Multimedia **A**uthoring and **M**anagement using your **E**yes and **M**ind

H2020-ICT-2014 - 644780

D3.3 - Final implementation of the control paradigms of multi-modal interaction

Dissemination level:	Confidential (CO)
Contractual date of delivery:	Month 32, 31/12/2017
Actual date of delivery:	Month 33, 08/01/2018
Work package:	WP3 - Signal Processing for Interaction Control
Task:	T3.1 - Algorithms for gaze-based interaction with an eye-tracker
	T3.2 - Algorithms for mind-based interaction with an EEG recorder
	T3.3 - Algorithms for stress detection via bio-measurements
	T3.4 - Novel paradigms for multi-modal interaction
Type:	Prototype
Approval Status:	Final
Version:	0.5
Number of pages:	76
Filename:	D3.3_multimodal_interaction

Abstract: Deliverable D3.3 describes the analysis and implementations of algorithms for different interaction paradigms of MAMEM multimodal framework. The methodology for gaze, EEG, and GSR based interaction, experimental evaluations, and its impact in MAMEM pilot trials have been presented.

Figure 1: co-funded by the European Union

Copyright

History

Version	Date	Reason	Revised by
v0.1(alpha)	10/09/2017	Table of Contents Checked for completeness by the consortium	Elisavet Chatzilari, Chandan Kumar and Spiros Nikolopoulos
v0.2(beta)	01/12/2017	Added Content for internal review	Chandan Kumar, Elisavet Chatzilari, Georgios Liaros, Kostas Georgiadis, Vangelis Oikonomou
v0.3	31/12/2017	Incorporated reviewer comments	Chandan Kumar, Elisavet Chatzilari, Dario Comanducci, Jaap Ham
v0.4	03/01/2017	Updated prototype documentations	Chandan Kumar, Raphael Menges, Elisavet Chatzilari, Georgios Liaros
v0.5(Final)	08/01/2017	Final editing and submission	Chandan Kumar, Spiros Nikolopoulos

Author List

Organization	Name	Contact Information
CERTH	Spiros Nikolopoulos	nikolopo@iti.gr
CERTH	Elisavet Chatzilari	ehatzi@iti.gr
CERTH	Fotis Kalaganis	fkalaganis@iti.gr
CERTH	Vangelis Oikonomou	viknmu@iti.gr
CERTH	Georgios Liaros	geoliaros@iti.gr
CERTH	Kostas Georgiadis	kostas.georgiadis@iti.gr
CERTH	Katerina Adam	katadam@iti.gr
CERTH	Yiannis Kompatsiaris	ikom@iti.gr
UNI KO-LD	Chandan Kumar	kumar@uni-koblenz.de
UNI KO-LD	Raphael Menges	raphaelmenges@uni-koblenz.de
UNI KO-LD	Korok Sengupta	koroksengupta@uni-koblenz.de
UNI KO-LD	Steffen Staab	staab@uni-koblenz.de

Abbreviations and Acronyms

BCI	Brain Computer Interface
DoA	Description of Actions
GUI	Graphical User Interface
HCI	Human Computer Interface
NUI	Natural User Interface
MMI	Multi Modal Interface
SSVEPs	Steady State Visual Evoked Potentials
SMR	SensoriMotor Rhythm
ErrPs	Error Related Potentials
ERPs	Event Related Potentials
MDS	Multi Dimensional Scaling
EDA	Electro-Dermal Activity
SCL	Skin Conductance Level
SCR	Skin Conductance Response
SVM	Support Vector Machines
ERD	Event-Related Desynchronization
ERS	Event-Related Synchronization

Executive Summary

The objective of this deliverable is to describe the necessary algorithms for translating bio-signals (i.e. eye-tracking based, EEG and GSR signals) into meaningful control variable for effective interaction. In that direction, we present the analysis and implementations of different interaction modalities and then bring them together with fusion techniques as Multi Modal Interfaces. In the fusion techniques, eye-based control is considered as the primary mode of interaction, supported by additional modalities for enhanced user experience.

Contents

List of Figures

List of Tables

1 Introduction

People affected by motor impairments, such as incurred by Parkinson, neuromuscular diseases or spinal-cord injuries, face severe problems concerning information access and digital communication. MAMEM seeks to enhance the accessibility of these people who cannot use conventional modes of interaction like mouse, keyboard etc. Signals from eyes and brain can make the interaction smoother and more effective than before. It will help people go beyond the conventional methods of input and expand the usability of these interaction systems. Since eye tracking provides a natural way to navigate computer systems, this has become the primary mode of interaction in MAMEM, which would be complemented with additional modalities. We identified Web as a major platform to enhance accessibility, since it can help motor-impaired people to connect and communicate with friends and society, which is the major objective of MAMEM.

Eye tracking systems have greatly improved in recent years, and have become a viable and affordable option as communication channel, especially for people lacking fine motor skills. Hence, it has great potential to enable these users to access the vast information source on the World Wide Web. However, input by eye-gaze is challenging due to issues with accuracy and ambiguity of eye-gazing gestures. Therefore, interaction-context specific optimizations for eye tracking input are required for an efficient usage and satisfying experience for the users. In MAMEM, we argue for the need to optimize the interaction in order to achieve a more intuitive integration of eye-gaze input. In Section 2 we deduce the shortcomings of the state of the art gaze-based emulation approach and put up guidelines for an improved gaze-based interaction. Furthermore, we propose the method and implementation of GazeTheWeb that follows these guidelines by adapting the gaze interaction in Web browsing environment. We present the lab study results, where GazeTheWeb accomplished almost all designated tasks significantly faster in comparison to the traditional method of hardware emulation, and achieved higher usability and satisfaction among the users. Moreover, we present the results from MAMEM first phase trials which signifies the applicability of the direct gaze-based interaction for the target user group.

While eye-tracking offers a natural interaction interface, controlling a computer with eyes only cannot reach 100% accuracy level. Erroneous clicks are expected due to the known shortcomings of gaze based interaction (e.g. Midas Touch problem). Our objective in MAMEM is to complement the gaze-based interaction through the incorporation of brain signals, transforming in this way GazeTheWeb in a multi-modal interface that alleviates these shortcomings. Towards this goal and considering that typing is one of the most intensive interactions, our first multi-modal function for the MAMEM system is an error-aware keyboard that automatically corrects the gaze-based typing errors (Section 3). Therefore, we optimize the detection of errors through EEG signals (i.e. ErrPs) by developing a spatial filtering approach that maximizes the SNR of the signals. Then, we analyze the data from the Pilot I Trials, and based on this analysis we optimize the keyboard interface so as to elicit time-locked ErrPs. Finally, we present our novel method for detecting eye-typing errors through the combination of EEG (ErrPs) and gaze data and validate its performance in a set of lab experiments with the newly designed keyboard.

Next, we analyze the SMR data that were captured during the Pilot I trials, showing that the motor-impaired participants perform significantly better than their respective able-bodied control group participants. This can be attributed to the fact that due to the loss of their fine-motor skills, they try to make imaginary movements in their every-day life, which can be seen as a form of "training". Nevertheless, user training for SMR is considered imperative to achieve sufficient accuracy. In this direction, we present MM-Tetris, a reinvention of the popular Tetris game, that can serve as a gamified environment to train the users' brains to produce distinguishable SMR signals. In designing MM-Tetris, we have utilized all three sensors (eye-tracking for controlling tetrimino horizontal movements, EEG through SMR for rotating the tetriminos and GSR through stress detection for adjusting tetrimino movement speed). Our focus has been in transforming the cue-based SMR detection (i.e. the onset of the imaginary

movement is known by providing the user with a cue to start the movement), which is the typical case in the literature, to self-paced detection of imaginary movements (i.e. the user is free to perform imaginary movements at their own pace and the system must detect it in an asynchronous way without knowing the movement's onset) through SMR.

For the stress detection through GSR measurements, we provide an analysis of the Pilot I trial data in Section 4. More specifically, we have applied the algorithm presented in D3.2 to the data and we demonstrate the processing results step-by-step by visualizing the outputs of the algorithm.

After discussing the components and algorithms for different modalities of the MAMEM system, in Section 5 we provide the multimodal interaction paradigms that will be available to the end users. More specifically, the rationale behind the MAMEM system is to rely on the more advanced gaze-based interaction through GazeTheWeb and complement it with EEG and GSR-based functionalities in order to alleviate its shortcomings. In that regard we propose three multimodal interaction paradigms, i.e., the error aware keyboard, the reading and selection mode switch, and the MM-Tetris game.

2 Gaze-based interaction and analysis

Interaction by eye-gaze is challenging due to accuracy and ambiguity issues, and gaze specific optimizations are required for a better user experience. In this chapter, we first describe the major challenges of gaze-based interaction for application control. Then we summarize how different gaze specific optimizations could help overcoming these challenges. Furthermore, we discuss how the existing approaches lack these optimizations aspects, and propose how a gaze-adapted approach can resolve these shortcomings and provide a superior user experience for gaze interaction. We implement this optimization in GazeTheWeb, evaluate its performance in a lab study, and report on its feasibility from MAMEM first phase results.

2.1 Challenges of Gaze-based Control

Various factors influence gaze estimation of remote eye tracking systems. Users may move the head, change their visual angle in relation to the tracking device or wear visual aid distorting the recorded eye image. Further limiting factors are ambient lighting, the sensor resolution of the utilized camera or the chosen calibration algorithm.

During tracking, these factors result in an uncertainty that is modeled as *gaze jitter* (or precision) and *gaze drift* (or accuracy) [21]. To deal with gaze jitter in the context of interaction, various filtering approaches for online gaze signal smoothing [21, 37] have been successfully introduced (in the context of MAMEM, we discuss the filtering approach in deliverable D2.3). Gaze drift is tackled by adaptive designs like enlarged and screen centered interface elements, or visual feedback [9, 33, 36].

Furthermore, the user is performing a double-task with their visual perception for gaze interaction. In addition to the usual exercise of inspection and orientation, gaze acts as indicator for selection at the fixated screen position. Unintended interaction via gaze is called *Midas Touch* [31], after the myth where a king transforms everything he touches into gold and is threaten by hunger as even his food turns into gold. Several alternatives have been explored to distinguish between inspection and selection, like blinking or pupil dilation [29]. However, dwell time based selection has been the most natural and widely accepted mode of input to solve the conflict of eye gaze acting both as sensor and controller [30, 46].

2.1.1 Limitations of Existing Approaches

The current gaze-based assistive softwares emulate the traditional input devices of mouse and keyboard to include eye gaze as an interaction medium. The approach has existed for several years, e.g., the *Eye-gaze Response Interface Computer Aid* (ERICA) system [39] was incepted in 1983 and included gaze-control mechanisms, featuring mouse and keyboard emulation at operation system level. Today's commercial systems like Tobii Dynavox Windows Control[1] or myGaze Power[2] incorporate similar principles. Recently, Microsoft has integrated gaze-controlled mouse and keyboard emulation for Windows.[3] Moreover, there exist open source alternatives, like OptiKey [1], which works with the last generation of low-cost eye trackers.

These approaches share a conceptual structure that is depicted Figure 2. The remote eye tracking device tracks the eyes of the user in front of the monitor and transfers the estimated gaze data to the emulation software. To compensate the precision limitations of the gaze estimation, a real-time filtering is applied on the data points. A panel with triggers is rendered to a window on the screen. The triggers are sensitive to the gaze as they are activated after a fixation exceeds a preset dwell time. The panel is presented to the user on top of the other windows in the graphical interface of the operation system. The gaze control options grouped

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

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

[3] https://blogs.msdn.microsoft.com/accessibility/2017/08/01/from-hack-to-product-microsoft-empowers-people-with-eye-control-for-windows-10

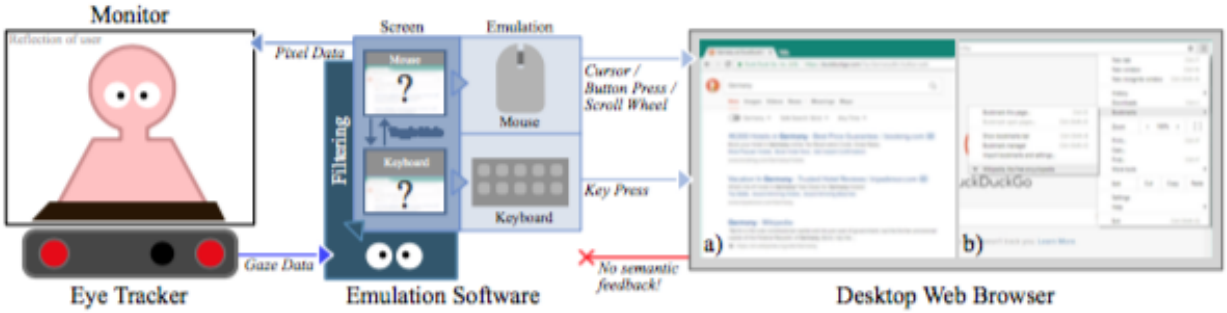


Figure 2: The emulation approach takes gaze input from the eye tracker, translates it via an overlay interface (shares screen-space with the active application) to mouse or keyboard commands and inputs these in the active application. In context of a Web browser, the pointing via mouse can be a) e.g., utilized for interface interaction (back navigation), navigation to select a link (link navigation) or scrolling of the page. Through this mechanism also b) hierarchical menus are accessible, although tedious to use.

by the device they are able to emulate, as in the case of OptiKey [1] there is one mode providing mouse emulation and another mode providing keyboard emulation. As screen space is limited, it is not feasible to put all possible traditional interaction functionalities into a single panel. Switching between both modes is itself achieved via a trigger that is activated through the dwell time mechanism. Commands issued via triggers in these modes are translated to operation system level hardware commands like a cursor movement in the mouse mode or a key press in the keyboard mode. The focused application (in case of Web browsing a desktop Web browser) receives these commands as if they were originating from a physical hardware device and the application acts like controlled by the traditional input devices for both webpage interaction a) and native interface selection b), in case of a desktop Web browser. There is no feedback provided for the emulation software by the application. This approach lacks in adapting the application menu for relevant functions for gaze-based interaction. Although the panel with the triggers itself is designed for gaze interaction, the indirectly controlled application is not. As mouse pointing is precise and fast, desktop applications make use of nested menus with small entries. This is quite a challenge for gaze-based pointing [52], why emulation software utilizes a multi-step magnification or multi-step refinement for selection.

