.2 Procedure. Before the experiment of dictated task was executed, all participants underwent an interactive tutorial with persuasion [68] as first experimental session of the trial. The tutorial introduced the fundamental functionalities of GazeTheWeb, necessary to fulfill the dictated tasks. The dwell time of the system was set to one second, same as for the lab study at the university that we have described in the previous section and sufficient for untrained users. The participants were asked to perform dictated everyday tasks: To read and reply to an E-Mail using "Gmail.com" (E-Mail Task), to edit a photo using "Picresize.com" (Photo Task), to post on social media using "Twitter.com" (Social Media Task), and to watch a video using "Youtube.com" (Video Task). For more details about the single tasks within the experiment, please refer to the Appendix B. Before the dictated tasks, the participants did a 15 minutes break from the trial. After the dictated tasks, the participants filled a SUS questionnaire. During the dictated tasks, the EEG, GSR, and HR signals were recorded for project-related evaluation purposes.

6.1.3 Results. Analogously to the previous section, we will first present the results of the objective measures and then provide the subjective feedback.

[23]http://www.ich.org/products/guidelines/efficacy/efficacy-single/article/good-clinical-practice.html
[24]Photo by Centre for Research & Technology Hellas - Information Technologies Institute.
Photographer: Tasos Papazoglou-Chalikias.

Objective Results. GazeTheWeb successfully enabled the participants to complete the dictated tasks. The results are shown in Table 1, where observation count indicates the number of participants that completed the specified task. There are some participants for whom a single or the complete tasks failed, which is visible through the observation count. One participant with Parkinson's Disease faced claustrophobia and could not proceed the tasks, thus, was excluded from the experiment. For two participants with spinal-cord injury, the eye tracking device was not able to track their eye gaze. They were excluded from the trial as well. For one participant without motor impairment, there was an issue with the internet connection during the photo editing task, why it could not be executed.

Task completion times are denoted with time mean, standard deviation, and median measurements for the respective groups. The last row demonstrates the differences in task completion time between the participants of the target group and the participants of the control group. Because a Shapiro-Wilk test does indicate that the task completion times might be not normally distributed, we have decided to perform a Mann-Whitney U test to compare the task completion times of the two unpaired groups of participants. The Mann-Whitney U test with the two sample sets of task completion times of target and control group reveals no significant difference between both groups, as all computed two-tailed p-values are by far greater than 5%.

Subjective Results. The average SUS score of target group is 72.17, while average score of control group lies at 77.36. These scores are above the average of 68 [11], indicating an above average usability score of GazeTheWeb, analogous to the results of prior evaluation in Section 5.3.

6.1.4 Discussion. The evaluation with the participants from the target groups demonstrates the feasibility regarding everyday tasks like writing an email, editing a photo, participating in a social network, or watching a video. GazeTheWeb allowed the participants to successfully perform four real world tasks on modern Web pages with eye tracking as input method.

Furthermore, the explicit comments and positive feedback from patients specifies the acceptability of gaze-based interaction among target users. The participants liked the intuitiveness of interaction in GazeTheWeb, specifying comments such as ***"I find it especially intriguing and fun that I can direct my actions with my eyes, rather than with my hands"***. The participants also mention the persuasiveness of the technology ***"I found this easy to learn, and the next thing that I want is to improve my speed at using the keyboard in this new way"***. Some of the participants who were already

Table 1. Task evaluation of target against control group, including E-Mail Task (E-Mail), Photo Task (Photo), Social Media Task (Social), and Video Task (Video). Times are provided in seconds.

	Measure	E-Mail	Photo	Social	Video
Target Group	Observation Count	15	15	15	15
	Time Mean	300.87	152.93	246.6	148.6
	Time SD	239.12	72.11	98.16	62.1
	Time Median	210	130	230	125
Control Group	Observation Count	18	17	18	18
	Time Mean	255.17	144.06	253.22	157.44
	Time SD	147.39	76.34	99.44	68.56
	Time Median	202.5	148	220	135
Mann-Whitney U Test	U-value	134.5	119	132.5	127.5
	p-value (two-tailed)	*1.0*	*0.76*	*0.94*	*0.8*

comfortable with their existing means of interaction (e.g., switch input), liked the intuitiveness of the gaze-based interaction. However, at the same time comments like ***"I** am very curious to find out if it can exceed the comfort level that I have achieved with my current device**"*** indicate that the acceptability of eye tracking as assistive technology comprises further challenges such as to compete with existing means of hands-free interaction for individuals.

6.2 Field Study

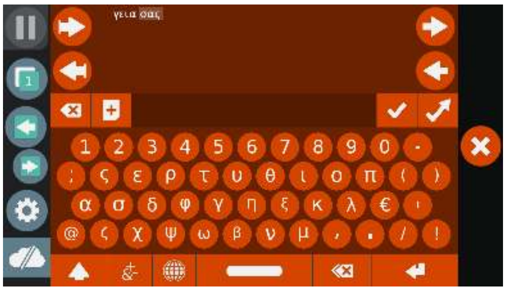
There is a vast amount of Web sites,[25] and it is imperative to investigate the feasibility of GazeTheWeb in handling diverse Web pages that are frequently browsed in an everyday scenario. Hence, in this section, we report on the long-term usage of GazeTheWeb in home environment. We present results from the second phase trials of the MAMEM project, in which GazeTheWeb had been offered to people with motor impairment for home use. The system consisted of a laptop (with GazeTheWeb installed as startup program) and an eye tracking device, deployed to the homes of the 30 participants for one month. The outcome of the field study does indicate whether the adaptations, as presented in Section 4.2.1, have been feasible in a real world scenario of unrestricted Web browsing. Three participants had additionally performed a single day multimodal experiment using eye tracking device, BCI, and GSR devices with a modified version of GazeTheWeb. However, we exclude the results from the multimodal experiment, as it is not in the scope of this article.

6.2.1 Methodology. Similar to the first phase of MAMEM trials, each associated partner organisation contacted participants of one potential target group. MDA Hellas, Greece, supported ten participants (4 female and 6 male, average age 31.5, sd = 4.8) with neuro-muscular diseases. These participants are referred to as MDA 1 to MDA 10 in the following. AUTH - School of Medicine, Greece, supported ten participants (4 female and 6 male, average age 55.6, sd = 7.3) with Parkinson's disease, referred to as AUTH 1 to AUTH 10 in the following. SHEBA - Academic Medical Center Hospital, Israel, supported ten participants (10 male, average age 38.1, sd = 10.8) with spinal cord injury, referred to as SHEBA 1 to SHEBA 10 in the following. Similar to the first trial, the clinical protocol was followed as per the ICH-GCP65 guidelines, and an ethical approval was obtained.

A major aim of the study was to assess the natural Web browsing behaviour of the participants. Hence, there were no guidelines, rewards, or requests to influence them for using the system. Most of the participants already had some computer accessories and assistive solutions available at their home. Therefore, we did not expect that the participants would completely switch their existing means of interaction and start using GazeTheWeb system excessively. However, the persistent usage of GazeTheWeb even by some participants would be relevant indicators on its functionality and effectiveness in supporting daily browsing activities.

Apparatus. GazeTheWeb has been localized to match the language of the participants. The localization does not only include the screen texts of the interface, but also the text input means, i.e., the virtual keyboard for eye typing. The modified keyboard layouts for Greek and Hebrew language to match the native language of the participants are depicted in Figure 15. The participants were able to choose between an English, a German, a Greek, and a Hebrew keyboard layout through a gaze-controlled drop-down menu (available through the virtual button with the "globe" symbol on the lower panel of the virtual keyboard interface mode). The laptops dedicated for the field study were equipped with a myGaze-n eye trackers by Visual Interaction, which performs dark pupil tracking at a frequency of 30 Hz. GazeTheWeb started automatically after the startup of the laptop and offered the participant to perform a calibration of the eye tracker. Furthermore, if the participant eyes could not be detected for 30 seconds, the participants have been offered a

[25]http://www.internetlivestats.com/total-number-of-websites checked at 18th February 2019


(a) Greek layout with left to right direction of text.

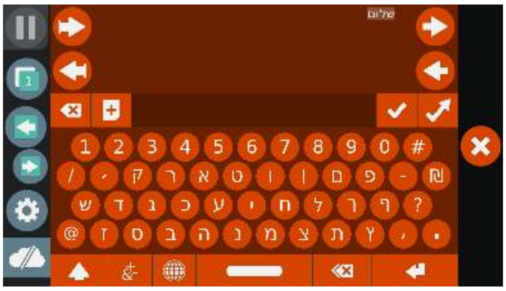

(b) Hebrew layout with right to left direction of text.

