Assessing the Usability of Gaze-Adapted Interface against Conventional Eye-based Input Emulation

Chandan Kumar[1], Raphael Menges[1] and Steffen Staab[1,2]
[1]*Institute for Web Science and Technologies (WeST)*
University of Koblenz–Landau, Germany
[2]*Web and Internet Science Research Group (WAIS)*
University of Southampton, UK
Email: {kumar, raphaelmenges, staab}@uni-koblenz.de

***Abstract*—In recent years, eye tracking systems have greatly improved, beginning to play a promising role as an input medium. Eye trackers can be used for application control either by simply emulating the mouse and keyboard devices in the traditional graphical user interface, or by customized interfaces for eye gaze events. In this work, we evaluate these two approaches to assess their impact in usability. We present a gaze-adapted Twitter application interface with direct interaction of eye gaze input, and compare it to Twitter in a conventional browser interface with gaze-based mouse and keyboard emulation. We conducted an experimental study, which indicates a significantly better subjective user experience for the gaze-adapted approach. Based on the results, we argue the need of user interfaces interacting directly to eye gaze input to provide an improved user experience, more specifically in the field of accessibility.**

***Keywords*-eye tracking; gaze input; eye-controlled interfaces; social networks; user-centered design; usability analysis**

I. Introduction

One of the major goals of novel interaction techniques is to enable physically impaired people to be more socially integrated. In todays social life, online platforms like Facebook, Google Plus or Twitter play an important role in sharing experiences of one's life with other people [1]. Novel interaction techniques can help users with severe disabilities to become more active in sharing content and support communication with their friends and family through social networks. These platforms act like an open window to the world for disabled individuals – if they are able to use such services. Most social network platform interfaces like modern apps or webpages are built for today's conventional input methods like mouse and keyboard or touch, restricting the proper access for physically disabled people. Eye-based input cannot simply replace physical variants, since the gaze is both used for perception and interaction [2]. The methods for computer control by eye-based input can be divided into two main categories: either eye tracking is used to emulate physical input devices, e.g., mouse and keyboard in the normal graphical user interface [3], [4], [5], or the second approach, to execute desired interaction by direct interpretation of input by eye movements in a customized interface [6], [7], [8]. In this work, we follow the second approach and demonstrate a gaze-adapted Twitter application, which provides the social browsing functionality in a completely gaze-controllable interface. Furthermore, our goal is to evaluate the usability of proposed interface which represents a modern approach to gaze-based interaction mechanism.

Evaluation of gaze-controlled interactions lacks a universal benchmark; the comparison with physical mouse and keyboard input sources would be biased, and comparison with other assistive technologies would be only valid for a specific target user group. Most of the current approaches follow a trivial approach of qualitative questionnaire to judge the usability [9]. However, it is rather interesting to investigate how different approaches of gaze interaction advance the impact and end-user experience. In this work, we propose a comparative evaluation of gaze-adapted interfaces with a more conventional approach of gaze-based emulation. In the conventional emulation approaches, eyes replicate mouse and keyboard functionality, and user can perform all essential computer applications (e.g., social media browsing).

II. Eye-Controlled Applications

Modern remote camera-based eye tracker devices determine eye movement by (a) identification of the eyes in the video data stream, (b) the application of an eye model in order to determine eye orientation and gaze direction, and (c) a mapping of the inferred gaze onto the screen coordinates [10], [11]. However, the accuracy is not yet sufficient to identify the exact pixel position of users' gaze. Eye trackers can only determine the angle of the user's gaze up to around 0.5 degrees, which is about 10 pixels on a 17 inch monitor in 60 cm distance [6]. Gaze-controlled applications that need selection of a rather exact pixel coordinate – as provided by mouse pointer devices – employ different kinds of zooming approaches [10], [12]. In addition, there is an uncertainty about the user's gaze on whether it should be interpreted as intended interaction or as orientation and reading attempt, commonly known as the *Midas Touch* problem [13]. Dwell time has been introduced as the most common technique to tackle this, i.e., if the user's fixation exceeds a predefined threshold, it is classified as intended action.