Activation of actions like scroll wheel emulation is performed by dwelling the corresponding trigger within the panel. Since there is no feedback from the application, the user has to switch her fixation onto the controlled application to check for the effect. There is no general way for semantic feedback in the context of existing approaches.

Another issue for these approaches is the switching between emulation methods, e.g. mouse and keyboard mode of OptiKey. As the emulation software has no contextual awareness of the controlled application, it cannot serve the appropriate emulation facility automatically. The user has to decide which device is necessary to emulate in order to execute a command.

2.2 Optimizing Gaze Interaction

To overcome the above-mentioned limitations, there have been various approaches that can be categories in either the gaze signal processing or the interaction optimization. The two constraints of sufficient size for triggers to compensate for gaze drift and dwell time to distinguish selection from inspection are the standard for gaze-based interaction systems. However these constraints affect the usability and makes the interaction a tedious process for end-users. Therefore, we discuss the following optimization principles, i.e., how interaction could be enhanced for improved usability and to provide a faster and smooth experience.

2.2.1 Functional Relevance

The application interface menu consists of various triggers assigned to a functionality. Common examples are tool-bars in office suites or the window list in the graphical interface of an operation system. When making a menu available for gaze interaction, the size of the triggers must compensate for gaze drift and the dwell time must be chosen so that no unintended activation is initiated by the user. The required trigger size results in limited number of triggers on the screen. Fixing this issues with nested hierarchies requires multiple dwell times, as every button activation requires at least an one-dwell-time-long fixation by the user. Hence it is imperative to identify the most prominent functions of an application and present them to the user as top level menu within the menu layout rather than in a sub menu.

2.2.2 Semantic Feedback

For gaze-based selection, the user is required to fixate on the trigger to activate it via the dwell time mechanism. However, the trigger usually only provides local feedback about the feature activation itself, but not about the effect in the environment. Hence, the user might be required to repeatedly shift attention between the area of actual attention and the triggers to cause the desired effect. The additional cost of perceptual and cognitive load, caused by shifting the focus and repeated scanning of the environment [45] accounts for low usability. This can be improved by providing information about the effect on the trigger, as shown by Istance *et al.* [28], so the user can decide on further actions without shifting her attention.

2.2.3 Contextual Awareness

Application interfaces provide the user with proper means of interaction. In an optimal case, the interaction facilities are adapted to the context they are used in, like a number input that only allows for numbers. This can be extrapolated to gaze interaction, as the input channel transmits only a single fixation coordinate. Various methods for translating this limited input into higher level interactions like cursor positioning or text input are available and optimized for the required context. A gaze-controlled interface should be aware of the current usage context and might offer proper input translation methods like specialized number input to utilize gaze as input in the most convenient way.

2.3 Gaze adaptation for Web

Based on the derived optimization principles, we propose the design of a gaze-adapted Web browser. Figure 3 depicts the setup and data flow between the components. Similarly to the existing approaches of emulation, gaze is estimated by a remote eye tracking device (1) and filtered in real-time (2) to reduce gaze jitter. Gaze is utilized as input for dwelling on triggers within the user interface of the gaze-adapted Web browser (3), which does not consist of a single panel with triggers but an interface entirely designed for gaze interaction. The native interface of the browser for tab management b) consists of trigger which are directly linked to their labeled functionality like marking a bookmark or editing the URL. For the screen including the actual webpage a), all important actions like going back and forward can be placed nearby. Interaction trigger like for text input can be placed directly over the input field on the webpage. These screens are presented exclusively to the user within single window (4). By triggering actions within in the interface regarding the webpage interaction, commands are send to the Web engine that provides the webpage (5). The engine performs the requested command like change of an URL or a text entry and parses the page for adaptable elements, which are passed back to the gaze-adapted browser (7). These identified elements and information are then used for the indicated trigger overlays on the page and further adaptations (8). This does make the webpage appear *to be designed for gaze-based interaction.*

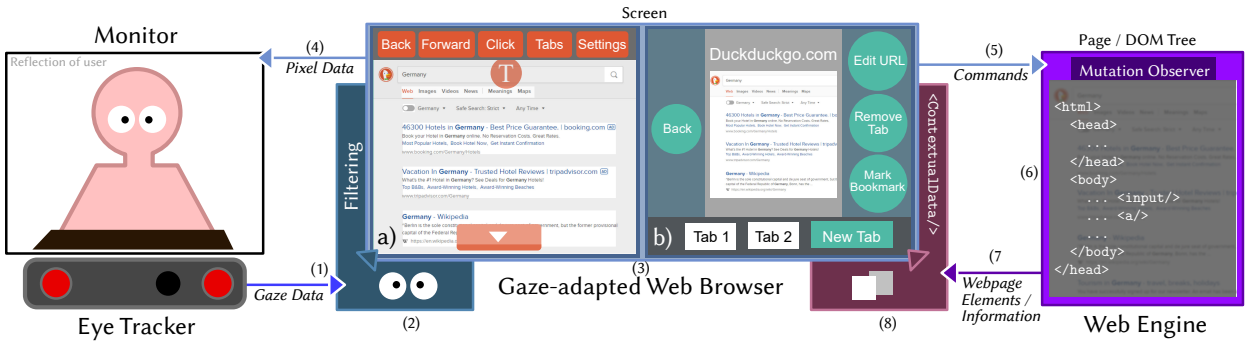


Figure 3: Extraction of webpage elements and rendering in custom framework enables direct utilization of gaze. This allows a) gaze-controlled overlays on webpages (here a incomplete mock-up with an search engine, featuring Web browser menu on top, button over text input search field, highlighting of links and scroll down button) and functional dwell time based buttons. Additionally, b) menu structures can be be designed to be directly steered by eye gaze.

This approach conforms with the principle of *Functional relevance*, as the most common browsing operations [32] are placed nearby the webpage a) and less frequently used features are accessible through a second level menu b). The user can switch between both screens with a trigger in the interface (in the figure it is "Tabs" in a) and "Tab 1" in b)).

Scrolling can be improved through the placement of the scrolling trigger above the page instead of the panel. As the Web engine provides feedback about the current scrolling status, the status can be displayed on the trigger for *Semantic Feedback* to inform the user about the effect of her action.

Manual activation of the virtual keyboard is not required, as the intention to input text is implicitly provided by the activation of the trigger upon the input field of the page. This *Contextual Awareness* allows for serving the user with appropriate facilities to achieve the desired task. This is possible through the parsing of the webpage structure and identification of elements for gaze-adaptation, like the text input field for the context of text entry.

To realize the proposed Web browser framework, we need to combine both a custom gaze-adapted interface and the Web engine component, which identifies the interactable elements. In the following section we describe how the identification of interactable elements on dynamic Web pages can be performed within the Web engine in real-time by utilizing features of the DOM standard. The section afterwards describes a gaze-adapted interface to complete the realization of a gaze-controlled Web browser framework.

2.3.1 Web Engine

A reliable and fast identification of interactable elements on Web pages is required to perform the proposed gaze adaption of Web browsing. The Section 2.3.1.1 describes the general Web page structure and we argue why initial parsing of the Web page is not sufficient for gaze adaptation. We explain in Section 2.3.1.2 how real-time identification and extraction of dynamic elements can be implemented in a modern Web engine that is following the current DOM standard.

2.3.1.1 Structure of Web Pages

A Web browser receives a requested Web page as an 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

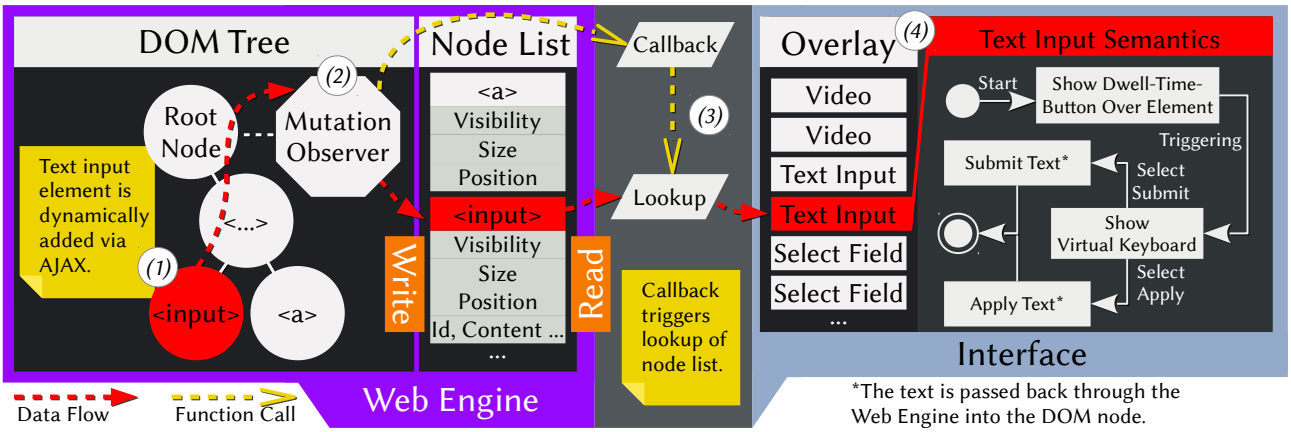


Figure 4: Data flow for real-time element identification. E.g., a text input field is added dynamically to the Web page via an AJAX execution. It is appended to the DOM tree *(1)* below an existing structure. The Mutation Observer on the root node *(2)* is notified about the change and stores the new node in a list of observed nodes. In addition, the interface is called to perform a lookup of the node list *(3)*. The newly added node is found and the interface can display an associated virtual button above the field *(4)*.

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, visibility).

Past approaches of gaze-controlled Web browsers by Abe *et al.* [3] and the WeyeB [60] prototype parse the HTML document after initial loading and identify hyperlinks for gaze adaptation. However, most modern Web pages are *dynamic* and their structure and layout change after initial loading. The dynamics are 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*,[4] 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 content via client-server communication at local or remote events. Hence, the availability, position and associated functionality of interactive Web page elements can change, affecting the directly gaze-controlled interface that is closely associated with these elements. Therefore observation of Web page structure needs to work at run-time to track changes in appearance and functionality, which would enable the effective realization of gaze-adapted interaction.

2.3.1.2 Real-time Observation of the Web Page

The DOM standard features the *Mutation Observer* [24], which is a mechanism to observe changes on the DOM tree and DOM node properties. It is available in the Web engines of

[4] `http://httparchive.org/interesting.php`

Table 1: Web page elements identified and extracted for the realization of gaze enhanced interaction design

Element Type	Description	Usage
(*E*1) Hyperlink	Elements of `<a>` tag.	See Section 2.3.2.1.
(*E*2) Fixed	Elements with style property `position: fixed`.	See Section 2.3.2.2.
(*E*3) Overflow	Elements with style property `overflow: scroll` or `auto`.	See Section 2.3.2.2.
(*E*4) Text Input	Elements of `<input>` tag.	See Section 2.3.2.3.

modern browsers[5] and allows for a system agnostic and efficient way to track changes within dynamic Web pages. One can inject a JavaScript snippet including a Mutation Observer instance into the `document.body` node of the DOM tree every time a new page is loaded. The Mutation Observer receives records about creation, update or removal within the DOM tree. An example is is depicted within the Web engine component in Figure 4, where a Mutation Observer observes a newly attached text input field and stores information like visibility, size, position, id or textual content in a separate node list. Every observation also triggers a callback towards the interface including an identifier of the created, updated or removed DOM node. The interface accesses the node within the Web engine using the identifier in the callback argument and transfers the data into the memory space of the interface. The retrieved data then can be used for adaptation, e.g., providing a virtual button as overlay.