Fig. 15. GazeTheWeb has been localized to meet the constraints of the field study. The localization not only includes the screen texts of the interface but also the available keyboard layouts and text editing facilities.

recalibration upon return. The system automatically logged the Web browsing activity to a custom Firebase[26] real-time database, e.g., the system logged loaded Web pages and times, clicks, amount of inserted text, and general interface use.[27] The participants were able to disable all data recording by selecting a virtual button with a crossed cloud in the primary interface (see the virtual button beneath the "gear wheel" symbol at the bottom of left panel in Figure 15).

6.2.2 Procedure. Prior to the field study, each of the participants had been visited to verify if the eye tracking device would work with them. Afterwards, the systems had been placed for one month at the homes of the participants at an accessible place within their homes. At the time of deployment, a person from the medical supervisory team gave an introduction and initial guidance through the system in native language to the participant and care taker. After the deployment, participants were free to use the system as per their needs and preferences. The person from the medical supervisory team also provided telephone support in native language, if there were any issues during the usage. After one month usage, the system was collected back from the participant home, and the participants were asked to fill questionnaires including SUS to quantify their gaze interaction experience.

6.2.3 Results. GazeTheWeb ran successfully for the entire one month period, and we could observe the user activity through the log data on Firebase. More importantly, the participants could visit and interact with variety of Web sites as per their need and preferences, which signifies the usbability of GazeTheWeb in supporting everyday browsing operations and handling dynamic Web pages. An average SUS score of 73.2 among all participants indicates the general acceptability and satisfaction delivered by GazeTheWeb. In the following, we report on the usage behaviour of participants during a period of one month.

Overview. Over the course of the field study, we have collected data during a system run-time of 186.24 hours. Because we could check for the presence of participants from the gaze signals, we report about 118.93 hours of active participation in front of the system. There has been regular use of the system by some of the participants, as plotted in Figure 16. The participants have browsed to 456 unique domains, on which they have visited 8415 Web pages. See Table 2 for top ten domains as browsed by the participants. In total, there have been 8,027 clicks on Web pages and 22,811 characters have been entered in text inputs. Furthermore, participants have browsed to 498 URLs

[26]https://firebase.google.com
[27]Due to ethical reasons, we did not record the textual content itself, but stored the string-edit distances.

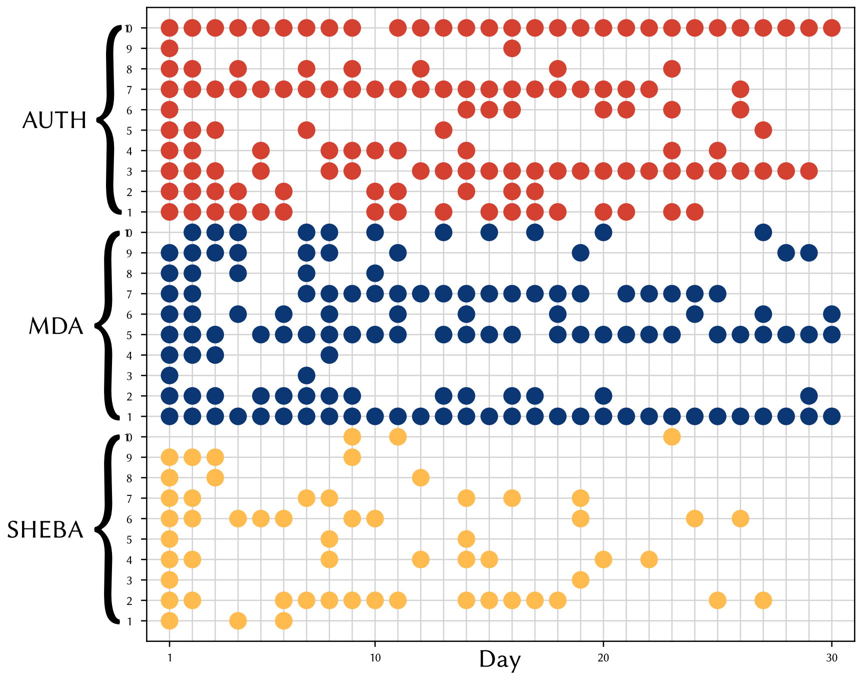


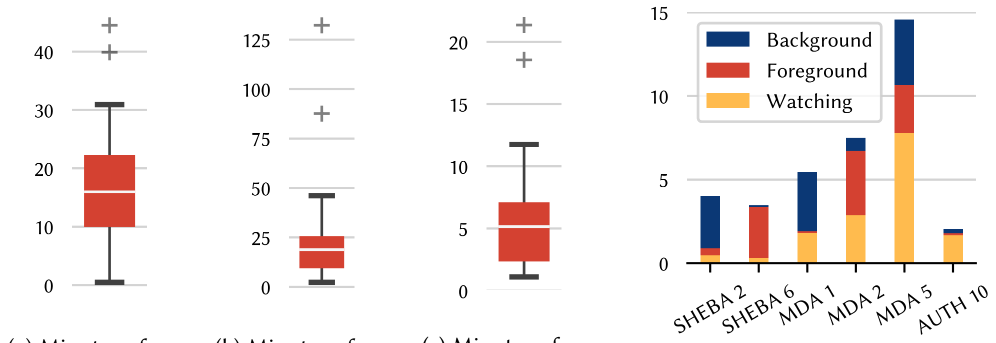
Fig. 16. Daily use of GazeTheWeb in the field study. The vertical axis of the plot shows the participants. The horizontal axis displays the days since the setup of the system at the homes of the participants. Each dot signifies at least on start of the system by a certain participant on a specific day.

by typing and added 189 bookmarks. They also made use of the multi-tab browsing through 857 tab switches. We recorded 1,455 recalibrations of the eye tracker device, i.e., not counting the initial calibrations.

Unsurprisingly, we found that the usage of GazeTheWeb follows a long-tailed distribution, with few participants being most active and most participants using GazeTheWeb to a lesser extent.

Table 2. Top 10 visited Web sites by participants. "Visits" are caused through URL input, navigation, bookmarks, or history use. "Stays" are visits that exceed an active time of one minute, during which the participant had been registered by the eye tracking device. "Pages" is the count of visited pages on the Web site. "Hours" is the time of browsing by a participant on the Web site. "duckduckgo.com" had been preset as search engine.

Rank	Domain	Visits	Stays	Pages	Hours
1	facebook.com	369	229	1208	24.3
2	youtube.com	271	203	1396	26.5
3	duckduckgo.com	235	37	483	2.1
4	google.gr	123	24	270	1.6
5	mail.google.com	100	66	774	4.0
6	accounts.google.com	71	7	258	0.5
7	instagram.com	54	34	125	1.4
8	google.com	54	12	136	0.8
9	twitter.com	52	9	65	0.8
10	newsit.gr	50	46	257	3.7



(a) Minutes of MDA 1 on Facebook

(b) Minutes of MDA 5 on YouTube

(c) Minutes of AUTH 10 on GMail

(d) Hours on YouTube

Fig. 17. Prominent uses of GazeTheWeb in the field study. (a) shows a box plot of daily times that MDA 1 was actively on Facebook, day count = 30. (b) shows a box plot of daily times that MDA 5 was actively on YouTube, day count = 19. (c) shows a box plot of daily times that AUTH 10 was actively operating an GMail, day count = 22. Values in (a)-(c) are provided in minutes. (d) shows the hours of different participants on YouTube video pages. Hereby, "Watching" is the time the eye tracking device had detected the participant in front of the screen; "Foreground" is the time the video has been visible but the participant has been absent; "Background" is the time another tab had been chosen visible.

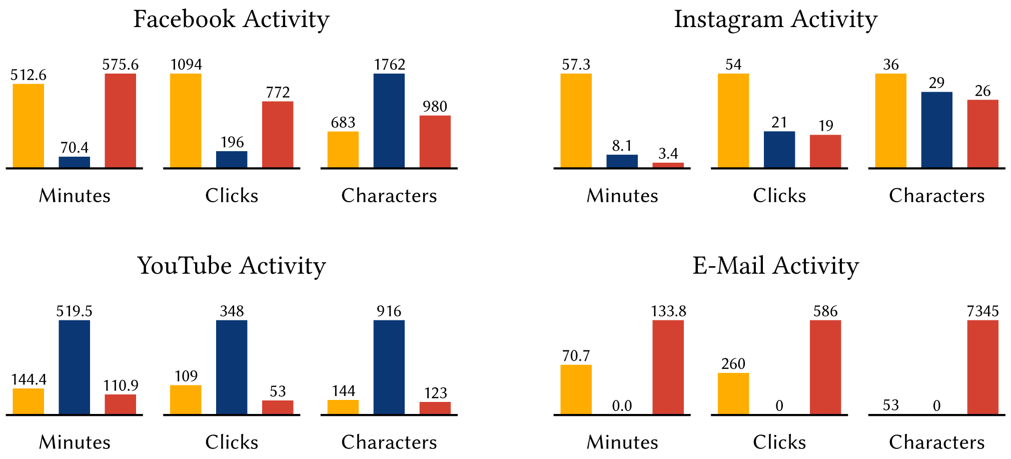


Fig. 18. Web browsing activity of 🟨 MDA 1, 🟦 MDA 5, and 🟥 AUTH 10 on popular platforms.