To operate the conventional computer applications (developed for mouse, keyboard or touch input) by gaze,

emulation software like Tobii Dynavox Windows Control[1] or myGaze Assistive[2] are available as commercial systems. There exist open source alternatives, like OptiKey [5], which work effectively with the newest generation of low-cost eye trackers. These applications utilize dwell time and zooming effects to emulate both mouse and keyboard events on operating system level. Such emulation approaches work for most applications, since the low-level input of mouse and keyboard is almost always supported for computer software. However, it is independent from the software development framework of applications, e.g., rendering techniques and internal data structures which might be critical for gaze-adapted paradigms are ignored. For example, eye gaze pointing may benefit from the semantic information, which areas on the screen are interactive. Furthermore, manual switching between mouse and keyboard emulation needs additional interaction overhead.

For the users, it might be imperative to interact with gaze-adapted applications that respond directly to their gaze without an overhead of mouse and keyboard emulation. For the gaze responsive design, such a native application would have control on the underlying data structures and interaction possibilities. There have been several approaches of building gaze responsive elements to adapt the native application, e.g., eye-controlled gaming [14], eye typing [7], gaze-based image retrieval [15]. However, until now there have been no specific studies to compare the impact and usability of such gaze-adapted interface. It is stimulating to analyze, whether the difference between a gaze-adapted interface and by emulation executed applications is significant. In this work, we have developed a gaze-adapted Twitter application[3] and compared it to the mobile Twitter page displayed in Mozilla Firefox[4], controlled with emulation of mouse and keyboard using OptiKey software. Our prototype is open-source and available at GitHub[5].

III. Gaze-Adapted Twitter

According to the global social media research summary of 2016 [16], Twitter is one of the most popular social networks. Users can share their thoughts in short messages limited to 140 letters. The concept of tweeting, retweeting and following in Twitter is rather simple, and a feasible option for social communication via gaze-based interaction. Moreover, Twitter API[6] provides the most suitable programming interface to send and access information. This is critical to create a custom application including all prominent features users need to use this social platform. In comparison

[1]https://www.tobiidynavox.com/software/windows-software/windows-control

[2]http://www.mygaze.com/products/mygaze-assistive

[3]https://youtu.be/NQQfB7nf3qw

[4]https://www.mozilla.org/en-US/firefox

[5]https://github.com/MAMEM/GazeTheWeb/tree/master/Tweet

[6]https://dev.twitter.com/rest/public

Figure 1. Personal wall containing the latest tweets by people whom the user follows. The tweets are presented in the centered Content Area and actions available for last fixated tweet are presented in the vertical Action Bar on the right side of the screen. On top one can navigate through the different screens with a global navigation menu.

Facebook Graph API[7] does not allow to edit friends, create events or exchange direct messages between users. Google Plus API[8] offered no functions to write postings, although it seems to be available at the time this paper is written. The public Twitter API is only restricted in terms of user registration and the usual call limit within a time interval.

A. Design and Functionality

Design and positioning of interactive elements is a significant aspect of gaze-adapted applications. Unintended activation of interaction confuses users and has extremely negative impact on user experience. Moreover, they need passive areas for resting their gaze [2]. In the gaze-adapted Twitter interface, the login page (Figure 2) features only three buttons (aka sensitive fields queuing remaining dwell time visually to a user) to trigger the entry of username, password and the connection request to Twitter. The two initially named buttons summon a dwell time based keyboard (Figure 3) to enter the requested string, namely username or password. The keyboard design follows the common QWERTY key layout to be comparable to the keyboard emulation included in OptiKey. The keyboard's dwell time is adaptable by users through the activation of the double arrow buttons on the upper right of the layout. After connection with the user's Twitter account, the personal wall containing tweets of people the user is following is presented in the center of the screen (Figure 1), called *Content Area*. When the tweet at the bottom is selected, automatic vertical scrolling is executed to center the currently viewed tweet and to display older ones. Fixation of the tweet at top triggers a similar behavior, as long newer tweets are available. Highlighting the currently selected tweet in darker gray supports the