However, the current standard of the Mutation Observer does have some limitations, since it is not able to track changes of computed styles. These are style properties of a node that are not defined by the node itself but adopted from either a style class in the style sheet or a property defined by parent nodes. E.g., when an unobserved container changes its visibility property this might also affect observed child nodes like a text input field, and the Mutation Observer is not notified about its visibility change. But observing the complete DOM tree is slow on complex Web pages and replicating the behavior of the styling mechanism is conceptually a hard task. Therefore, we propose to frequently poll the properties of identified elements during run-time and a check for changes. This polling can be limited to already identified and adaptation-relevant elements, as creation and removal of nodes is fully observable by the Mutation Observer itself.

The described identification method can be implemented on Web engines that follow the DOM standard, however the gaze-adaption itself requires flexible rendering mechanisms and integration of the eye tracking environment. In the next section we present our realization of the interface that adapts the Web page interaction for direct gaze-control using information from the Web engine (e.g., page height, scrolling status) and identified and extracted Web page elements. Refer to Table 1 for details about the specific Web page elements $E1 - E4$, which are required for the realization of GazeTheWeb design, as described in the next section.

2.3.2 GazeTheWeb

GazeTheWeb is a browser which implements the gaze interaction principle for Web access, i.e., it integrates the control functionality of webpages in an eye tracking environment and provides full-featured Web browsing with eye-based input. GazeTheWeb incorporates the unsupervised extraction of webpage elements with gaze-interaction principles, while relying on Chromium Embedded Framework (CEF) as browser backend. The Chromium based framework provides more utility and control to include eye gaze events in the browser, rather than building browser extensions. E.g., we can receive webpages as pixels that give a complete power over webpage rendering, which is essential for dynamic interface adaptations for gaze interactions. Furthermore, CEF makes use of a multi-process and multi-thread approach to deliver a stable and fast user experience.

GazeTheWeb prototype was first demonstrated at the 14th International Web for All conference 2017 (W4A'17) [51], where it won the TPG challenge award for its usability and impact in the field of Web accessibility[6]. Furthermore, the technical framework received WWW 2017 honorable mention award for the approach of unsupervised extraction of Web elements [34] .

The interface in Figure 5 represents the three-column design of GazeTheWeb browser application, satisfying the principle of *Element Positioning* by bringing the core functionalities to the primary interface. On the left side, the *Web panel* covers common actions like going back and forward or opening the tab overview of the browser. It also consists of a globally visible

[5]Check compatibility at the bottom: `https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver`

[6]`http://www.w4a.info/2016/2017`

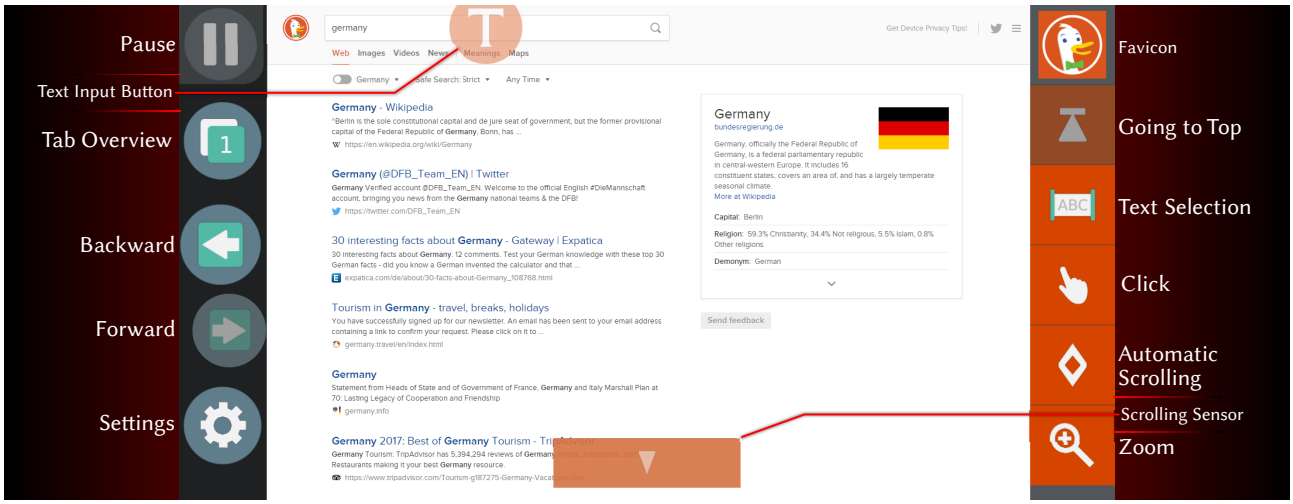


Figure 5: Front interface of the Web browser, showing the search results page of the DuckDuckGo search engine. Web panel on the left provides global functions. The Tab panel on the right exposes functions that can be executed on the current page. In the middle is the Web view, which combines the webpage rendering with gaze-controllable overlays.

pause button that can be activated to forestall all eye-based interactions. This feature has been added to enable a non-distracted view on the interface during a presentation or self-exploration by the user.

The *Tab panel* on the right side of the interface exposes the actions that can be performed on the current webpage, e.g., text selection, click emulation, or to enable automatic scrolling. For a better visual integration of Tab panel and page overlay, a prominent color is identified from the favicon provided by the webpage, and is used as color for the interface elements. In example of Figure 5, the most prominent color of the favicon has been calculated as red, so all buttons in the Tab panel are colored in red. This improves the visual integration of the gaze-controlled application buttons with the active webpage and symbolizes a functional context, as the colored buttons trigger actions that are performed on this webpage.

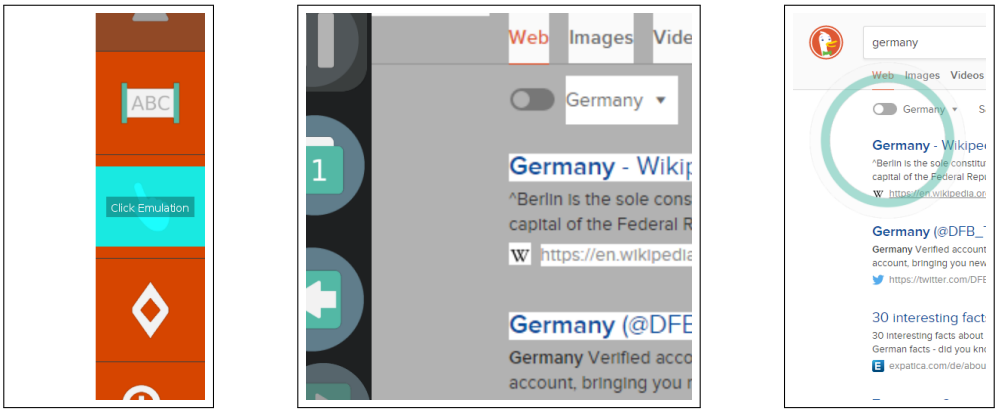
The center column of the interface contains the actual webpage rendered by the underlying Chromium browser, called *Web view*. The webpage itself does not react to gaze input (besides the later introduced automatic scrolling approach), but instead, interactive overlays are added with respect to selectable objects and input fields within the page. For example, there is a T-letter within a circle placed on the text input field of search, which can be activated to submit a query on the *DuckDuckGo*[7] search page. We describe these essential browsing functionalities in more detail in the following subsections, while providing details about the incorporation of webpage semantics for an advanced gaze-controlled interface.

2.3.2.1 Navigation

Navigation to content within the page or other pages via hyperlinks is an essential component of Web browsing behavior. Whereas simple back and forward navigation is provided as integrated in the Web panel, in-page navigation is a non-trivial task. For such navigational task, basic dwell time selection strategy on screen does not work, because in Web surfing scenarios links are visually rather small. The clickable elements may be also spatially dense on a webpage, and the eye tracking accuracy is not sufficient to translate a gaze position directly to a click coordinate. This makes it impossible to precisely choose intended links with simple gaze fixations and dwell time selection.

There are alternative methods proposed in the literature such as color coding [44] or dedicated buttons [58]. These methods may work accurately for a controlled environment but are not scalable to the Web, as it is not trivial to extract all navigational elements like textual

[7] `https://duckduckgo.com`



(a) Initiation (b) Continuous zooming (c) Selection

Figure 6: Navigation over hyperlinks. This figure depicts the three states of (a) initiation of the procedure, (b) the continuous zooming while the user tracks the link to select and (c) the performed selection, visualized by an collapsing circle.

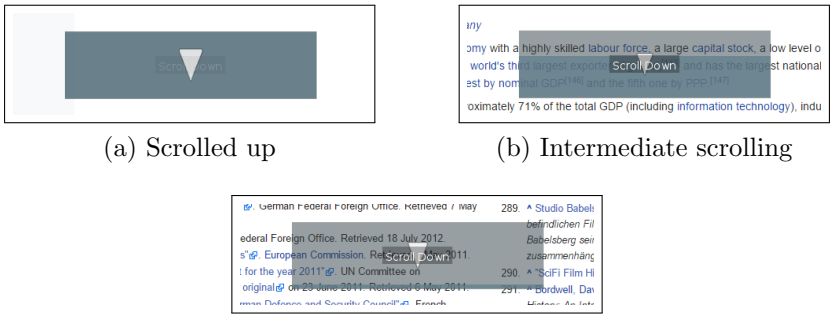
hyperlinks or buttons of a webpage. Especially through the technologies like *Asynchronous JavaScript and XML* the content of the webpage can change arbitrary times during a session. There may be even custom JavaScript-driven interfaces, which are impossible to extract in an unanimous manner.

Therefore, we propose a click event supported by but not depending on extracted hyperlink elements trough *Contextual Support*. Pure emulation tools employ a two-step clicking process to tackle accuracy issues [39]. The first click opens a window with zoomed view, where a second dwell time is executed before a click is performed. Instead, we utilize a continuous zooming behavior demanding on smooth pursuit by the user. A click is initiated with the activation of the finger pointing button in the Tab panel, see left image in Figure 6. Then, all elements representing a hyperlink on the webpage are highlighted, in order to support the user in the decision process what to click on, providing *Dynamic Feedback*. Zooming process starts with any gaze coordinate upon the Web view, as displayed in the middle image of Figure 6. This zooming is centered at the smoothed gaze position on the webpage. The zoom level is increased when the user focuses on a certain page coordinate and decreased when the gaze starts to wander. Additionally, the content of Web view is adaptively moved so the fixated coordinate is closer to the center of screen. This improves accuracy of the selection process, as the calibration of the outer screen areas is more prone to be inaccurate [21]. When the zoom level is over a certain threshold, a click is performed at the focused page coordinate. If there is no detected hyperlink element at the determined gaze coordinate but there exists one in the nearby region, the coordinate is moved to the center of the nearest link element in order to perform the click. Moreover, a visual feedback is provided by a shrinking circle around the clicking point, as visible in the right image of Figure 6.

2.3.2.2 Scrolling

Scrolling by gaze has been a major topic in gaze-based interaction research. Various approaches have been published, like a two-step mechanism where a user first selects to scroll and then is presented a scale to control speed [60], or the automatic scrolling with respect to gaze coordinates position on screen [38, 54].

In GazeTheWeb we integrate the two relevant methods by providing the user a choice between the manual approach with visual sensors that instantly react to focus, and an automatic one, where the content beneath the gaze position is constantly centered. Both approaches provide *Dynamic Feedback* by cueing the current scrolling status to the user at the same coordinate as interaction takes place. Furthermore, they implement *Contextual Support* through the awareness of the page being able to be scrolled or not.



(a) Scrolled up (b) Intermediate scrolling

(c) Scrolled down

Figure 7: The scrolling sensor provides feedback about the scrolling of the page. First, the page is scrolled up and the sensor if filled (a). At fixation by the user, the scrolling starts and the progress is displayed by the vertical filling status of the sensor (b). When the bottom of the page is reached, the sensor is transparent (c) and fades out.

The visual sensors are displayed at the top and bottom of the screen, when the page height exceeds the available screen space and scrolling is possible. In Figure 5, the page reached its topmost scrolling position, thus, the upper scrolling sensor is hidden but the lower one is visible. These sensors are specifically designed elements whose activation is not based on dwell time but they immediately react to gaze focus. The longer a user focuses on the sensor, the higher is its penetration and scrolling fastens up. Whenever the focus is lost, it recovers to a non-penetrated state and scrolling stops. Instead of a dwell time based approach, this behavior has been chosen since it provides an instantaneous and more natural interaction for the challenge of scrolling. For an in-place feedback about the current scrolling status, a vertical progress bar covers the scrolling sensors, as depicted in Figure 7. It represents the different visual states of scrolling by determining absolute scroll position and the height of the webpage.