Less activity may come from a multitude of factors including (i) low motivation of participants in general (including depressive moods of severely impaired subjects), (ii) low interest in using a technical system in general, (iii) low interest in dealing with technical limitations of GazeTheWeb, or (iv) inability to perform some activity with GazeTheWeb. With this field study we do not target issues such as (i) or (ii), but we aim to understand practical limitations of GazeTheWeb. Therefore, we assume that the more active participants are better representatives when it comes to judging the feasibility of GazeTheWeb.

Most Active Participants. The most active users have been MDA 1 with 23.27 hours in front of the system, MDA 5 with 21.9 hours, and AUTH 10 with 27.75 hours. During the one month period, the participants used many different internet platforms with GazeTheWeb. Figure 17 plots the session times of each participant on their favourite platform on a daily basis. MDA 1 was very active on Facebook (Figure 17a), MDA 5 spent a lot of time on YouTube (Figure 17b), and AUTH 10 visited 29 times "mail.google.com", including 396 sub-pages, over 22 days (Figure 17c). In general, YouTube has been the second most visited domain and the highest number of spent hours. Some participants made use of the tab system in GazeTheWeb to browse the Web while a YouTube video was played in the background (Figure 17d).

Activities. The different use of platforms across the participants can be also observed in interactions on the platforms. In Figure 18, the active minutes, clicks, and characters inputted have been plotted. We have clustered E-Mail Web portals as E-Mail Activity, i.e., visits on "mail.ru", "live.com", "outlook.live.com", "mail.google.com", "mail.yahoo.com", or "mail.walla.co.il". The use of different platforms is very heterogeneous among the participants.

6.2.4 Discussion. The frequent usage of GazeTheWeb during the field study including the visits of various Web sites, pages, and the activities (e.g., clicks, text entry) indicates that users were able to interact with the variety of pages and to perform desired browsing operations. The longer stay duration and revisits of popular platforms like Facebook, GMail, and Youtube (incorporating complex interaction elements in a Facebook posting, an E-Mail body, video controls) implies that GazeTheWeb could effectively adapt the associated complex interaction elements for gaze interaction.

However, for the pages where the participants did not stay long or did not revisit, one could argue that interaction was difficult or not functional. For example, the gaming domains specifically received less interest among participants, such as "games.yo-yoo.co.il" with two visits and a stay duration 14 minutes, "freegames.com" with one visit for 1.8 minutes and "actiongame.com" with three visits and a stay duration 12.2 minutes. Ideally, the users would stay longer in these domains as the intent would be on playing the offered games. Action games incorporate specialized controls for game play with high frequency of mouse and keyboard input, which would be non-trivial to adapt for gaze interaction in the proposed methodology of GazeTheWeb. There has been only one stay at "docs.google.com", with a duration of 6.2 minutes, and nobody has visited Google Maps. We incorporate these Web sites in our discussion about current limitations in adaptation of interaction for eye gaze input in the next section.

7 LIMITATIONS IN ADAPTATION OF WEB BROWSING FOR EYE GAZE INPUT

The evaluation of GazeTheWeb shows that gaze-based interaction is feasible with today's available eye tracking hardware, and that gaze-adapted interfaces can offer good user experience. We have identified the Web as an environment in which Web standards like HTML, CSS, and JavaScript, make it possible to adapt the interaction with rich Web pages for eye tracking as input method. A vast majority of Web sites conform to the Web standards with respect to interaction elements. Hence, the introspection and adaptation mechanism of GazeTheWeb works effectively with most Web pages that compose their content from standard elements. Figure 19 includes the screenshots of GazeTheWeb displaying pages of Web sites from the Alexa Top 50 sites.[28] Here, (a) to (j) are examples of pages where interaction elements could be retrieved and adapted successfully through the general introspection and adaptation mechanism described in GazeTheWeb.

[28]https://www.alexa.com/topsites

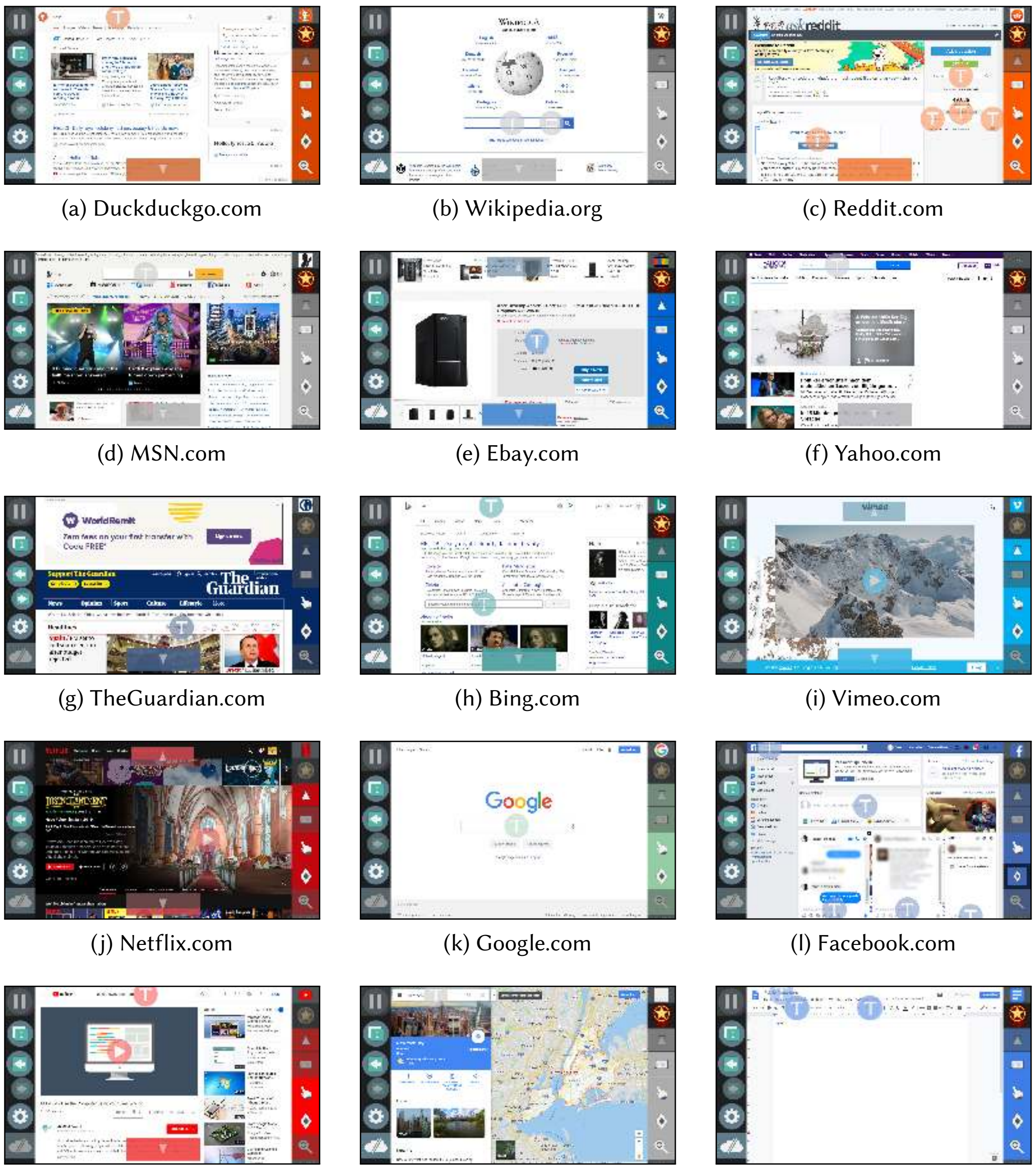

(a) Duckduckgo.com (b) Wikipedia.org (c) Reddit.com
(d) MSN.com (e) Ebay.com (f) Yahoo.com
(g) TheGuardian.com (h) Bing.com (i) Vimeo.com
(j) Netflix.com (k) Google.com (l) Facebook.com
(m) YouTube.com (n) Google Maps (o) Google Docs

Fig. 19. Various popular Web sites displayed in GazeTheWeb

7.1 Special Cases of Adaptation

To our experience, especially Web sites of some well-known internet companies make use of compound elements with manually scripted behaviour, which render a ubiquitous adaptation difficult. Hence, a few popular Web sites required additional engineering to retrieve the relevant interaction elements. In Figure 19, (k) Google.com, (l) Facebook.com, and (m) YouTube.com are examples of Web sites for which we modified our adaptation approach as described in the following.