[7]https://developers.facebook.com/docs/graph-api

[8]https://developers.google.com/+/web/api/rest

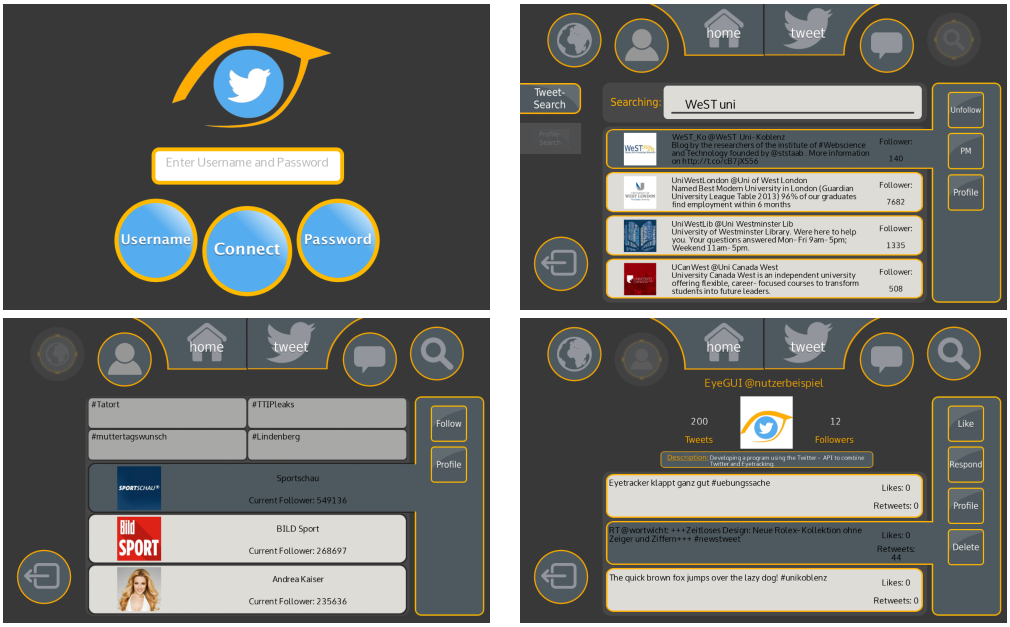

Figure 2. Further screens in the application. From upper left to lower right: Login Screen, Search Page, Discover Page and Profile Page

visual guidance of the user. Available actions on the tweet are shown in the so-called *Action Bar*. The Action Bar presents contextual actions on the selected tweet within the Content Area, so a user can scroll through the personal tweets without triggering unintentional action while looking at the Content Area. There are several interactions possible with the selected tweet on the Action Bar, e.g., `retweet`, `respond`, `like` or `unlike` and to go to the profile of the person who posted or retweeted this tweet. To emphasize the contextual awareness of the Action Bar, it is visually merged with the selected tweet inside the Content Area.

There are five other screens in addition to the personal wall, which can be accessed through the global navigation bar on the top:

- Personal wall
- Send a tweet
- Discover trends
- Visit own user profile
- Private messaging for direct user communication
- Search and view other profiles

It can be seen in Figure 2 that the other screens of the application like search or discover share the concept of Content Area for Twitter content and Action Bar that offers contextual actions.

Theses interfaces are developed through user centered design for eye interaction, i.e. interface components such as size, shape, appearance, feedback etc., which are vital to enhance eye tracking accuracy for input control [6]. For example, sizes of buttons are enlarged, their placement is based upon element usage frequency and application control, so natural viewing behavior does not interfere. All elements are designed especially for eye tracking in their size and the way the user interacts with them. Several alternatives for eye gaze input like blinking, pupil dilation [2] were explored, however dwell time based selection has been the most natural and widely accepted mode of input.