Automatic scrolling can be activated optionally in the Tab panel as alternative for the manual scrolling approach. When activated, the Web view content beneath the gaze is centered within the screen by scrolling the page appropriately. Furthermore, when the user is inspecting a detected fixed – or so-called viewport relative – region of the webpage (like a navigation bar or a static pop-up), any automatic scrolling is not intended and therefore not performed. The automatic approach is also the default behavior for scrolling within the page (e.g., Facebook chat window). We detect webpage elements that overflow, and imply automatic scrolling, since the space is too limited for additional manual scrolling sensor fields.

2.3.2.3 Text Input

Text input is an essential feature of Web interaction for various operations like entering a search query, an E-Mail address or chat messages. Through *Contextual Support*, the workflow for text input can be accelerated through the knowledge where the text input elements on webpages are, which content they already hold and how to submit them. We detect text input fields and overlay them with a dwell time based button. An example is depicted for the search field on top of DuckDuckGo webpage in Figure 5. When the user activates such an overlay button with dwell time input, the text input facility appears on the screen.

Conventional eye typing keyboards either cover the complete screen space in order to cope with given accuracy [18] or implement a multi-step selection process [26]. Figure 8 depicts the text input facility in our browser, including the virtual keyboard for entering characters, a display and button based text editing. A zooming effect has been incorporated in the keyboard to improve the accuracy of the key selection process while coping with the limited screen space shared with other functionality. The focused key increases its size and keys within a certain radius move away. This enables the user to easily select a neighboring key, when the currently focused one is not the objective of interaction. A letter is typed in using dwell time, analogously to other interactive elements of the interface.

Figure 8: Text input via dwell time based QWERTY keyboard, that employs zooming effect for more robust key selection. In the screenshot, the key with letter 'h' is fixated by the user. While this key is magnified, the surrounding keys are further spread to make it easier either to choose a key different from the current one or to stay at it. The growing cyan overlay indicates the remaining dwell time of fixation, as the letter is typed in when the overlay fills the complete key.

Between the text area and the key layout a horizontal stack holds other interface elements, like backspace, paste clipboard content and word suggestions. Furthermore, it provides the option to apply the text in an input field, or to directly submit the text. This is especially useful for single input fields like search queries or chat messages, as an additional click on "search" or "send" on the webpage can be avoided. Text editing is possible via moving the blinking cursor using the arrow buttons at the sides of the display area on the top of the screen. Any triggered insertion and deletion is performed at the current cursor position within the text.

2.3.2.4 Tab Management

The Web grants access to various activities, and hence tabbed browsing is an integral feature of modern browsers for fast and easy task switching. We applied the same features in our browser with gaze-controlled interface elements to deliver a wholesome surfing experience, while respecting the principle of an unclustered *Menu Structure*. A user can enter the tab overview by selecting the respective button in the Web panel (see left side of Figure 5).

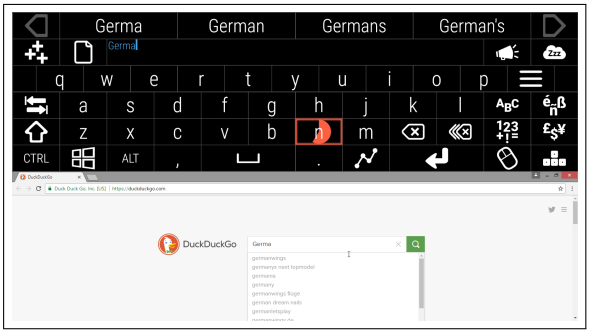
Figure 9 depicts the tab overview screen. The interface offers interactions with the current foreground tab by editing the URL, marking the open page as bookmark, reloading the page or removing the foreground tab. The screen for editing the URL also contains a button to show the list of available bookmarks, from which can be chosen instead of a manually typed address. The bottom panel shows the open tabs, where the user can either select an existing or open new tabs.

2.4 Performance and Feasibility

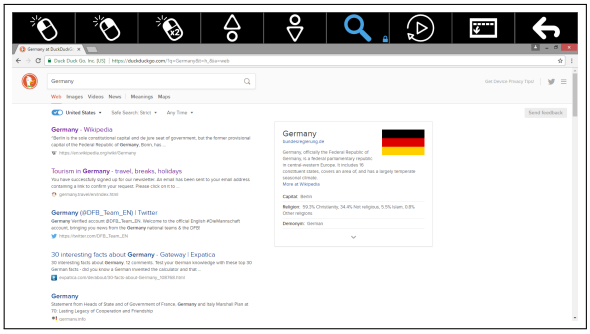
2.4.1 Lab Experiments

To quantify the performance and usability of GazeTheWeb, we propose a comparative evaluation with the traditional approach of mouse and keyboard emulation by eyes, i.e., OptiKey [1] was selected as a benchmark system (details are provided in the following subsection). In the experimental setup OptiKey was used to control the popular Google Chrome browser. The experimental study took place at Koblenz University campus, and was conducted in two different sessions. In the first session, we used research-grade eye tracker to compare both software setups under similar conditions. The second session was carried out with additional participants

Figure 9: Management of tabs via a gaze-controlled interface. The current foreground tab is previewed in the center. For this, different actions are exposed in the vertical panel next by. On the bottom, open tabs can be brought to foreground by fixation exceeding a dwell time or additional ones can be created in similar fashion.



(a) Text input via dwell time buttons in virtual keyboard. Inputted text is passed as emulated keyboard events to the active application.



(b) Mouse mode features various actions like left click (first button from left) or scrolling up and down (forth and fifth button from left).

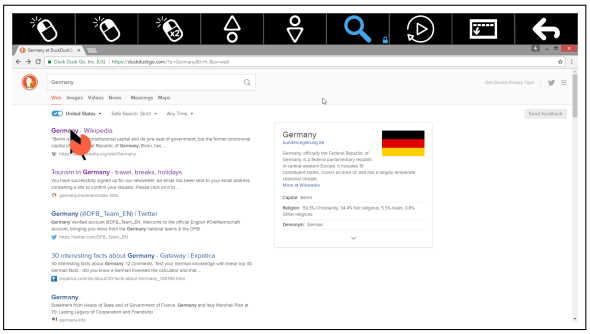
Figure 10: Two modes of OptiKey are used in the experiments. First, the keyboard mode (active after startup of OptiKey) enables text input on a virtual keyboard. Second, the mouse mode that features emulation of common mouse actions.

to reinforce the findings; furthermore a consumer-grade eye tracker was used to substantiate the systems viability with affordable low-cost hardware. All the participants were able-bodied students from the university and had experience in interaction with the Web using a traditional interfaces like a desktop browser.

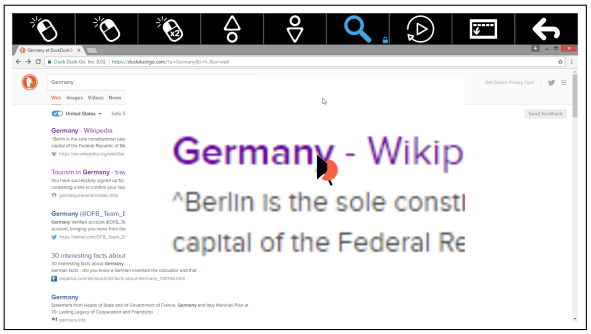
In order to conduct such comparative study we need to identify the most essential browsing activities. In this regard, the related work [17, 60] provides indication which features of a Web browser in context of accessibility are prominent and should be included for a controlled user study: Navigation, scrolling, text input and use of bookmarks. This inherently includes important Web browser actions as described by Kellar *et al.* [32].

2.4.1.1 Comparative Benchmark – OptiKey

There are multiple commercial systems available that support mouse and keyboard emulation for gaze-based interaction. Among these systems OptiKey is a popular open source application compatible with modern eye tracking devices. OptiKey utilizes similar interactions mechanics as the commercial ERICA system [39], like button activation through dwell time and simple magnification for enhancing accuracy of eye-based pointing. Similar to GazeTheWeb interface, a fixation on buttons reveal a minimal textual description of their functionality. The release of



(a) First dwell time to initiate click position.



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

Figure 11: Clicking is performed in two steps, after initiation via the left click button on the left of the panel. First, the user fixated 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 inclusive dwell time is necessary to choose the final point of click. Dwell time is visualized by the filling pie next to the the big black arrow that indicates the detected fixation position.

OptiKey had high impact on online media like *businessinsider.com*[8] and *alsnewstoday.com.*[9] It has also been used at MIT[10] for supporting people with motor disabilities in general computer usage. For the experimental evaluations, we employed OptiKey in combination with the Google Chrome browser (version 55 for the first session and version 58 for second), as application controlled with the emulated input devices. Google Chrome incorporates the same technology that underlies our GazeTheWeb framework, hence the actual webpage content is rendered similar for the end-users.

The OptiKey interface consists of different modes of emulation, i.e., tangible mouse and various keyboards (which emulate the physical keyboard). For the experimental setup, only the standard QWERTY keyboard (Figure 10a) and simple mouse (Figure 10b) modes were used. A user can switch between both modes with a dwell time based button in the interface. In the following, we describe the interaction procedures required to fulfill the browsing tasks with OptiKey, which are executed in the experiment.

Navigation

Navigation within a Web browser is mostly performed via the back button or by hyperlinks. A user can operate these functions of the Google Chrome browser via mouse emulation of the mouse mode in OptiKey. In the mouse mode, the left mouse button emulation button needs to be activated to initiate the pointing process. Then, a mouse pointer is displayed at the detected focus point within the screen. After a certain dwell time of the fixation, the content beneath the gaze is magnified and presented in the center of the screen as pop up overlay, as shown in Figure 11b. After another dwell time on this pop up, a click is performed at the magnified position. The extra zooming 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 11 for a screenshot of the magnified pointing mechanism in the middle of a fixation process.

Scrolling

Usually, scrolling on a desktop computer is performed with the scrolling wheel on the physical mouse. OptiKey adopts this paradigm and places options for scrolling emulation in its mouse mode. To perform an upwards scrolling emulation, the button for scrolling up has to be

[8] http://www.businessinsider.com/an-eye-tracking-interface-helps-als-patients-use-computers-2015-9

[9] https://alsnewstoday.com/2016/03/08/article-for-als

[10] https://vimeo.com/148316508

activated. Respectively, the button for scrolling down in order to scroll down. Then, the user has to fixate the position on the screen 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 button is available to repeat the last executed action. So, two dwell activations for the first scrolling event and a single dwell activation for consecutive scrolls are necessary.

Text Input

In the keyboard mode, buttons representing keys on the virtual keyboard are displayed in the interface. These buttons are activated by fixations that exceed a certain dwell time, similar to the other buttons featured in the OptiKey interface. The QWERTY layout has been chosen for the experimental setup, as depicted in Figure 10a and also used in the GazeTheWeb keyboard as presented in Section 2.3.2.3.

Bookmarking

To add the current page as bookmark in Google Chrome, the user has to perform a click on the star right to the URL bar. We also asked the participants to finish the task by a second click on the "Done" button within the bookmarking pop up. This results in two mouse clicks for bookmarking. 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 are necessary to access a saved bookmark, two for menu navigation and the third for selection. For a normalized experimental setup, we deactivated the optional bookmark bar beneath the URL bar of Google Chrome.

2.4.1.2 Methodology

First, we executed a study with healthy-bodied participants with a research-grade eye tracker, who performed the defined tasks with both software setups. Second, the same software setup but with a low-cost eye tracker has been executed with a different set of participants. The independent variable in this experiment was the utilized software combination (GazeTheWeb or OptiKey and Google Chrome). We controlled lab ambiance and eye tracker distance for each run, to provide comparable environments. Use of corrective lenses was noted. Depending on the software setup, we analyze the required time to succeed in a task and the subjective feedback from SUS, NASA-TLX and a custom heuristics questionnaire. In the following, the software setups with *GazeTheWeb* (GTW) and *OptiKey plus Google Chrome* (OK) are referred by their short naming within the parentheses.

Session A

In the first session in December 2016, 10 (4 female and 6 male) paid students in the age range from 24 to 31 years (average = 25.4, sd = 2.12) participated. They had no prior experience with operating GazeTheWeb and OptiKey, but 7 out of 10 had already participated in an previous eye typing evaluation with a different application. 4 participants wore corrective lenses (2 female and 2 male). All of them were familiar with the QWERTY keyboard layout used by the keyboards of both softwares. The participants were paid a small amount for their effort after the session. A SMI RED-n remote eye tracking device was used as input. This device is a research-grade eye tracker and samples the eye-gaze with a constant frequency of 60Hz.