Google Search. At the time of our assessment, there were at least two text inputs stacked onto each other for the search box on the Google start page. It appears that one text input is responsible to display suggestions in gray color and the other text input is used to display the actual user input in black color. The stacking is realized with the CSS z-index attribute that defines the order in rendering. Thus, we have introduced a special case for the Google start page to recognize the text input that has to be filled with the phrase for the search query. One might argue for a general solution to adapt interaction elements by sorting according to their z-index. However, it is not easy to figure out which parts of the interaction elements are presented in front, when there are multiple elements with only partially overlapping areas.

Facebook Chat. The Facebook chat prohibits insertion of text into the text input of the chat window through JavaScript. The text appears for a short time after insertion and then it is automatically removed. This could be a strategy to avoid bots from spamming automatically on the chat. Hence, we implemented an alternative approach for text insertion in a Facebook chat window. Instead of inserting text through JavaScript, we take the text as received from the eye typing interface mode and simulate keystrokes through the Web engine. We spawn keystrokes according to each character in the text, within a time window of a few milliseconds.

YouTube Video. There were two issues with video interaction. The first one we faced for videos in general, the second one is specific to the YouTube video platform.

First, the full-screen video function does not work via a direct call from JavaScript code. The `node.webkitRequestFullscreen()` mechanism employs the heuristic that the screen can not be entirely covered without the consent of the user. However, the request for full-screen does work when activated from a button click event by the user. Therefore, we create an invisible button in the DOM upon entering the video interface mode, and we attach the function to make the video full-screen to the click event of the invisible button. A click is simulated on the button, which makes the video full-screen. Afterwards, we remove the invisible button from the DOM.

Second, when the described procedure is executed on a video on the YouTube platform, the video stays only for a short time in full-screen and returns automatically to its page embedding. We were able to resolve this YouTube-specific issue by requesting full-screen not for the video itself, but for the seventh-grade parent of the video node. We suspect that YouTube wants to avoid code injections that make the video full-screen while removing the advertisement overlays. The advertisement overlays are properly displayed with our approach, even though they are not embedded into the video file.

7.2 Challenges for Adaptation

There are more standards defined, which might be considered to improve the adaptation even further, e.g., the `<nav>` tag,[29] which allows the identification of the menu bar of Web pages that could be used for in-depth menu adaptations for gaze-control. "Drag-n-Drop" has also been standardized[30] and, thus, is another candidate for adaptation in GazeTheWeb. Furthermore, there are numerous ARIA role[31] annotations available to include hints about the interface semantics in order to make elements more accessibility. We already use a small subset of AIRA roles in our classification of interaction elements as described in Figure 5. In the future we would extend the support of additional AIRA roles like "Grid", "Tree", or "Menubar".

[29]https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav
[30]https://www.w3.org/TR/2010/WD-html5-20101019/dnd.html
[31]https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Roles

We observed that some popular Web applications that offer specialized controls using non-standardized interaction mechanisms do not work appropriately with GazeTheWeb. The sites (n) Google Maps and (o) Google Docs in Figure 19 showcase this scenario. A general adaptation for these custom interface elements is not feasible, as they base on specific scripts and non-standardized interaction mechanisms. As a consequence, their accessibility is limited, i.e., the design hinders the adaptation of interaction for novel input devices. This issue is also evident in Web pages that attempt to prohibit bots to access certain areas of a page, e.g., "CAPTCHAS".[32] The latest versions of those mechanisms validate humans by their natural interaction through mouse and keyboard or touch events on Web sites. Neither gaze-based emulation nor our approach let users pass those validations and users have to choose alternative ways for validation.

More semantic descriptions of interfaces would be desirable for successful adaptions for gaze control, or other input channels. However, not all common elements are standardized or semantically annotated, and we would have to infer the possibilities of interaction by different ways. For highly customized interaction contexts (e.g., map navigation, document editing, game controls), we imagine a future research approach would employ automatic interaction-template matching, i.e., to cater different Web services with similar interaction behaviour. For example, there are common patterns of interactions for map navigation (panning and zooming), regardless of the Web site that is offering the corresponding service like Google Maps,[33] Bing Maps,[34] or Open Street Maps.[35] An intelligent interaction-context-recognition would detect the context of map navigation on these Web sites and offer a unified, gaze-adapted interface mode for map navigation. The interface mode would be the same for all map services, yet, spawn interaction events as expected by the different services. This might be realized with advanced interpretation of the JavaScript code and detection of the use of input events in the JavaScript code, which might be substituted with events caused from eye gaze.

8 CONCLUSION AND FUTURE WORK

Eye gaze has been explored as a communication channel for interaction between human and computer systems for at least 30 years. However, most research and application has focused on how to compensate for the errors in gaze estimation and how to deal with the dual roles of eyes to resolve the Midas Touch problem. Since mouse as pointing device and keyboard as typing device have dominated the access to computers, considerable effort has been put in replicating the functionality of both interaction means with eye tracking-based interaction. In the past, researchers and companies have taken gaze-control methods for these two interaction means and put them together into emulation approaches allowing computer access for people with severe motor impairment.

In this article, we have discussed how the emulation approach introduces an overhead in human inspection efforts and input actions. We put up propositions based on common principles for good user experience that aim to reduce the overhead of gaze-based interaction and visual search, taking the semantics of a controlled application's interface into account. We argue that the introspection in the Web environment allows us to adapt the interaction for eye gaze as input for a large number of use cases. To this end, we describe how to efficiently retrieve the interface semantics from a dynamic Web environment. Moreover, we provide details about our realization of the proposed principles in GazeTheWeb, a gaze-controlled Web browser that acts as bridge between Web and eye tracking environment. We utilize the proposed introspection method to retrieve the interface semantics of Web pages and adapt the interaction elements, as well as to adapt the interaction with the Web browser for eye tracking as input method. The proposed approach of using interface

[32]https://www.w3.org/TR/turingtest
[33]https://www.google.com/maps
[34]https://www.bing.com/maps
[35]https://www.openstreetmap.org

semantics to adapt interaction improves user experience and has not been explored in prior work of eye tracking-based interaction.

We have evaluated the user experience of GazeTheWeb in comparison to the emulation approach. The results on task completion time, and more importantly the subjective measures like the outcomes of SUS and NASA-TLX surveys and heuristics questionnaire, signify that our proposed interface adaptations in GazeTheWeb improve the user experience in comparison to the current emulation approaches. This contrasts with eye tracking research so far, which mostly assesses the speed and accuracy of isolated interactions like target acquisition with eye tracking compared to other modalities like mouse, keyboard, or touch. Furthermore, it emphasizes our contribution towards making gaze-based interaction not only useful but rather usable for daily use. We also validate the feasibility of GazeTheWeb in a lab and field study with the target users of eye tracking as an assistive technology. The overall successful task completions, positive feedback from our lab studies, substantiated by long term usage at a home environment, indicate that GazeTheWeb is effective in letting people with motor impairment execute everyday browsing tasks.

Takeaway

We argue that it is time for the eye tracking technology to step out of the laboratory setting and be explored as input technology for daily use. Current research is mostly focused on micro optimization of gaze-based interaction, e.g., finding the optimal dwell time, testing different selection methods, or adapting the size of interface elements. Often, the optimization of one aspect hinders the interaction in other terms. We conclude to shift the research perspective rather towards a macro optimization of gaze-based interaction. The success of everyday use of eye tracking will depend on stable calibrations, intuitive interaction paradigms, and rewarding user experience. This will shift eye tracking as input method beyond being an option for people with motor impairment into everybody's life. There is also a growing commercial interest towards eye tracking technology.[36],[37],[38] Then, GazeTheWeb might be an important step towards general use of eye tracking in everyday applications by everyone.

The principles we have outlined in this article do not only allow for using gaze as a stand-alone modality, but also to combine gaze control with other input modalities providing for intriguing potential to shape future computer-human interaction. There is research showing that interaction through eye tracking in combination with a trigger can be faster than a mouse alone [50]. In general, other modalities can remove the overload of inspection and selection from eye tracking and allow for an even more natural selection processes than traditional pointing devices. Especially eye tracking in combination with voice input does offer a huge potential, as both technologies compensate for each others shortcomings, i.e., gaze provides the spatial context of attention, and voice plays an important role in confirming users' intention.