B. Implementation

Twitter offers a public *Representational State Transfer* (REST) API[9] for developers to access a user's information and to submit tweets and messages. For this, the user has to activate communication over this REST API[10], which is included in our application at first successful login[11]. The API follows the *OAuth*[12] protocol. There are various libraries[13] available to access the REST API in a convenient way in different programming languages. Since our application is written in C++ language, we decided to delegate the *twitcurl*[14] library for all communication with the social network. It supports many necessary calls from the v.1.1 API and has been extended on some features by manual request. This can be easily achieved since twitcurl itself delegates the communication to the *curl*[15] library. Result of all requests is a JSON string that is parsed into a DOM object tree by *RapidJSON*[16] for easier access.

[9]https://dev.twitter.com/rest/public
[10]https://apps.twitter.com
[11]https://dev.twitter.com/oauth/application-only
[12]https://dev.twitter.com/oauth
[13]https://dev.twitter.com/overview/api/twitter-libraries
[14]https://github.com/swatkat/twitcurl
[15]https://curl.haxx.se
[16]http://rapidjson.org

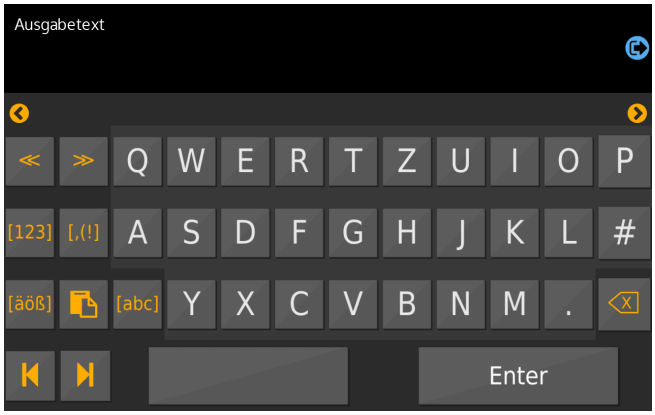

Figure 3. Dwell time based keyboard with QWERTY layout, featuring adaptable dwell time, adjusted manually by the user.

The interface elements were implemented using eyeGUI library [17], which supports the development of interactive gaze interfaces with many vital aspects, like rendering, layout, dynamic modification of content, support of graphics and animation.

IV. Evaluation

We conducted an experiment to evaluate the significance and usability of the proposed application with the state of art mouse and keyboard emulation software OptiKey. This is an open source software that can replace mouse, allowing to click, scroll and drag by eye movements. Moreover, OptiKey offers a virtual gaze-controlled keyboard as alternative to a physical one, enabling dwell time based typing into any application. Figure 4 shows the OptiKey usage scenario in Mozilla Firefox browser, considered for the experiment.

Both applications were evaluated at university campus, where 13 participants (10M, 3F) aged between 20-39 took part in the experiment. The participants had no prior experience with eye tracking. Both applications were controlled with a low-cost Tobii EyeX[17] device on a 24 inch screen with a resolution of 1280x800 pixels. Dwell time was set for both setups to a time of one second and the dictionaries for word predictions were equalized. Additionally, the default pie-chart dwell time visualization in OptiKey has been replaced by a grow-effect, to match the one of our gaze-adapted Twitter application.

A. Procedure

The participants had to perform representative tasks with the system while being observed by supervisors. Two supervisors were present, one introduced the project and assisted the participant through the tasks. The other one observed the participant's reactions and feedback. Each participant was instructed on how the experimental process will be carried out in a short training session. A brief introduction to the systems and a calibration process with the eye tracker were presented in a short tutorial. Participants were asked to perform specific tasks representing the most common social media usage: Write a tweet and publish it; find a particular user and follow her; find and like a certain tweet about specified topic; explore the application like you would do for social media browsing (5-10 minutes).

The tasks pose challenges to the participants in using the keyboard, searching for profiles and using the `follow` and `like` feature. These tasks inherently represent essential control paradigms on traditional applications, e.g. mouse movement, click, scroll, navigation etc. The independent and the control variables were carefully noted prior to the experimental process. For counterbalancing, i.e., to negate the system bias with tasks, some participants started performing tasks first with gaze-adapted Twitter, while the others started with the OptiKey software. The usability questionnaires (described below) were asked for each system, immediately after the respective experimental session.