Session B

The second experimental session took place in April 2017. It used the same software setup as the first session, but a different eye tracking environment and a new set of participants. It was

conducted with 10 paid students from university campus (5 female and 5 male), aged 23 to 31 years (average = 25.7, sd = 2.31). 5 out of 10 participated had some prior eye tracking related experience. 3 participants wore corrective lenses (1 female and 2 male). Again, all were familiar with the QWERTY layout used by both softwares. For the eye tracking environment, Tobii EyeX device was used, which represents the first generation of consumer-grade eye tracking devices. There is no exact definition of its sampling rate, however a minimum of 30Hz up to 70Hz is to be expected.[11]

Apparatus

For both sessions, the dwell time of both systems was set to one second, as appropriate for untrained users and chosen in related evaluations [60]. The remote eye tracking devices were attached to the bottom of 24inch monitor, with resolution had been set to 1600x900 pixels. Artificial illumination and blocking of sunlight provided a similar lighting condition for all participants. Additional to the calibration before training and task execution, recalibration has been performed if a participant reported about a input bias.[12]

2.4.1.3 Procedure

Evaluations known to the authors examining gaze-controlled Web browsers have been purely testing feasibility [39, 60]. We propose to compare two approaches of gaze interpretation in terms of both performance and usability in the context of a Web browser. For the experimental process, counter-balancing for the order of the software was used so as to minimize the bias of one system over the other.

Training

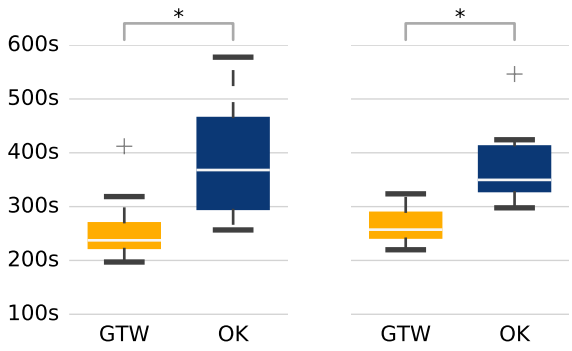
Prior to each execution, an explanation about the general nature of the tasks was given. Each of the participants were instructed about their designated tasks for the experiment. After the oral explanation session, they were provided with a training phase that helped them to understand the functionality of the software and the functionality of the required interaction elements, including an initial eye tracker calibration. Upon completion of the training, they were given some time to get accustomed to the environment on their own, so that they have a certain confidence when the actual experimental session started. After this, they were given a break of a few minutes followed by the start of the experimental session, including a second calibration of the eye tracking device. This was done twice per experiment slot, for each software setup.

Tasks

The tasks involved each participant to enter a search string on a particular search engine and then access a hyperlink from the search results. Once they loaded the linked page, scrolling and link navigation features were tested on reaching three different pages from the found page. For each subpage, they were asked to add it to the list of bookmarks. Each participant was instructed to enter the search string "Germany" on the DuckDuckGo landing page. They were asked to click on the English Wikipedia link of Germany from the search results. Upon reaching the Germany page, they had to scroll to the constituent states from where their first work was to select North Rhine Westphalia (NRW). Upon reaching the page, they had to scroll down to the cities of NRW and select Bonn from there. Once on the Bonn page, they had to bookmark it and after that use the back navigation twice to get again to the constituent states section. The same process was executed for the constituent State of *Baden-Württemberg* and

[11] `https://help.tobii.com/hc/en-us/articles/212818309-Specifications-for-EyeX`

[12] Tobii Technology, AB, Drift Effects, in User Manual: Tobii Eye Tracker and ClearView Analysis Software. Tobii Technology AB. p. 15, 2006.



(a) Session A (b) Session B

Figure 12: Box plot of the times by the participants for the execution of all tasks. There is a significant difference between the times for both systems in both session.

the city *Mannheim*, respectively *Lower Saxony* and *Oldenburg Land*. After the final bookmark of Oldenburg Land, they were instructed to access *Bonn* from their list of bookmarks.

To sum up the task, each participant 1) navigated across links, 2) scrolled pages, 3) entered a search string via the browser, 4) created bookmarks and 5) accessed a bookmark from the list of bookmarks. These tasks were designed in accordance to the most common tasks involving the usage of a Web browser [32], as discussed at the start of this section.

Survey

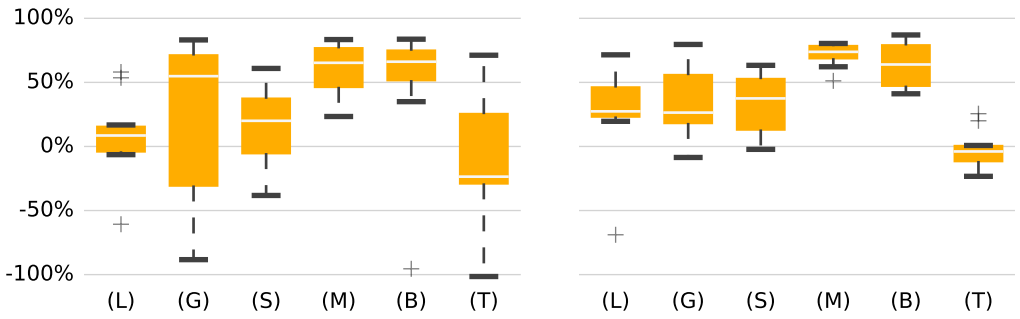
After the participants finished the tasks using one environment, they were handed a SUS and a NASA-TLX questionnaire. The *System Usability Scale* (SUS) [2] questionnaire is used to measure the overall usability of applications. The SUS contains 10 questions, which are answered on a five point likert scale from *strongly disagree* to *strongly agree*. The *NASA Task Load Index* (NASA-TLX) [25] contains six components: mental, physical, temporal demands, success and performance, effort load, and stress and frustration. For each component, the participant can decide the most applicable scores on a scale from 1 (low) to 100 (high) in 5-point steps. In addition to SUS, a custom heuristic evaluation for eye tracking was created, to analyze whether the interaction elements have a good visibility and were intuitive to be used for interaction.

2.4.1.4 Results

First, we present the objective measures of our evaluation. Then we give an overview of the subjective feedback by the participants.

Objective Results

All people succeeded in the the given tasks with both systems. Although different hardware has been used in the two sessions and there was a gap of 5 month of time between them, the general tendencies are similar for both sessions. The participants were in average over 100 seconds faster with *GTW* than with *OK* to complete the procedure. Furthermore, the time differences between both systems are normally distributed. This allows us to assess the significance of the differences by a paired t-test with calculating the two-tailed p-value. For Session A, there is a significant difference in the completion time for *GTW* (M=260.2s, SD=63.1s) and *OK* (M=418.2s, SD=172.7s), with t(9)=-3.57, p=0.006. Analogously for Session B, we can report a significant difference in the completion time for *GTW* (M=263.6s, SD=33.7s) and *OK* (M=376.7s, SD=73.1s), with t(9)=-4.09, p=0.003. See Figure 12 for a box plot showing the consistent times the participant required to fulfill all tasks in *GTW* whereas the times for



(a) Session A (b) Session B

Figure 13: Box plot showing the time improvement of the participants for various browsing tasks when using *GTW* over *OK* in percentage. The improvement has been calculated as $(Time_{OK} - Time_{GTW}) \div (Time_{OK})$. (L) search result selection via hyperlink navigation, (G) going back to previous page, (S) scrolling on Wikipedia, (M) marking a Wikipedia entry as bookmark, (B) selecting a bookmark, (T) input of the initial search query via the virtual keyboard.

OK showed much more variety. The experiments let us conclude that the participants where significantly faster with the *GTW* system in performing the provided task than in *OK*.

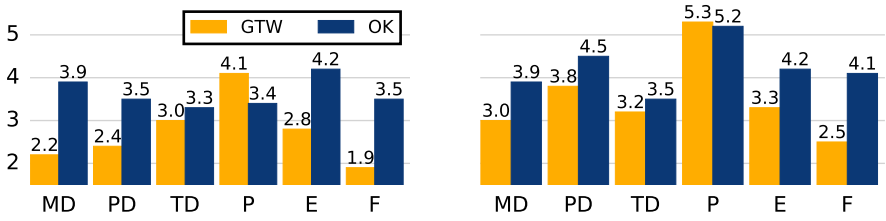
In addition, we plot in Figure 13 the relative time improvements of participants in diverse browsing activities when using the *GTW* system in comparison to *OK*. Speed of navigation through selection of the search results (L) and going back (G) were improved for almost all participants Especially in going back, half of the participants were at least 25% faster using *GTW*. The timings for scrolling (S) could be improved similarly. Especially for multi-step actions like adding (M) and selecting bookmarks (B), the direct gaze-controlled interface of GazeTheWeb shows a high improvement over the combination of OptiKey and Google Chrome. About 75% of the participants improved their time by 50% in *GTW* compared to *OK*. Due to the similar design and functionality of the virtual keyboard in both system, participants performed in text input (T) very similar with no perceptible trend. Because the text input time differences are normally distributed, we can strengthen this claim by a paired t-test to assess the two-tailed p-value. There is no significant difference in the text input times by the participants of Session A for *GTW* (M=17.2s, SD=5.88s) and *GTW* (M=18.43s, SD=10.45s), with t(9)=-0.334, p=0.746. A similar outcome can be computed for Session B with the text input times for *GTW* (M=17.73s, SD=2.6s) and *OK* (M=17.7s, SD=3.56s), with t(9)=0.0379, p=0.97. For both sessions, the p value is close to or nearly equal one, which denotes the resemblance of text input in both systems.

Subjective Results

The subjective evaluation of the experimental session was achieved with the SUS for general usability assessment, NASA-TLX to measure the workload and a custom heuristics criteria to evaluate the Web browser design in the two sessions.

Table 2a shows very high average SUS usability scores of 82.5 and 71.75 for *GTW* in comparison to *OK*, indicating the high acceptability rate of the system among the participants. The difference between the SUS scores by the participants are normally distributed 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 in Session A for GTW (M=82.5, SD=13.64) and OK (M=58.5, SD=19.05), t(9)=3.82, p=0.0041. A similar trend is visible for the SUS scores of Session B, with GTW (M=71.75, SD=17.20) and OK (M=51.5, SD=20.04), t(9)=1.86, p=0.096.

The consistently better ratings of *GTW* over *OK* in terms of NASA-TLX mental workload (MD), level of effort (E) and sense of stress and irritation (F) is presented in Figure 14. The



(a) Session A (b) Session B

Figure 14: Raw NASA-TLX scores which assess the workload of the participants. MD = Mental Demand, PD = Physical Demand, TD = Temporal Demand, P = Performance, E = Effort, F = Frustration.

Table 2: Average subjective feedback by the participants about the usability of the systems.

(a) System usability scores (SUS)

Session 1		Session 2	
GTW	OK	GTW	OK
82.5	58.5	**71.75**	51.5

(b) Heuristics scores

Heuristics Metrics	Session 1		Session 2	
	GTW	OK	GTW	OK
#1 Visibility	**8.5**	7.9	**8.4**	7.4
#2 Size	7.3	7.4	**8.6**	6.5
#3 Reading	**8.4**	5.3	**7.6**	6.8
#4 Link Navigation	**7.3**	5.3	**7.2**	5.4
#5 Error Recovering	**8.4**	6.3	**7.5**	6.8
#6 Conventional	**8.0**	6.9	**7.8**	5.9

feeling of success (P) in accomplishment of the task was also slightly higher for *GTW* in comparison to *OK*.

To asses the aspects of the gaze-controlled interfaces [65], we set up a eye tracking interface heuristics questionnaire asking: **#1** How was the *visibility* of the main interaction elements? **#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 results of this questionnaire is shown in Table 2b, where we can see that features like the ease of recovering from errors (#5) made on *GTW* is way higher in comparison to *OK*. The intuitive factor (#3) and ease of hyperlink navigation (#4) is also higher for *GTW*.

2.4.1.5 Discussion

The objective of the study was to assess the difference in performance, usability and workload of GazeTheWeb and a state-of-the-art emulation approach, where gaze-based input is emulating mouse and keyboard events. Objective results from this study with able-bodied users indicate a better user performance for GazeTheWeb. The task completion is significantly lower for our browser than the emulation-based approach. Since the experimental tasks are orientated at everyday tasks consisting of essential browsing activities like navigation, scrolling, text input and bookmarking, the results indicate a high potential for a improvement of performance for users compared to the state-of-the-art systems. Especially interaction via the gaze-adapted interfaces like bookmarking or back navigation requires lower amount of time than the emulated control of classic interfaces. The menu structures of traditional interfaces with small buttons bear a challenge for gaze-based input emulation, while the gaze-adapted interface allows a straight forward interaction. It helps to overcome the limitations of eye-gaze input and provides direct visual feedback, as discussed in the eye-gaze adaptation optimizations of Web browsing.

The average text input times are similar for both systems (the p-values are close to 100%). This was expected as both the keyboards utilizes same mechanism, and the dwell time was equalized. This also verifies that experimental conditions with both systems were identical, and the variation in task completion time indeed justifies the faster Web interaction.