In this regard, GazeTheWeb as an open-source browser[39] provides a framework for researchers to investigate methods for improved interaction with eye gaze and other input modalities in the Web environment. In this direction, the prototype of GazeTheWeb has already been used to integrate voice input and EEG for multimodal interaction in Web browsing environment [41, 72]. The effect of persuasive technology with artificial agent has been evaluated with GazeTheWeb [27]. Moreover, eye tracking researchers foresee the integration of their algorithms in GazeTheWeb for further improved interactions [17]. Our aim is to continue supporting such developments and avail GazeTheWeb with updates and improvements in future.

[36]https://techcrunch.com/2017/06/26/apple-acquires-smi-eye-tracking-company
[37]https://techcrunch.com/2016/12/28/the-eye-tribe-oculus
[38]https://techcrunch.com/2016/10/24/google-buys-eyefluence-eye-tracking-startup
[39]https://github.com/MAMEM/GazeTheWeb

ACKNOWLEDGMENTS

This work has been performed as part of the MAMEM[40] project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement number 644780. We also acknowledge the financial support by the Federal Ministry of Education and Research of Germany under the project number 01IS17095B. The authors would like to thank Korok Sengupta (Institute for Web Science and Technologies, University of Koblenz-Landau) for his assistance in conducting the lab studies at the university campus, and Daniel Müller (Active Vision Group, University of Koblenz-Landau) for the integration of the Web engine into GazeTheWeb. The authors would like to acknowledge the support from MAMEM project partners[41] for the successful execution of MAMEM studies. The authors are also very grateful to all the participants who took part in all evaluation studies and provided us with relevant feedback.

REFERENCES

[1] Kiyohiko Abe, Kosuke Owada, Shoichi Ohi, and Minoru Ohyama. 2008. A system for Web browsing by eye-gaze input. *Electronics and Communications in Japan* 91, 5 (2008), 11–18. https://doi.org/10.1002/ecj.10110

[2] Sheetal K. Agarwal, Anupam Jain, Arun Kumar, and Nitendra Rajput. 2010. The World Wide Telecom Web Browser. In *Proceedings of the First ACM Symposium on Computing for Development (ACM DEV '10)*. ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/1926180.1926185

[3] Deepak Ahya and Daniel Baudino. 2006. Method to enhance user interface and target applications based on context awareness. US Patent App. 10/853,947.

[4] Pierre A. Akiki, Arosha K. Bandara, and Yijun Yu. 2016. Engineering Adaptive Model-Driven User Interfaces. *IEEE Transactions on Software Engineering* 42, 12 (Dec 2016), 1118–1147. https://doi.org/10.1109/TSE.2016.2553035

[5] Michael Ashmore, Andrew T. Duchowski, and Garth Shoemaker. 2005. Efficient Eye Pointing with a Fisheye Lens. In *Proceedings of Graphics Interface 2005 (GI '05)*. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 203–210. http://dl.acm.org/citation.cfm?id=1089508.1089542

[6] Richard Bates and Howell Istance. 2002. Zooming Interfaces!: Enhancing the Performance of Eye Controlled Pointing Devices. In *Proceedings of the Fifth International ACM Conference on Assistive Technologies (Assets '02)*. ACM, New York, NY, USA, 119–126. https://doi.org/10.1145/638249.638272

[7] Wolfgang Beinhauer. 2006. A Widget Library for Gaze-based Interaction Elements. In *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications (ETRA '06)*. ACM, New York, NY, USA, 53–53. https://doi.org/10.1145/1117309.1117338

[8] Michael Bensch, Ahmed A Karim, Jürgen Mellinger, Thilo Hinterberger, Michael Tangermann, Martin Bogdan, Wolfgang Rosenstiel, and Niels Birbaumer. 2007. Nessi: an EEG-controlled web browser for severely paralyzed patients. *Computational intelligence and neuroscience* 2007 (2007).

[9] Ralf Biedert, Georg Buscher, Sven Schwarz, Manuel Möller, Andreas Dengel, and Thomas Lottermann. 2010. The Text 2.0 Framework: Writing Web-based Gaze-controlled Realtime Applications Quickly and Easily. In *Proceedings of the 2010 Workshop on Eye Gaze in Intelligent Human Machine Interaction (EGIHMI '10)*. ACM, New York, NY, USA, 114–117. https://doi.org/10.1145/2002333.2002351

[10] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. 2004. Semantic Pointing: Improving Target Acquisition with Control-display Ratio Adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 519–526. https://doi.org/10.1145/985692.985758

[11] John Brooke. 2013. SUS: A Retrospective. *Journal of Usability Studies* 8, 2 (Feb. 2013), 29–40. http://dl.acm.org/citation.cfm?id=2817912.2817913

[12] Brian Burg, Andrew J. Ko, and Michael D. Ernst. 2015. Explaining Visual Changes in Web Interfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software Technology (UIST '15)*. ACM, New York, NY, USA, 259–268. https://doi.org/10.1145/2807442.2807473

[13] George Candea, Mauricio Delgado, Michael Chen, and Armando Fox. 2003. Automatic Failure-Path Inference: A Generic Introspection Technique for Internet Applications. In *Proceedings of the The Third IEEE Workshop on Internet Applications (WIAPP '03)*. IEEE Computer Society, Washington, DC, USA, 132–141. http://dl.acm.org/citation.cfm?id=832311.837386

[14] Ellis Carolyn. 1991. Sociological Introspection and Emotional Experience. *Symbolic Interaction* 14, 1 (1991), 23–50. https://doi.org/10.1525/si.1991.14.1.23

[40] http://www.mamem.eu

[41] http://www.mamem.eu/project/consortium

[15] Scott Carter, Amy Hurst, Jennifer Mankoff, and Jack Li. 2006. Dynamically Adapting GUIs to Diverse Input Devices. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility (Assets '06)*. ACM, New York, NY, USA, 63–70. https://doi.org/10.1145/1168987.1169000

[16] Chia-Hui Chang, Mohammed Kayed, Moheb R. Girgis, and Khaled F. Shaalan. 2006. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering* 18, 10 (2006), 1411–1428.

[17] Zhaokang Chen and Bertram E. Shi. 2017. Improving Gaze-based Selection using Variable Dwell Time. *CoRR* abs/1704.06399 (2017). arXiv:1704.06399 http://arxiv.org/abs/1704.06399

[18] Albert M Cook and Janice Miller Polgar. 2014. *Assistive Technologies-E-Book: Principles and Practice.* Elsevier Health Sciences.

[19] Daniel K. Davies, Steven E. Stock, and Michael L. Wehmeyer. 2001. Enhancing Independent Internet Access for Individuals with Mental Retardation through Use of a Specialized Web Browser: A Pilot Study. *Education and Training in Mental Retardation and Developmental Disabilities* 36, 1 (2001), 107–113. http://www.jstor.org/stable/24481620

[20] Vagner Figueredo de Santana, Rosimeire de Oliveira, Leonelo Dell Anhol Almeida, and Marcia Ito. 2013. Firefixia: An Accessibility Web Browser Customization Toolbar for People with Dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility (W4A '13)*. ACM, New York, NY, USA, Article 16, 4 pages. https://doi.org/10.1145/2461121.2461137

[21] Antonio Diaz-Tula and Carlos H. Morimoto. 2016. AugKey: Increasing Foveal Throughput in Eye Typing with Augmented Keys. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3533–3544. https://doi.org/10.1145/2858036.2858517

[22] Morgan Dixon and James Fogarty. 2010. Prefab: Implementing Advanced Behaviors Using Pixel-based Reverse Engineering of Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 1525–1534. https://doi.org/10.1145/1753326.1753554

[23] Morgan Dixon, James Fogarty, and Jacob Wobbrock. 2012. A General-purpose Target-aware Pointing Enhancement Using Pixel-level Analysis of Graphical Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 3167–3176. https://doi.org/10.1145/2207676.2208734

[24] Tobii Dynavox. 2017. *Photograph of computer system with Tobii eye tracker and running Tobii Windows Control software.* http://www.tobiidynavox.de/wp-content/uploads/2016/06/TobiiDynavox_EyeMobileMini_front_-1030x687.png

[25] Anna Maria Feit, Shane Williams, Arturo Toledo, Ann Paradiso, Harish Kulkarni, Shaun Kane, and Meredith Ringel Morris. 2017. Toward Everyday Gaze Input: Accuracy and Precision of Eye Tracking and Implications for Design. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1118–1130. https://doi.org/10.1145/3025453.3025599

[26] Leah Findlater, Alex Jansen, Kristen Shinohara, Morgan Dixon, Peter Kamb, Joshua Rakita, and Jacob O. Wobbrock. 2010. Enhanced Area Cursors: Reducing Fine Pointing Demands for People with Motor Impairments. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 153–162. https://doi.org/10.1145/1866029.1866055

[27] Sofia Fountoukidou, Jaap Ham, Uwe Matzat, and Cees Midden. 2018. Using an Artificial Agent as a Behavior Model to Promote Assistive Technology Acceptance. In *Persuasive Technology*, Jaap Ham, Evangelos Karapanos, Plinio P. Morita, and Catherine M. Burns (Eds.). Springer International Publishing, Cham, 285–296. https://doi.org/10.1007/978-3-319-78978-1_24

[28] Krzysztof Z. Gajos. 2008. *Automatically generating personalized user interfaces.* University of Washington.