System Usability Scale (SUS) questionnaire [18] has been used to measure the usability of applications. The SUS contains 10 questions with five point likert scale from *strongly disagree* to *strongly agree*. The mental workload of the participants is measured by the NASA-TLX standardized questionnaire [19], [20]. The *NASA Task Load Index* (NASA-TLX) contains six components: mental, physical, temporal demands, frustration, effort and performance. For each component, the participant can decide the most applicable scores on a scale from 1 (low) to 100 (high) in 5-point steps. The participants were also asked to put weights on each scale by comparing all scales pairwise. The benefit of weighting is to increase the meaning of more relevant scales. Additionally, participants were asked to fill out a custom questionnaire regarding the functionality, handling and design of the corresponding application.

Based on the idea of *Discount Usability Testing* by Jacob Nielsen, the evaluation study was designed on a low-budget basis [21]. Instead of video recording, the participants were invited to use the *thinking aloud* method. This method delivers insight into the thoughts of the participants and replaces an expensive acquisition of equipment.

B. Results

Figure 5 shows the results of SUS evaluation for gaze-adapted Twitter and OptiKey, which clearly indicate high acceptance of gaze-adapted Twitter among participants. It achieved a score of 72, which is above the average score of 68 as per SUS guidelines[18], while OptiKey only achieves a result of almost 50, which is considered as the lowest grade. With the paired t-test, there was a significant difference in the scores for gaze-adapted twitter (M=71.92, SD=12.87) and OptiKey (M=49.8, SD=14.12), conditions; t(12)=3.498,

[17]https://tobiigaming.com/product/tobii-eyex

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

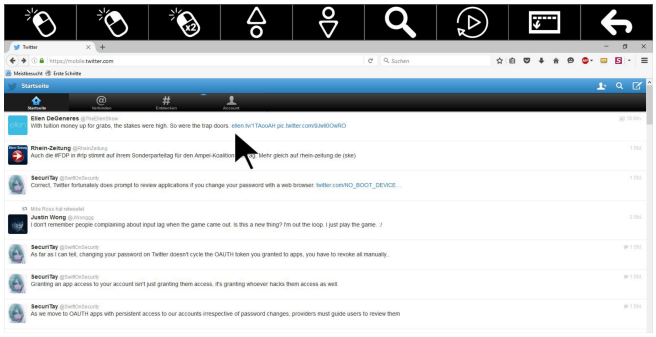

Figure 4. Mobile Twitter webpage displayed with Mozilla Firefox and controlled with OptiKey emulation software.

p=.0044 (95% CI 8.340 to 35.891). These results indicate that the usability of gaze-adapted Twitter is higher than conventional approach of eye-based emulation. Furthermore users explicitly provided feedback on the proposed interface being consistent, better orientation and less erroneous. Despite being a novel interface, a direct link with eye tracker seems much more intuitive and convenient to end users, in contrast the mouse-based emulation was identified cumbersome interaction process.

Figure 6 shows the average scores of the participants on NASA-TLX scales, which indicates that the emulation based software imposes higher task load on users in comparison to gaze-adapted interface. More specifically, in the pairwise weighting of NASA-TLX scales, mental demand, effort and frustration were judged as the most relevant scales by the participants. For these scales, gaze-adapted interface exhibit much lower score (as shown in figure 6) representing less effort required by users. Specially on the scale of frustration a big difference between the systems can be observed (scored and weighted about two times higher for OptiKey). This indicates that using the eye movement to emulate mouse and keyboard increases the frustration as the users have to imagine the control with traditional input devices and execute the emulation. The overall task load index of the