The subjective SUS feedback indicates superiority of GazeTheWeb compared to the emulation approach, as the rating by the participants positions it far above the general median of software. The average SUS score of 82.5 for *GTW*, when the research grade eye tracker was used, is considered to be in the highest order as per SUS guidelines.[13] 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 scores of 58.5 and 51.5 are 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 NASA-TLX, all major points of workload are rated lower for GazeTheWeb in both sessions. The participants felt especially less mental demand, less effort and less frustration, but a higher sense of success. An interpretation for this positive results in subjective feedback might be that the extra command-translation layer of the emulation 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 study comparing the interaction with Twitter using a gaze-adapted interface and an emulation approach [35]. 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 then for the combination of OptiKey and Google Chrome.

2.4.2 Pilot Phase I trials analysis

It is imperative to investigate the feasibility of gaze-controlled browsing to accomplish daily Web-based activities by the people who would benefit from this accessible technology. Hence in this section we discuss the analysis of the study was conducted as part of MAMEM project first phase trials [53]. In these phase I trials, participants completed a series of training tasks to allow them to learn how to use the MAMEM system, after which participants were asked to perform dictated every-day tasks: Reading and replying an E-Mail (E-Mail Task), editing a photo (Photo Task), using social media (Social Media Task) and watch a video (Video Task).

2.4.2.1 Methodology

The trials were executed at three clinical sites, each responsible for one cohort (As described in detail in MAMEM deliverable D6.4). 6 participants (1 female, 5 male) with Parkinson disease (average age 64, sd 6.69), 6 participants (2 female, 4 male) suffering from neuro-muscular diseases (average age 34.2, sd 6.18), and another 6 participants (1 female, 5 male) with spinal-cord injury participated in the trials (average age 45, sd 16.4). Additionally, 18 healthy subjects (5 female, 13 male; average age 45, sd 13) attended the experiments with the same setup (6 at each site), acting as benchmark. All the participants had no prior experience with eye tracking.

Before the experiment of dictated task was executed, all participants underwent an interactive gaze-based training game, which introduced the fundamental interaction of GazeTheWeb, necessary to fulfill the dictated tasks (more details in MAMEM deliverable D5.4). Dwell time of the system was set to one second, same as for the experiment at Koblenz university.

[13] `http://www.measuringu.com/sus.php`

2.4.2.2 Results

Objective Results

The raw results of the task time completion time are shown in Table 3. The results are analyzed in terms of difference between the target group and healthy participants. An independent two-tailed t-test with the two sample sets reveals no significant difference between both group, as all computed p-values are by far greater than 5%.

There are some participants, for whom the single or complete tasks failed, which is visible through the observation count that does not sum up to the whole 36 observations. One participant with Parkinson faced Claustrophobia and could not proceed the tasks, therefore was excluded from the experiment. For two spinal-cord injured participants, the eye tracker did not work. They were excluded as well. For one healthy participant, there was an issue with the Internet connection during the photo editing task, and therefore this task could not be executed for this participant.

Table 3: Task completion time 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	300.9	152.93	246.6	148.6
	Time Deviation	239.12	72.11	98.15	62.1
Control Group	Observation Count	18	17	18	18
	Time Mean	255.17	144.06	253.22	157.44
	Time Deviation	147.39	76.34	99.44	68.56
t-test	Degrees Of Freedom	22	30	31	31
	t-Value	0.65	0.34	0.19	0.39
	p-value (two-tailed)	*0.53*	*0.74*	*0.85*	*0.7*

Subjective Results

Measurement of usability extracted from the SUS survey also shows no significant difference between the two groups. Average SUS score of target group is 77.36, while average score of control group lies at 72.17. This is above average of 68 [2], indicating an above average usability for GazeTheWeb. There is no significant difference between both score sets, since it is $t(30) = 1.33$, $p = 0.2$ for an independent two-tailed t-test.

2.4.2.3 Discussion

The presented study with the target group participants demonstrates the feasibility regarding every-day tasks like writing an email, editing a photo, participating in a social network or watching a video. The system allowed almost all participants to perform the four real world tasks, no barriers introduced by our system were reported. There is clearly no significant difference between target and control group, as all p-values are far above 5%. This demonstrates the feasibility of GazeTheWeb for the target group of people facing motor disabilities. This also indicates a validity to the results achieved in the performance study at university campus, i.e., GazeTheWeb would support people lacking fine motor control to interact with the Web faster in comparison to current approaches. Furthermore, the SUS scores are harmonious with the prior studies, confirming a general acceptability of the system.

3 EEG-based interaction and analysis

During the past decades, Brain-Computer Interfaces (BCIs) have been widely utilized for providing alternative communication options to both handicapped and able-bodied people through several applications, including relevant information recommender systems [20], spellers [10, 15], robotic limbs [50] and wheelchair controls [47]. Advances in machine learning methods in conjunction with a wide circulation of consumer HCI products (e.g. low cost EEG capturing devices, eye tracking systems, etc.) allowed BCIs to evolve into multimodal systems offering more capable, adaptable, versatile and natural interfaces.

While BCIs have managed to achieve significant improvement in terms of detecting the users intentions over the last years, in a real world setting, interpreting brain commands is still an error-prone procedure forcing the users in unintentional interaction errors. Although multimodal interaction offers more reliable solutions, multidimensional control and target detection performance continue to constitute the major barriers for further deployment.

MAMEM's multimodal system builds on top of the mature eye-tracking technology as the main interaction methodology and combines information from the EEG sensor with the focus on alleviating these two shortcomings. First, for the multidimensional control, SMR offer the option of user interaction without a visual stimulus, which is important since the users' visual system is engaged for the gaze-based interaction. In this direction, our main objective is to transform the typical cue-based imaginary movement detection (i.e. a visual/audio cue triggers the users to make an imaginary movement) to self-paced (i.e. the user can perform an imaginary movement at will and the system is able to detect it in an asynchronous manner).

Towards this goal, we have explored two paths; first, we present the utilization of the Hjorth's descriptors in the SMR signal detection, which have been used in other fields to detect the change-point in time series (Section 3.1.2). This can be particularly useful in transforming the SMR detection system from cue-based to self-paced. Furthermore, the SMR data from the Phase I Pilot Trials are analyzed showcasing that motor-impaired people, the target users of the MAMEM technology, not only can produce distinguishable SMR signals, but already have a better performance on it, which can be attributed to unintentional training mechanisms (i.e. continuously trying to perform movements mentally after losing the physical ability to do so). Finally, we present the MAMEM version of the popular Tetris game (Section 3.1.3), mentioned hereafter as MultiModal Tetris (MM-Tetris), which utilizes SMR signals for rotating the game tetriminos, while gaze input is used for moving the tetrimino in the horizontal axis. Our experiments focus on investigating for the optimal methodology to detect self-paced SMR in the context of the MM-Tetris game.

In order to overcome the limited target detection performance in HCIs and apart from creating more sophisticated machine-learning techniques or adding further modalities, scientists have also exploited the users ability to perceive errors in order to improve the performance of HCIs. Towards this direction, the most common approach is by detecting Error-Related Potentials (ErrPs), a special type of Event-Related Potentials (ERPs) that appear shortly after the user recognizes an error. Previous studies have shown that ErrPs can be utilized to correct spelling errors in P300 spellers [14], adapting classifiers and reduce the calibration time and improve performance of code-modulated Visual Evoked Potentials (c-VEP) BCIs [66]. In a similar vein, MAMEM opts to utilize an error detection mechanism through EEG signals so as to offer an automatic correction system that will complement the gaze-based keyboard. Towards this goal, we first present a novel spatial filtering algorithm (Section 3.2.1) that directly maximizes the signal-to-noise-ration (SNR) of an ERP-related EEG signal. Then we analyze the ErrP data from the Phase I Pilot Trials, identifying the keyboard design shortcomings for eliciting ErrPs and based on this analysis, we propose a new keyboard design. Finally, we present a novel algorithm for detection typing errors that combines the ErrPs with gaze information to increase the detection performance of errors and validate the performance of the presented multimodal method using the newly designed keyboard.

3.1 SMR

In this section, we provide an analysis of SMR data from Phase I of our project. In addition to that, we analyze data from our first attempt to introduce a new protocol for SMR training based on the popular Tetris game. At first, we provide a short introduction to the utilized methods and algorithms for detecting the SMR signals. More specifically, in Section 3.1.1.1 a short description of the well - known CSPFB (Common Spatial Patterns with Filter Banks) algorithm is provided, while a novel feature extraction algorithm based on Hjorth's Descriptors [6] is presented in Section 3.1.1.2.

Afterwards, in Section 3.1.2 we present an experimental analysis on the Phase I Pilot trials data (cf. D6.4 for the data and protocol description). The first goal of this analysis is to investigate on the differences between the patient and the healthy control groups. For this we rely on the state-of-the-art methodology of CSP with FB, and our results indicate that the patients can provide better SMR signals for the detection of both between left and right imaginary movements and between movement and no movement. Then, we introduce the utilization of the Hjorth's descriptors in the SMR context. These descriptors with the aid of multivariate control charts have been used in other fields to detect the change-point of nonlinear timeseries [56]. This can be particularly useful for self-paced asynchronous BCI by identifying the initiation of Motor Imagery in the EEG-signal. Thus our initial objective is to investigate whether these descriptors can be successfully ported in the EEG domain for detecting imaginary movements through the SMR signals. Indeed, our analysis shows the potential of using Hjorth's descriptors in the typical SMR detection scenario compared to the state-of-the-art CSPFB algorithm. In the future, we plan to investigate how these descriptors can facilitate the asynchronous BCI scenario.

Finally, in Section 3.1.3 we provide information on the proposed MM-Tetris (MultiModal Tetris) game, a reinvention of the popular Tetris game to be be controlled by the user's eyes and mind. MM-Tetris can be used in order to provide a gamified and thus more appealing interface for training the users to produce better SMR signals. In this direction, we present the algorithms and methods that have been used in order to create the game. More specifically, first the interface along with the controls of the game is presented. Then we provide information on the data collection process. Finally, we present our method for transforming the cue-based SMR-detection to self-paced, as well as a methodology for evaluating the SMR detection rate in this new application scenario.

3.1.1 Methods and algorithms

3.1.1.1 Basic Approach - Conventional Method

The Common Spatial Pattern (CSP) algorithm is effective in constructing optimal spatial filters that discriminates two classes of EEG measurements in SMR - based BCI. However, the performance of this spatial filter is dependent on its operational frequency band. To avoid this effect in [5] a variation of CSP algorithm based on filter bands was proposed (CSPFB). This approach is adopted in our study. More specifically, the CSPFB algorithm consists of four basic steps: frequency filtering, spatial filtering, feature extraction and classification. The general architecture of the adopted approach is described in Fig. 15.

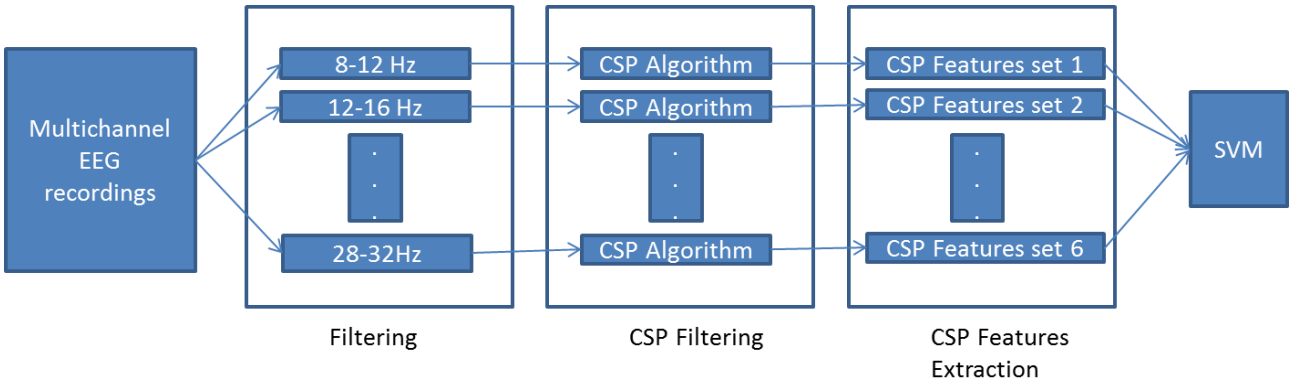


Figure 15: Architecture of CSPFB alogirthm.

The first stage employs a filter bank that bandpass filters the EEG measurements into multiple bands. The second stage performs spatial filtering on each of these bands using the CSP algorithm [42]. As we see, each pair of bandpass and spatial filter yields CSP features that are specific to the frequency range of the bandpass filter. In the third stage, we extract the CSP features from the filter bank. The fourth stage uses a classification algorithm (SVMs) to model and classify the selected CSP features.