[29] Aryeh Gregor, Ms2ger, Alex Russell, Robin Berjon, and Anne van Kesteren. 2015. *W3C DOM4.* W3C Recommendation. W3C. http://www.w3.org/TR/2015/REC-dom-20151119/.

[30] Tovi Grossman and Ravin Balakrishnan. 2005. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 281–290. https://doi.org/10.1145/1054972.1055012

[31] Human Performance Research Group. 1988. Nasa Task Load Index (TLX): Paper and Pencil Package. http://humansystems.arc.nasa.gov/groups/tlx/downloads/TLX_pappen_manual.pdf. Accessed: 2nd May 2016.

[32] Vicki L. Hanson, Jonathan P. Brezin, Susan Crayne, Simeon Keates, Rick Kjeldsen, John T. Richards, Calvin Swart, and Shari Trewin. 2005. Improving Web Accessibility Through an Enhanced Open-source Browser. *IBM Syst. J.* 44, 3 (Aug. 2005), 573–588. https://doi.org/10.1147/sj.443.0573

[33] Katarzyna Harezlak, Pawel Kasprowski, and Mateusz Stasch. 2014. Towards Accurate Eye Tracker Calibration - Methods and Procedures. *Procedia Computer Science* 35 (2014), 1073 – 1081. https://doi.org/10.1016/j.procs.2014.08.194 Knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference, KES-2014 Gdynia, Poland, September 2014 Proceedings.

[34] Simon Harper and Yeliz Yesilada. 2008. *Web accessibility: a foundation for research.* Springer Science & Business Media.

[35] Eric Horvitz. 1999. Principles of Mixed-initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 159–166. https://doi.org/10.1145/302979.303030

[36] Visual Interactive. 2017. *myGaze Power catalogue.* http://www.mygaze.com/fileadmin/download/mygaze_power/myGaze_Power_catalogue.pdf

[37] Howell Istance, Richard Bates, Aulikki Hyrskykari, and Stephen Vickers. 2008. Snap Clutch, a Moded Approach to Solving the Midas Touch Problem. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications (ETRA '08).* ACM, New York, NY, USA, 221–228. https://doi.org/10.1145/1344471.1344523

[38] Rob Jacob and Sophie Stellmach. 2016. What You Look at is What You Get: Gaze-based User Interfaces. *interactions* 23, 5 (Aug. 2016), 62–65. https://doi.org/10.1145/2978577

[39] Robert J. K. Jacob. 1990. What You Look at is What You Get: Eye Movement-based Interaction Techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90).* ACM, New York, NY, USA, 11–18. https://doi.org/10.1145/97243.97246

[40] Søren Staal Jensen and Tina Øvad. 2016. Optimizing web-accessibility for deaf people and the hearing impaired utilizing a sign language dictionary embedded in a browser. *Cognition, Technology & Work* 18, 4 (01 Nov 2016), 717–731. https://doi.org/10.1007/s10111-016-0385-z

[41] Fotis Kalaganis, Elisavet Chatzilari, Spiros Nikolopoulos, Yiannis Kompatsiaris, and Nikos Laskaris. 2018. An error-aware gaze-based keyboard by means of a hybrid BCI system. *Scientific Reports* 8, 1 (2018). https://doi.org/10.1038/s41598-018-31425-2

[42] Ahmed A. Karim, Thilo Hinterberger, Jürgen Richter, Jürgen Mellinger, Nicola Neumann, Herta Flor, Andrea Kübler, and Niels Birbaumer. 2006. Neural internet: Web surfing with brain potentials for the completely paralyzed. *Neurorehabilitation and Neural Repair* 20, 4 (2006), 508–515.

[43] Melanie Kellar, Carolyn Watters, and Michael Shepherd. 2006. The Impact of Task on the Usage of Web Browser Navigation Mechanisms. In *Proceedings of Graphics Interface 2006 (GI '06).* Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 235–242. http://dl.acm.org/citation.cfm?id=1143079.1143118

[44] Kurt Koffka. 1924. Introspection and the method of psychology. *British Journal of Psychology* 15, 2 (1924), 149–161. https://doi.org/10.1111/j.2044-8295.1924.tb00170.x

[45] Chandan Kumar, Raphael Menges, Daniel Müller, and Steffen Staab. 2017. Chromium Based Framework to Include Gaze Interaction in Web Browser. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW '17 Companion).* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 219–223. https://doi.org/10.1145/3041021.3054730

[46] Chandan Kumar, Raphael Menges, and Steffen Staab. 2016. Eye-Controlled Interfaces for Multimedia Interaction. *IEEE MultiMedia* 23, 4 (Oct 2016), 6–13.

[47] Chandan Kumar, Raphael Menges, and Steffen Staab. 2017. Assessing the Usability of Gaze-Adapted Interface against Conventional Eye-Based Input Emulation. In *2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS).* 793–798. https://doi.org/10.1109/CBMS.2017.155

[48] Manu Kumar, Tal Garfinkel, Dan Boneh, and Terry Winograd. 2007. Reducing Shoulder-surfing by Using Gaze-based Password Entry. In *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS '07).* ACM, New York, NY, USA, 13–19. https://doi.org/10.1145/1280680.1280683

[49] Manu Kumar, Jeff Klingner, Rohan Puranik, Terry Winograd, and Andreas Paepcke. 2008. Improving the Accuracy of Gaze Input for Interaction. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications (ETRA '08).* ACM, New York, NY, USA, 65–68. https://doi.org/10.1145/1344471.1344488

[50] Manu Kumar, Andreas Paepcke, Terry Winograd, and Terry Winograd. 2007. EyePoint: Practical Pointing and Selection Using Gaze and Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07).* ACM, New York, NY, USA, 421–430. https://doi.org/10.1145/1240624.1240692

[51] Manu Kumar and Terry Winograd. 2007. GUIDe: Gaze-enhanced UI Design. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems (CHI EA '07).* ACM, New York, NY, USA, 1977–1982. https://doi.org/10.1145/1240866.1240935

[52] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos Morimoto, and Margrit Betke. 2016. EyeSwipe: Dwell-free Text Entry Using Gaze Paths. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16).* ACM, New York, NY, USA, 1952–1956. https://doi.org/10.1145/2858036.2858335

[53] Fabrizio Lamberti, Gianluca Paravati, Valentina Gatteschi, and Alberto Cannavo. 2017. Supporting Web Analytics by Aggregating User Interaction Data From Heterogeneous Devices Using Viewport-DOM-Based Heat Maps. *IEEE Transactions on Industrial Informatics* 13, 4 (2017), 1989–1999. https://doi.org/10.1109/TII.2017.2658663

[54] Chris Lankford. 2000. Effective Eye-gaze Input into Windows. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications (ETRA '00).* ACM, New York, NY, USA, 23–27. https://doi.org/10.1145/355017.355021

[55] Christof Lutteroth, Moiz Penkar, and Gerald Weber. 2015. Gaze vs. Mouse: A Fast and Accurate Gaze-Only Click Alternative. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software; Technology (UIST '15).* ACM, New York, NY, USA, 385–394. https://doi.org/10.1145/2807442.2807461

[56] Yu-Seung Ma, Jeff Offutt, and Yong-Rae Kwon. 2006. MuJava: A Mutation System for Java. In *Proceedings of the 28th International Conference on Software Engineering (ICSE '06)*. ACM, New York, NY, USA, 827–830. https://doi.org/10.1145/1134285.1134425

[57] I. Scott MacKenzie. 1992. Fitts' Law As a Research and Design Tool in Human-computer Interaction. *Hum.-Comput. Interact.* 7, 1 (March 1992), 91–139. https://doi.org/10.1207/s15327051hci0701_3

[58] I. Scott MacKenzie. 2012. Evaluating eye tracking systems for computer input. In *Gaze interaction and applications of eye tracking: Advances in assistive technologies*. IGI Global, 205–225.

[59] Jalal U. Mahmud, Yevgen Borodin, and I. V. Ramakrishnan. 2007. Csurf: A Context-driven Non-visual Web-browser. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*. ACM, New York, NY, USA, 31–40. https://doi.org/10.1145/1242572.1242578

[60] Päivi Majaranta. 2009. *Text entry by eye gaze*. University of Tampere.