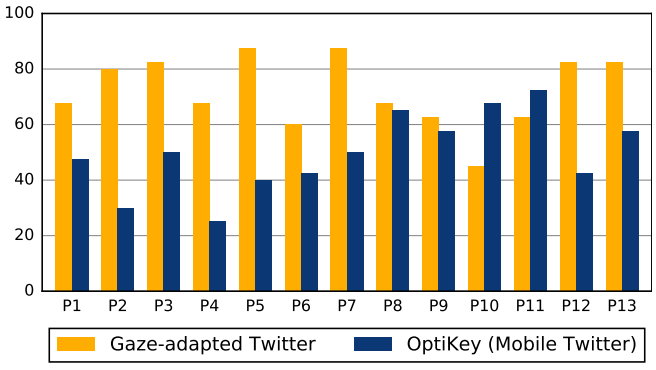

Figure 5. System Usability Scale scores

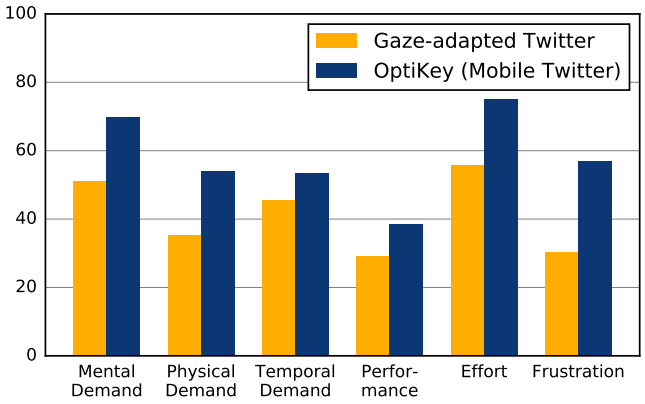

Figure 6. NASA-TLX average scores

systems were 63.8 and 46.7 for OptiKey and gaze-adapted Twitter respectively. With the paired t-test, there was a significant difference in the scores, conditions; t(12)=2.5879, p=.0238 (95% CI 31.469 to 2.701).

In addition the comparative study results, the participants shown some behavior patterns (observed by the supervisors) which provides relevant guidelines for gaze-based interfaces. For example the participants felt stressed when the interface reacted constantly to their gaze, and when they found a place in the interface to rest the eyes, a stress reduction was observed. Hence the placement of interactive elements is an important aspect. Another significant aspect is the role of feedback, i.e., while performing input by the gaze, users are very focused on the visual search task and overlook system's help (e.g., auto text suggestions while typing). This indicates that the feedback mechanism should be integrated in the visual search space of a particular task. Furthermore, the participants prefer the option to personalize the interaction with respect to their expertise, e.g., the typing speed of the keyboard (variation of the dwell time).

V. Conclusion

Several individuals are affected by severe physical deficiencies, which significantly limit their ability to communicate with the world. To tackle this problem, numerous studies have been carried out to create effective technologies. Among these, eye tracking has emerged as a promising and affordable technique. Eye tracking can be a great asset for social media access, however the conventional approaches or eye mouse emulation do not deal with the complex design challenges of eye-controlled interfaces. We proposed a gaze-adapted interface that supports unobtrusive gaze-based interaction with Twitter. We evaluated the proposed interface against OptiKey, which is a conventional approach to control computer applications via mouse and keyboard emulation. We reported that gaze-adapted interface is regarded as intuitive and easily interpretable for end users in the prominent social media browsing tasks. Despite of

its novelty, it performed well on usability analysis and required less mental demand from participants. The results imply that gaze-adapted applications with a direct control of gaze events must be considered for enhanced usability and performance of eye tracking input environment.

The presented system could perform all essential operations, furthermore, we are currently working on additional enhancements to include sophisticated gaze interaction features such as secure login/password entry, and better image/video interaction. In the future we want to enhance the usability by combining gaze with other modalities such as touch, speech, and brain-computer interfaces.

ACKNOWLEDGMENT

We would like to acknowledge the efforts from the students who participated in the research project[19] and had a major role in prototype development and evaluation. Furthermore, the work is supported by MAMEM[20] that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement number: 644780.

REFERENCES

[1] S. A. Hall, "The social inclusion of people with disabilities: A qualitative meta-analysis." *Journal of ethnographic & qualitative research*, vol. 3, no. 3, 2009.