3.1.1.2 Hjorth's Descriptors

Based on the standard deviation (SD) of the signal and its derivatives Bo Hjorth [6] defined three parameters that can provide a quantitative description for any signal in both time and frequency domain. These parameters, named as Hjorths Descriptors (HD), are Activity, Mobility and Complexity and are defined based on the spectral moment (m_k) and variance (σ_k) as follows: Activity is defined as the mean power or the variance of the signal and is measured as the SD of the signals amplitude:

$$A = m_0 = \sigma_0^2 \tag{1}$$

Mobility can be interpreted as the mean frequency of the signal and is estimated as the ratio of the SD of the signals first temporal derivative to the SD of the signal:

$$M = \sqrt{\frac{m_2}{m_0}} = \frac{\sigma_1}{\sigma_0} \tag{2}$$

Complexity is the change in frequency of the signal and is quantified as the ratio of the Mobility of the signals first derivative to the Mobility of the signal:

$$C = \frac{\sqrt{\frac{m_4}{m_2}}}{\sqrt{\frac{m_2}{m_0}}} = \frac{\frac{\sigma_2}{\sigma_1}}{\frac{\sigma_1}{\sigma_0}} \tag{3}$$

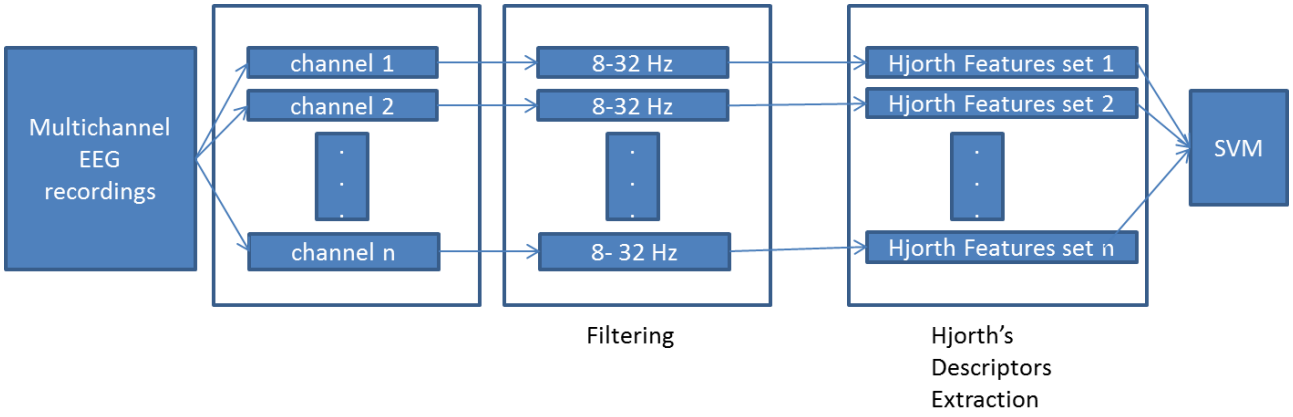


Figure 16: Architecture of Hjorth-based algorithm.

The signal processing pipeline based on Horth's descriptors is depicted in Fig. 16. As we can see, the recordings of each channel are bandpass filtered from 8 to 32 Hz. Then Activity, Mobility and Complexity are extracted from each channel and are then concatenated to produce the final feature set, which is used as input to the SVM classifier.

3.1.2 Pilot Phase I trials analysis

Data gathered during the SMR tasks from Pilot Phase I trials (cf. D6.4) were used for analysis purposes in this section. More specifically, the EEG data regarding only the calibration part of the SMR task were used for the cohorts from the Parkinson's Disease Group (PG) and the Neuromuscular Disease Group (NG) and the Spinal Cord Injury Group (SG), excluding participants PP3 and SP1 that did not complete the trial. The separate analysis of the calibration data was made in an effort to focus only in the brain oscillations initiated only by the Motor Imagery (MI) task while excluding any additional brain activity that may arise from the external stimuli provided during the feedback sessions.

The separation of the left imaginary movement trials from the right imaginary movement trials, mentioned hereafter as left-right scenario, was the initial rationale of the experimental procedure. In addition, using the same data, we examined the movement-nomovement scenario where each trial was divided in pre-stimulus and post-stimulus periods based on the provided timestamps. The objective of this scenario is to investigate whether we can detect differences between the cases where the participants intended to make any imaginary movement, either left or right, and the cases that they were not. This scenario is more fitting to the MAMEM multimodal system, with respect to both the reading-selection mode navigation and the Tetris game (more details on the MAMEM multimodal system can be found in Section 5).

3.1.2.1 Implementation details

The evaluation procedure was performed in a single-subject analysis. A leave-one-out-cross-validation (LOOCV) scheme was adopted to validate the accuracy of the proposed algorithms and Support Vector Machines (SVMs) with a linear kernel and MATLAB's default parameters were used for the classification tasks.

In the movement-nomovement scenario each trial was separated into two parts, consisting of 4 seconds each, considering the first 4 seconds (i.e. 1-4 seconds) as the nomovement class and the next 4 (i.e. 5-8 seconds) as the movement class. Likewise in the left-right scenario only the second 4-second period was used as input signal. The above decision was made with respect to the experimental design, as the initiation of the imaginary movement and therefore the brain de/synchronization is expected after the fourth second. EEG signals have been acquired using the EbNeuro device resulting in multichannel recordings of 62 channels. More information about the recording procedure can be found in D6.4.

3.1.2.2 Comparing patients to healthy

Our objective in this section is to compare the ability of the two groups (healthy control and patients) in producing distinguishable SMR signals during the two previously described scenarios. In this direction, we have relied on the state-of-the-art CSPFB algorithm presented in Section 3.1.1.1.

Our first experiment focuses on discriminating between left and right imagined movements (left-right scenario) with the obtained accuracy for each individual being displayed on Table 4. We can see that the average accuracy for all groups is similar, with the NG participants (49.2%) performing slightly better than the other two groups, (48.4%) (47.70%) for PG and SG respectively. Nevertheless, we observe a significant variability among subjects, as there are several participants with high classification accuracy (reaching 60%), whereas others do not even reach the baseline (50%). Linking the subjects' accuracies with their corresponding IDs we can see that the ones with the highest accuracies arise mostly from the motor impaired participants,

regardless the group they belong to. In this direction, participants were further categorized to patients and able-bodied, creating six new groups; Parkinsons able-bodied Group (PAG), Parkinsons patients Group (PPG), Neuromuscular able-bodied Group (NAG), Neuromuscular patients Group (NPG), Spinal able-bodied (SAG) and Spinal patients (SPG). Figure 17 presents the average accuracy for each newly-defined group, with PPG and NPG performing significantly better than the able bodied groups and SPG performing slightly better. We can therefore confirm the initial observation that patients can perform better in this scenario opposed to the healthy-control groups.

Table 4: Accuracy of CSPFB algorithm for the discrimination of the left-right scenario

PG		NG		SG	
ID	CSPFB (%)	ID	CSPFB(%)	ID	CSPFB(%)
PP1	57.5	NP1	65	SP1	-
PP2	37.5	NP2	40	SP2	57.5
PP3	-	NP3	62.5	SP3	42.5
PP4	65	NP4	42.5	SP4	47.5
PP5	47.5	NP5	47.5	SP5	47.5
PP6	60	NP6	52.5	SP6	45
PH1	52.5	NH1	52.5	SH1	47.5
PH2	45	NH2	57.5	SH2	45
PH3	42.5	NH3	45	SH3	55
PH4	37.5	NH4	55	SH4	52.5
PH5	37.5	NH5	37.5	SH5	40
PH6	50	NH6	32.5	SH6	45
mean	49.20	mean	48.4	mean	47.70

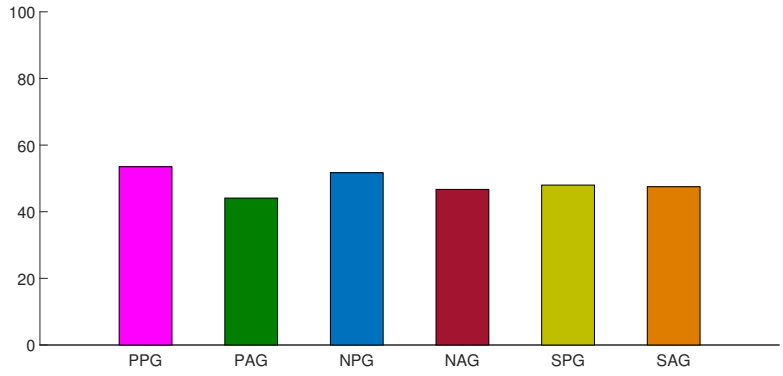


Figure 17: Average Accuracy of groups separated to motor impaired and able bodied for the left-right scenario.

Also, part of our analysis was to examine the ability of CSPFB algorithm to discriminate between EEG trials that contain motor activations and EEG trials that do not contain motor activations (movement-nomovement scenario). The results of this comparison for the three initial groups can be found in Table 5. It is evident that, in this scenario, the NG and SG subjects perform significantly better than the PG subjects, with the difference in average accuracy reaching 10% (54.1% 53.9% and 45.8% respectively for NG, SG and PG). Despite the observed gap between the groups, we can identify the trend observed in the first scenario being carried out here as well, while observing the individuals results. One can notice (see Figure 18) the superiority of the patients' performance against the one of their matched able-bodied participants. It is important to mention that there is a slight increase in the difference between able-bodied and motor impaired. This is an expected outcome, as the classification task for the second scenario is considered to be more straightforward due to the significant difference in the

brain activations when the participant is engaged to a task and when there is no involvement at all (resting state).

Table 5: Average Accuracy of groups separated to motor impaired and able bodied for the movement-nomovement scenario

PG		NG		SG	
ID	CSPFB (%)	ID	CSPFB(%)	ID	CSPFB(%)
PP1	43.75	NP1	58.75	SP1	-
PP2	55	NP2	47.5	SP2	60
PP3	-	NP3	55	SP3	61.25
PP4	51.25	NP4	58.75	SP4	63.75
PP5	38.75	NP5	56.25	SP5	41.25
PP6	55	NP6	61.25	SP6	51.25
PH1	43.75	NH1	63.75	SH1	63.75
PH2	43.75	NH2	46.25	SH2	46.25
PH3	27.5	NH3	40	SH3	71.25
PH4	47.5	NH4	50	SH4	51.25
PH5	53.75	NH5	56.25	SH5	43.75
PH6	43.75	NH6	55	SH6	38.75
mean	45.80	mean	54.10	mean	53.9

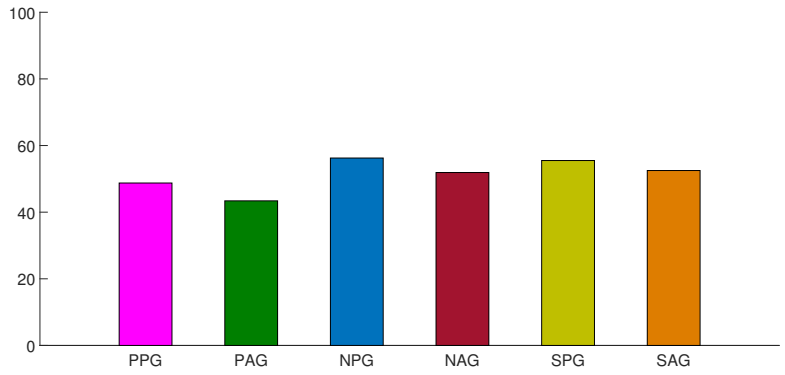


Figure 18: Average Accuracy of groups separated to motor impaired and able bodied for the movement-nomovement scenario.

3.1.2.3 Evaluating Hjorth's descriptors in the SMR conext

In addition, the ability of the Hjorth-based algorithm to discriminate the movement-nomomement trials was examined. The movement-nomovement scenario was chosen to continue experimenting with as it is the most prominent for being used in the final MAMEM system. The purpose of this analysis was to investigate whether Hjorth's Descriptors can be adequately adjusted to the MI EEG signals. A comparison between the CSPFB algorithm and the aforementioned algorithm is provided in Table 6. It is evident that the HD algorithm performs significantly better compared to the CSPFB both in the group-based and the subject-based analysis. It is also worth mentioning that when the HD algorithm is selected the accuracy is higher in all but one (i.e. SH1) subjects compared to the typical CSPFB algorithm. Subsequently, this notion carried out in the average accuracy. Overall, in all comparisons made between the groups, the difference in classification accuracy is prominent ($\geq 20\%$). Moreover, there are cases (i.e. subjects) where the improvement can exceed 30% (e.g. PH3, NH2 and SP5).

Furthermore, we can note that the two observation made when signals were analyzed using CSPFB can be made in this case as well. Firstly, PPG, NPG and SPG (i.e. patients) perform at a higher level compared to the matched control groups (PAG, NAG and SAG) with a similar difference. Secondly, the NG and SG participants outperform the PG participants with the

difference in the overall accuracy exceeding 10%. All the above observations, increase the possibility that the motor-impaired participants can adapt better to the MI scenarios.