[61] Päivi Majaranta, Hirotaka Aoki, Mick Donegan, Dan Witzner Hansen, and John Paulin Hansen. 2011. *Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies* (1st ed.). IGI Global, Hershey, PA.

[62] Jennifer Mankoff, Anind Dey, Udit Batra, and Melody Moore. 2002. Web Accessibility for Low Bandwidth Input. In *Proceedings of the Fifth International ACM Conference on Assistive Technologies (Assets '02)*. ACM, New York, NY, USA, 17–24. https://doi.org/10.1145/638249.638255

[63] Raphael Menges, Chandan Kumar, Daniel Müller, and Korok Sengupta. 2017. GazeTheWeb: A Gaze-Controlled Web Browser. In *Proceedings of the 14th Web for All Conference (W4A '17)*. ACM. http://dx.doi.org/10.1145/3058555.3058582

[64] Raphael Menges, Chandan Kumar, Korok Sengupta, and Steffen Staab. 2016. eyeGUI: A Novel Framework for Eye-Controlled User Interfaces. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction (NordiCHI '16)*. ACM, New York, NY, USA, Article 121, 6 pages. https://doi.org/10.1145/2971485.2996756

[65] Raphael Menges, Chandan Kumar, Ulrich Wechselberger, Christoph Schaefer, Tina Walber, and Steffen Staab. 2017. Schau genau! A Gaze-Controlled 3D Game for Entertainment and Education. In *Journal of Eye Movement Research*, Vol. 10. 220. https://bop.unibe.ch/JEMR/article/view/4182

[66] Raphael Menges, Hanadi Tamimi, Chandan Kumar, Tina Walber, Christoph Schaefer, and Steffen Staab. 2018. Enhanced Representation of Web Pages for Usability Analysis with Eye Tracking. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. ACM, New York, NY, USA, Article 18, 9 pages. https://doi.org/10.1145/3204493.3204535

[67] Jakob Nielsen and Rolf Molich. 1990. Heuristic Evaluation of User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*. ACM, New York, NY, USA, 249–256. https://doi.org/10.1145/97243.97281

[68] Spiros Nikolopoulos, Panagiotis C. Petrantonakis, Kostas Georgiadis, Fotis Kalaganis, Georgios Liaros, Ioulietta Lazarou, Katerina Adam, Anastasios Papazoglou-Chalikias, Elisavet Chatzilari, Vangelis P. Oikonomou, Chandan Kumar, Raphael Menges, Steffen Staab, Daniel Müller, Korok Sengupta, Sevasti Bostantjopoulou, Zoe Katsarou, Gabi Zeilig, Meir Plotnik, Amihai Gotlieb, Racheli Kizoni, Sofia Fountoukidou, Jaap Ham, Dimitrios Athanasiou, Agnes Mariakaki, Dario Comanducci, Edoardo Sabatini, Walter Nistico, Markus Plank, and Ioannis Kompatsiaris. 2017. A multimodal dataset for authoring and editing multimedia content: The MAMEM project. *Data in Brief* 15 (2017), 1048 – 1056. https://doi.org/10.1016/j.dib.2017.10.072

[69] Dan R. Olsen, Jr., Sean Jefferies, Travis Nielsen, William Moyes, and Paul Fredrickson. 2000. Cross-modal Interaction Using XWeb. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST '00)*. ACM, New York, NY, USA, 191–200. https://doi.org/10.1145/354401.354764

[70] Marco Porta and Alessia Ravelli. 2009. WeyeB, an Eye-controlled Web Browser for Hands-free Navigation. In *Proceedings of the 2nd Conference on Human System Interactions (HSI'09)*. IEEE Press, Piscataway, NJ, USA, 207–212. http://dl.acm.org/citation.cfm?id=1689359.1689396

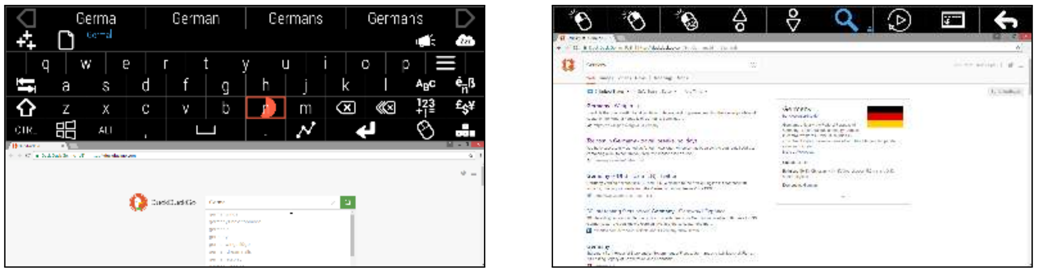
[71] Simon Schenk, Marc Dreiser, Gerhard Rigoll, and Michael Dorr. 2017. GazeEverywhere: Enabling Gaze-only User Interaction on an Unmodified Desktop PC in Everyday Scenarios. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3034–3044. https://doi.org/10.1145/3025453.3025455

[72] Korok Sengupta, Min Ke, Raphael Menges, Chandan Kumar, and Steffen Staab. 2018. Hands-free Web Browsing: Enriching the User Experience with Gaze and Voice Modality. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications (ETRA '18)*. ACM, New York, NY, USA, Article 88, 3 pages. https://doi.org/10.1145/3204493.3208338

[73] Korok Sengupta, Chandan Kumar, and Steffen Staab. 2017. Usability Heuristics for Eye-controlled User Interfaces. 19th European Conference on Eye Movements. http://cogain2017.cogain.org/camready/poster3-Sengupta.pdf

[74] Korok Sengupta, Raphael Menges, Chandan Kumar, and Steffen Staab. 2017. GazeTheKey: Interactive Keys to Integrate Word Predictions for Gaze-based Text Entry. In *Proceedings of the 22Nd International Conference on Intelligent User Interfaces Companion (IUI '17 Companion)*. ACM, New York, NY, USA, 121–124. https://doi.org/10.1145/3030024.3038259

[75] Korok Sengupta, Jun Sun, Raphael Menges, Chandan Kumar, and Steffen Staab. 2017. Analyzing the Impact of Cognitive Load in Evaluating Gaze-based Typing. In *30th IEEE International Symposium on Computer-based Medical Systems*,



(a) Keyboard mode passes inputted text as emulated keyboard events to the active application.

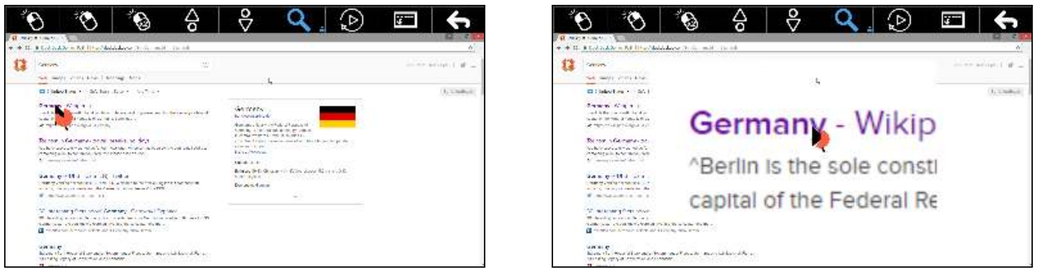
(b) Mouse mode features various mouse events like left click (first virtual button from left) and scrolling (forth and fifth virtual button from left).

Fig. 20. Two modes of OptiKey are used in the experiments. First, the keyboard mode (active after startup of OptiKey) enables text input on a virtual keyboard. Second, the mouse mode that features emulation of common mouse events.