[2] R. Jacob and S. Stellmach, "What you look at is what you get: Gaze-based user interfaces," *interactions*, vol. 23, no. 5, pp. 62–65, Aug. 2016.

[3] R. Bates and H. O. Istance, "Why are eye mice unpopular? a detailed comparison of head and eye controlled assistive technology pointing devices," *Universal Access in the Information Society*, vol. 2, no. 3, pp. 280–290, 2003.

[4] C. Lutteroth, M. Penkar, and G. Weber, "Gaze vs. mouse: A fast and accurate gaze-only click alternative," in *Proceedings of the 28th Annual ACM Symposium on User Interface Software; Technology*, ser. UIST '15. New York, NY, USA: ACM, 2015, pp. 385–394.

[5] "Optikey: Type, click, speak. v2.2.4," https://github.com/OptiKey/OptiKey/wiki.

[6] C. Kumar, R. Menges, and S. Staab, "Eye-controlled interfaces for multimedia interaction," *IEEE MultiMedia*, vol. 23, no. 4, pp. 6–13, Oct 2016.

[7] I. S. MacKenzie and X. Zhang, "Eye typing using word and letter prediction and a fixation algorithm," in *Proceedings of the 2008 Symposium on Eye Tracking Research; Applications*, ser. ETRA '08. New York, NY, USA: ACM, 2008, pp. 55–58.

[8] R. Menges, C. Kumar, D. Müller, and K. Sengupta, "Gazetheweb: A gaze-controlled web browser," in *Proceedings of the 14th Web for All Conference*, 2017.

[9] M. Porta and A. Ravelli, "Weyeb, an eye-controlled web browser for hands-free navigation," in *Proceedings of the 2Nd Conference on Human System Interactions*, ser. HSI'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 207–212.

[10] P. Majaranta and A. Bulling, "Eye tracking and eye-based human–computer interaction," in *Advances in physiological computing*. Springer, 2014, pp. 39–65.

[11] M. Kumar, J. Klingner, R. Puranik, T. Winograd, and A. Paepcke, "Improving the accuracy of gaze input for interaction," in *Proceedings of the 2008 symposium on Eye tracking research & applications*. ACM, 2008, pp. 65–68.

[12] C. Kumar, R. Menges, D. Müller, and S. Staab, "Chromium based framework to include gaze interaction in web browser," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 219–223.

[13] R. J. K. Jacob, "The use of eye movements in human-computer interaction techniques: What you look at is what you get," *ACM Trans. Inf. Syst.*, vol. 9, no. 2, pp. 152–169, Apr. 1991.

[14] C. Schaefer, R. Menges, K. Schmidt, M. Kuich, and T. Walber, "Schau genau! an eye tracking game with a purpose," in *Applications for Gaze in Games*, 2014.

[15] L. Kozma, A. Klami, and S. Kaski, "Gazir: gaze-based zooming interface for image retrieval," in *Proceedings of the 2009 international conference on Multimodal interfaces*. ACM, 2009, pp. 305–312.

[16] D. Chaffey, "Global social media research summary 2016," *Smart Insights: Social Media Marketing*, 2016.

[17] R. Menges, C. Kumar, K. Sengupta, and S. Staab, "eyegui: A novel framework for eye-controlled user interfaces," in *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, ser. NordiCHI '16. New York, NY, USA: ACM, 2016, pp. 121:1–121:6.

[18] "System usability scale (sus)," http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html, accessed: 2016-05-02.

[19] K. Vertanen, "Nasa-tlx in html and javascript," https://www.keithv.com/software/nasatlx/nasatlx.html, accessed: 2016-05-02.

[20] H. P. R. Group, "Nasa task load index (tlx): Paper and pencil package," http://humansystems.arc.nasa.gov/groups/tlx/downloads/TLX_pappen_manual.pdf, accessed: 2016-05-02.

[21] J. Nielsen, "Usability 101: Introduction to usability," 2003.

[19] https://west.uni-koblenz.de/en/news/talks/gazetheweb-tweet-your-eyes

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