Vol. Special Track on Multimodal Interfaces for Natural Human Computer Interaction: Theory and Applications. IEEE.
[76] Ben Shneiderman. 1997. *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (3rd ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[77] Laurianne Sitbon, Oscar Wong, and Margot Brereton. 2014. Efficient Web Browsing with a Single-switch. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design (OzCHI '14)*. ACM, New York, NY, USA, 515–518. https://doi.org/10.1145/2686612.2686693
[78] Jiguo Song and Gabriel Parmer. 2013. Toward Predictable, Efficient, System-level Tolerance of Transient Faults. *SIGBED Rev.* 10, 4 (Dec. 2013), 53–56. https://doi.org/10.1145/2583687.2583700
[79] Wolfgang Stuerzlinger, Olivier Chapuis, Dusty Phillips, and Nicolas Roussel. 2006. User Interface Façades: Towards Fully Adaptable User Interfaces. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, New York, NY, USA, 309–318. https://doi.org/10.1145/1166253.1166301
[80] Julius Sweetland. 2016. Optikey: Type, Click, Speak. https://github.com/OptiKey/OptiKey.
[81] Jingtao Wang and Jennifer Mankoff. 2002. Theoretical and Architectural Support for Input Device Adaptation. *SIGCAPH Comput. Phys. Handicap.* 73-74 (June 2002), 85–92. https://doi.org/10.1145/960201.957220
[82] Benjamin Wassermann, Adrian Hardt, and Gottfried Zimmermann. 2012. Generic Gaze Interaction Events for Web Browsers Using the Eye Tracker as Input Device. In *Proceedings of the 2012 Workshop on Emerging Web Technologies at Conference on World Wide Web*. https://doi.org/10.13140/2.1.1871.9362
[83] Tim Weninger, Rodrigo Palacios, Valter Crescenzi, Thomas Gottron, and Paolo Merialdo. 2016. Web Content Extraction: A MetaAnalysis of Its Past and Thoughts on Its Future. *SIGKDD Explor. Newsl.* 17, 2 (Feb. 2016), 17–23. https://doi.org/10.1145/2897350.2897353
[84] Xuebai Zhang, Xiaolong Liu, Shyan-Ming Yuan, and Shu-Fan Lin. 2017. Eye Tracking Based Control System for Natural Human-Computer Interaction. *Computational Intelligence and Neuroscience: CIN* (2017).
[85] Xuan Zhang and I. Scott MacKenzie. 2007. Evaluating Eye Tracking with ISO 9241 - Part 9. In *Proceedings of the 12th International Conference on Human-computer Interaction: Intelligent Multimodal Interaction Environments (HCI'07)*. Springer-Verlag, Berlin, Heidelberg, 779–788. http://dl.acm.org/citation.cfm?id=1769590.1769678

A INTERACTION WITH OPTIKEY

The OptiKey interface consists of emulation panels with different modes of emulation, i.e., mouse and various virtual keyboards that emulate different kinds of physical keyboards. For the experimental setup, only the standard QWERTY keyboard (Figure 20a) and mouse (Figure 20b) modes were used. A user can switch between both modes with a virtual button in the emulation panel. In the following, we describe the interaction procedures required to fulfill the browsing tasks with OptiKey, which were executed in the lab study at our university.



(a) First dwell time to initiate click position.

(b) Magnified screen as second step in mouse click emulation for more accurate pointing.

Fig. 21. After selection of the "left mouse click" virtual button on the left of the emulation panel, clicking is performed in two steps. First, the user fixates the region to click. After a dwell time of one second, the region is magnified and displayed in the center of the screen. A second fixation with dwell time is necessary to choose the final point of click. The progress of dwell time is visualized by the filling pie next to the the big black arrow, which indicates the position of fixation.

Navigation. Navigation within a Web browser is mostly performed via the backwards functionality or by hyperlinks. A user can operate these functions of the Google Chrome browser via mouse emulation. In the mouse mode of the emulation panel, the "left mouse click" virtual button needs to be selected to initiate the pointing process. Then, a mouse pointer is displayed at the detected fixation upon the interface of the active application. After a certain duration of the fixation, the content beneath the gaze is magnified and presented in the center of the screen as pop up overlay. After another dwell time on the pop up, a click is performed at the magnified position. The additional magnification step brings the pointing accuracy to a level that enables reliable selection of a link that is represented by text in standard size. See Figure 21 for the process of clicking.

Scrolling. Scrolling on a desktop computer is performed with the scrolling wheel on the physical mouse. OptiKey adopts this paradigm and offers options to emulate the scrolling wheel in its mouse mode. To perform an upwards scrolling emulation, the virtual button for scrolling up has to be selected. Respectively, the virtual button for scrolling down has to be selected to scroll down in the application interface. Then, the user has to fixate the position in the interface of the active application where the scroll command should be executed. After a dwell time at the fixated position, the scroll emulation is performed. When the user needs to scroll on the same position multiple times, another virtual button is available to repeat the last executed action.

Text Input. In the keyboard mode, virtual buttons representing keys on the virtual keyboard are displayed in the interface. The QWERTY layout has been chosen for the experimental setup.

Bookmarking. To add the current page as bookmark in Google Chrome, the user has to perform a click on the star, which is placed in the interface of Google Chrome right to the URL bar. We also asked the participants to finish the task by a second click on the "Done" button within the bookmark pop up.

For accessing the available bookmarks, a procedure of multiple click emulations must be performed. In Google Chrome, the bookmarks are accessible through a sub menu of the general menu. It becomes visible after clicking at the three dots, next to the star for adding the current page as bookmark. In total, three clicks have been necessary to access a saved bookmark, two for menu navigation and the third for selection.

B TASKS FOR MAMEM FIRST PHASE TRIALS

The dictated tasks for the MAMEM first phase trials are denoted in the following. Required online accounts had been provided by the experimenter.

B.1 E-Mail Task

(1) User is asked to go to tab overview
(2) User is asked to add a new tab
(3) User is asked to go to the `Gmail.com` via manually typing of the URL
(4) User is asked to sign into `Gmail.com`
(5) User is asked to read a received E-Mail
(6) User is asked to respond to an E-Mail by writing “hello world”
(7) User is asked to send the E-Mail
(8) User proceeds to task B.2

B.2 Photo Task

(1) User is asked to go to tab overview
(2) User is asked to go to edit URL
(3) User is asked to visit the bookmark overview
(4) User is asked to go to the `Picresize`.com bookmark
(5) User is asked to choose a sample picture of choice on the Web page
(6) User is asked to rotate the picture 90 degrees counter clockwise
(7) User is asked to choose a special effect of choice
(8) User is asked to choose “I’m Done, Resize My Picture!”
(9) User proceeds to B.3

B.3 Social Media Task

(1) User is asked to go to tab overview
(2) User is asked to add a new tab
(3) User is asked to visit the bookmark overview
(4) User is asked to go to the `Twitter.com` bookmark
(5) User is asked to search for “MAMEM Project”
(6) User is asked to select the MAMEM project page
(7) User is asked to follow the MAMEM project
(8) User is asked to post a text message on MAMEM project’s page
(9) User proceeds to B.4

B.4 Video Task

(1) User is asked to go to tab overview
(2) User is asked to add a new tab
(3) User is asked to visit the bookmark overview
(4) User is asked to go to the `YouTube.com` bookmark
(5) User is asked to search for “gazetheweb mamem”
(6) User is asked to select the first video
(7) User is asked to pause the video
(8) User is asked to play again the video
(9) User is asked to close the tab with the YouTube page
(10) End of dictated tasks

C STATEMENT OF PREVIOUS RESEARCH

GazeTheWeb has been demonstrated at two events, the 14th International Web for All conference 2017 (W4A '17)[42] and the demo programme of The Web Conference 2017 (WWW '17), where it both times received critical acclaim and awards. The first demonstration was accompanied with an extended abstract (2 pages) and the second with a short paper (5 pages), details about these two short papers are given below. Neither of these two previous short papers discussed what constitutes the core scientific contributions of this paper, i.e., *(i)*, the issue of user experience in eye tracking-based interaction and a means to improve user experience, *(ii)*, the discussion of this issue in related literature, *(iii)*, the sophisticated adaptation of interface elements for gaze-based interaction, and, *(iv)*, the evaluation of GazeTheWeb in several user studies. These novel contributions described in this paper are not under submission to another venue or journal, and have not been previously submitted or published in a conference or journal.

More specifically, the two short papers presented the following contents:

- The prior work "GazeTheWeb: A Gaze-Controlled Web Browser" ([63], 2 pages) has demonstrated the initial prototype of the GazeTheWeb system together with some user pre-tests with healthy participants, only. The comprehensive analysis of eye tracking-based interaction, propositions to reduce interaction overhead, adaptations in GazeTheWeb, and evaluation studies are unique to the current submission, and render this submission our major publication developing and studying GazeTheWeb.
- The prior work "Chromium based Framework to Include Gaze Interaction in Web Browser" ([45], 5 pages) demonstrated the extraction method from a dynamic Web environment and hence relates to the technical aspects of "retrieving interface semantics" in Section 4.1. Although the underlying Chromium framework is the same, the approach to retrieve interface semantics with classification and tracking of interaction element is unique to current submission. Furthermore, the sophisticated adaptations via interface semantics for different interaction element types have not been explored in this or other prior work.

Finally, the user study described and presented in Section 6.1 has been performed as part of a multi-topic study [68]. This overall data collection has been presented as a data publication, the sole purpose of which is the documentation and archiving of gathered data for later analysis, but excluding details about purpose and motivation for the individual sections of the multi-topic study and without any interpretation and discussion of gathered data.

[42] http://www.w4a.info/2016/2017