It is evident that the proposed methodology can not only provide satisfactory results in the movement-nomevement scenario but also can surpass the classification scores provided by a well-established methodology. The ability to use it to detect the change-point in the EEG-signal remains to be investigated, as it is an intriguing perspective for asynchronous BCIs.

Table 6: Comparison of CSPFB and HD in the movement-nomovement scenario

PG			NG			SG		
ID	CSPFB (%)	HD (%)	ID	CSPFB(%)	HD(%)	ID	CSPFB(%)	HD(%)
PP1	43.75	63.75	NP1	58.75	90	SP1	-	-
PP2	55.00	76.25	NP2	47.50	66.25	SP2	60.00	62.50
PP3	-	-	NP3	55.00	63.7	SP3	61.25	85.00
PP4	51.25	71.25	NP4	58.75	90.00	SP4	63.75	91.25
PP5	38.75	62.50	NP5	56.25	77.50	SP5	41.25	85.00
PP6	55.00	75.00	NP6	61.25	88.75	SP6	51.25	85.00
PPG	48.75	69.75	NPG	56.25	79.40	SPG	55.50	81.75
PH1	43.75	57.50	NH1	63.75	83.75	SH1	63.75	61.25
PH2	43.75	55.00	NH2	46.25	85.00	SH2	46.25	78.75
PH3	27.50	61.25	NH3	40.00	70.00	SH3	71.25	83.75
PH4	47.50	77.50	NH4	50.00	76.25	SH4	51.25	82.50
PH5	53.75	75.00	NH5	56.25	80.00	SH5	43.75	57.50
PH6	43.75	62.50	NH6	55.00	57.50	SH6	38.75	73.75
PAG	43.40	64.80	NAG	51.90	75.40	SAG	52.50	72.90
mean	45.80	67.10	mean	54.10	77.40	mean	53.90	76.90

3.1.2.4 Discussion

In this study, we examined the possibility of using a SMR-oriented BCI by motor disabled people comparing their performance to matching control groups of healthy patients.The results showed that it is possible for participants with motor impairment to achieve acceptable classification scores, a trend that was not found for the healthy groups. The accuracy levels for the PPG, NPG and SPG groups roughly reached 70%, 80% and 80% respectively, whereas the performance of the PAG, NAG and SAG groups was 5% lower than their equivalent patient groups in accuracy units. This can be attributed to the fact that significant mental effort is required by patients to make a move or even attempt in their everyday life for several years. This could be considered as a training session for them making the familiarization process of the SMR protocol pretty straightforward. These results are promising regarding BCIs with the use of SMR, as the agonizing training process that could even take months can be shortened. Nevertheless, user training is imperative so as to reach sufficient accuracy levels for real world BCI applications. For this reason, and considering that most of Pilot I trials participants found the SMR training sessions to be tedious, in the next section we present MM-Tetris, a gamified environment to be used for user training.

Finally, considering that self-paced imaginary movement detection is a key aspect for utilizing SMR in real-world applications (i.e. the user can trigger the imaginary movement detection at any time without any cue), we presented the Hjorth's descriptors. These descriptors combined with some change-detection function have the potential for being used in a self-paced BCI scenario. However, so far, they have not been used for SMR data analysis, and thus our first goal in this section was to investigate whether they can be used as descriptors for SMR signal analysis. The results show that it is a promising direction and in the future we plan to investigate on how they can facilitate self-paced BCIs.

3.1.3 User training for SMR using Tetris

MM-Tetris is the multi-modal reinvention of the popular Tetris game, modified to be controlled without the conventional means (i.e. mouse and/or keyboard) but instead with the users eye-movements, mental commands and bio-measurements. Our motivation for re-inventing Tetris, is that in the context of BCI applications (and especially the ones relying on SMR) it is imperative to not only train the system to recognize but also the user to produce distinguishable signals. In this respect, the use of an application that will allow users to train themselves needs to precede the actual use of a BCI interface. Typical training approaches consist in presenting the user with arrows as cues to indicate what type of movement they are asked to perform mentally, while providing feedback as to what movement the system has detected. This feedback allows the users to adapt their mental strategy by optimizing their mental movement command in order to maximize the detection performance of the feedback mechanism. However, this is a tedious process and the users get bored and frustrated before mastering their SMR signals to a sufficient extent. Alleviating this problem, the MM-Tetris game can serve as a gamified SMR training application that will engage the user to keep training themselves.

In the proposed version of the game (Fig. 19), the use of eye-movements and mental commands work in a complimentary fashion, by facilitating two different controls, the horizontal movement of the tiles through the coordinates of the gaze and the tile rotation through SMR signals detection respectively. Additionally, bio-measurements provide the stress levels of the player, which in turn determine the speed of the tile's drop. In this way, the three modalities smoothly collaborate to facilitate playing a game like Tetris.

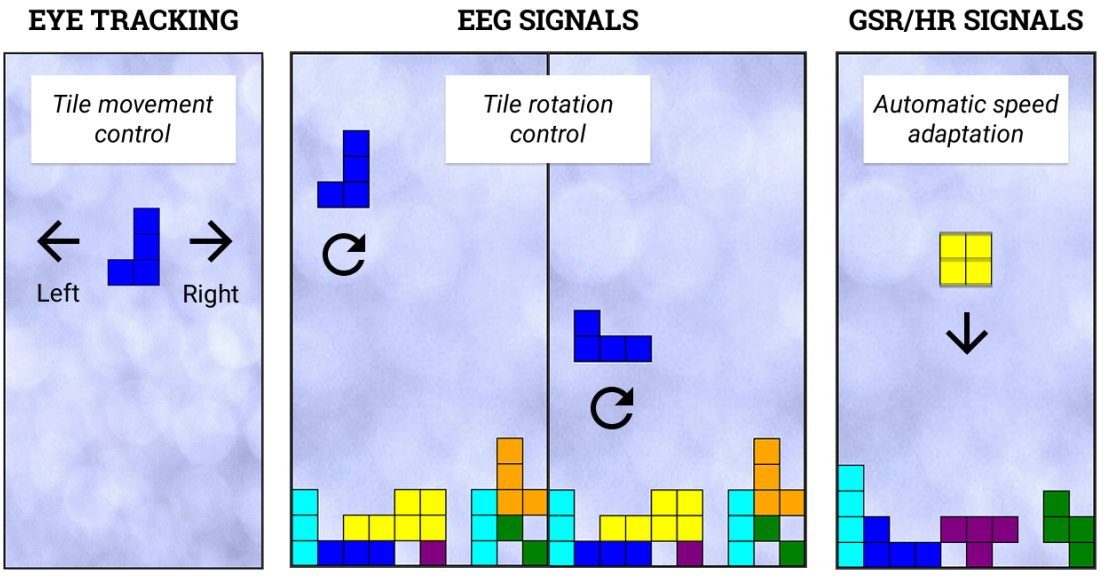


Figure 19: The MM-Tetris interface and commands.

In designing a functional MM-Tetris game, the first challenge we face is to provide self-paced SMR-based movement detection. Indeed, most existing works in the literature (including the work presented so far in the previous deliverables of WP3) focus on maximizing the accuracy of detecting SMR signals by choosing the "time window" of the trials that maximizes the accuracy during testing. However, this does not simulate a real application scenario where the user may do an imaginary movement at any arbitrary time and the system should detect this intention for movement in an asynchronous way. Motivated by this, the objective of this section is to investigate what is the optimal methodology for self-paced imaginary movement detection. Towards this goal, first we present the data collection methodology and the experimental protocol (Section 3.1.3.1 and second we analyze the data for self-paced movement detection (Section 3.1.3.2.

3.1.3.1 Experimental Protocol - Data Collection

The experiment begins with a calibration session using the OpenVIBE framework during which the participants are asked to perform imaginary movements of their left or right hand. An interface of visual cues is presented indicating whether the participants should perform the imagination of movement as well as which of the hands to move. More specifically, the calibration session starts with a black screen that lasts for 40 seconds during which the participant is asked to do nothing, i.e. not imagining any movement for the purpose of recording a baseline signal. Afterwards, an iteration of 40 trials is initiated consisting of 20 trials for the imagination of left hand movement and 20 for the imagination of right hand movement. The beginning of each trial is marked by a green cross that lasts for 1 second and allows the participants to prepare for the execution of imaginary movement. Then, a red arrow pointing towards the left or right direction instructs the participants to perform the movement and also which of their hands to use. After 5 seconds the arrow disappears and is followed by a black screen that lasts for a random duration between 5 and 20 seconds. The whole duration of the calibration session is approximately 20 minutes.

A classifier is then trained based on this calibration session and the participant is asked to perform the mental imagery task at will with the goal of controlling a feedback bar on the screen. The length of this feedback bar indicating the strength of the participants mental imagery skill, based on the classification score. The feedback bar was also coloured as green or red based on whether the EEG input was classified as movement or no movement. The purpose of this short session was to train the user on performing the mental imagery task and to find the best technique that will allow them later on to play the Tetris game. This training session lasted until the participant felt confident enough to control the feedback bar sufficiently.

At the final stage of the experiment, the participants are given the instructions on how to play the Tetris game. The eye-tracker is first connected and a calibration procedure is performed during which the participants are asked to fixate their gaze on 9 dots appearing on multiple locations on the screen. Then, the game starts with the first tetrimino appearing at a locked position on the top of the screen. The participants are given a few seconds in order to try to rotate the tetrimino at the desired angle using their brain signals. The exact duration of the position lock is determined based on the classification output which runs continuously in the background. Specifically, the classifier communicates the classification score each second to the Tetris game. If the score exceeds a certain threshold the tetrimino is rotated. The tetrimino unlocks when a decrease of the classification score is observed for 3 times since the last time the tetrimino was rotated. Afterwards, the rotation angle of the tetrimino is locked and it starts to drop while also allowing the participants to move it on the horizontal axis using input of the eye-tracker. The tetriminos are being moved by constantly following the participant's gaze location on the screen one step at a time. If the participants are looking at the same location for 5 seconds continuously, then the tetrimino becomes locked completely and drops down fast until it reaches the ground. In order to assist the participants, a feedback bar was informing them about the strength of the mental imagery task during the rotation stage of the game. Also a shape similar to a crosshair was assisting them with regards to the aiming of their gaze as well as locking their gaze when the correct position of the tetrimino is reached, thus avoiding undesired gaze movements. The EEG signals were recorded by the Enobio8 headset (cf. D2.3) using 8 wet electrodes that were placed on the CP1, CP2, C3, C1, C2, C4, FC1, FC2 areas of the international 10-20 system and the sampling rate of the system was 500Hz.

3.1.3.2 Data Analysis

Data Segmentation: The recorded sessions are segmented by overlapping windows of 2 seconds duration. The step of each window was set to as 100 samples or 0.2 seconds. Each window from now on referred to as trial, consisted of 1000 samples each of which was annotated as 1 or 0 based on whether the sample was recorded when the participant was performing a mental movement regardless of the direction. The label of each trial was then set based on

the majority of the 1000 samples to either 1 (movement) or 0 (no movement). After the segmentation, we have n_m and n_n number of trials for the movement and no movement labels respectively. Considering that the users were asked to perform no movement for more time compared to the movement case, the number of n_n trials is bigger than the number of movement trials n_m ($n_n > n_m$), and as such, the two classes are imbalanced.

Offline classification: The generated dataset during the calibration session is divided in half so as to produce a training and test set of $(n_m + n_n)/2$ trials. In order to tackle the class imbalance problem, which can hinder the learning process of the classification model, we balance the training set by randomly undersampling the trials that correspond to the majority class (no movement in this case), so as each class will have the exact same number of trials. The testing set on the other hand is left unbalanced to simulate a real-world application scenario, where the classification model will be applied in a continuous EEG signal stream, which is expected to be imbalanced. Each trial is then filtered with a butterworth bandpass filter in the ranges between 7Hz-36Hz and the CSPFB features are extracted for each trial. The filter banks that were used for extracting the features were in the frequency ranges of 8Hz-12Hz, 12Hz-16Hz, 16Hz-20Hz, 20Hz-24Hz, 24Hz-28Hz and 28Hz-32Hz. As previously mentioned, the features from the first half of each session were used to train a linear kernel SVM classifier and the other half was used for testing.

The testing was performed as follows; the unknown to the classifier portion of the recorded session was scanned with a sliding window of length 2 secs and step 0.2 secs. Each trial was passed to the previously trained classifier in order to retrieve a score. After 1 second, meaning that 5 scores are generated by the classifier, the sign of the median of the scores was used to determine whether to issue a rotation command or not. If the sign is positive, a rotation command is simulated and the classification is paused for 5 seconds.

Table 7: Offline experiments for evaluating Tetris performance.

Subject	**Precision**	**Recall**	**Accuracy**
S1	39.65	76.66	89.06
S2	41.09	100	93.28
S3	23.80	83.33	83.59
S4	35.21	83.33	88.9
S5	37.83	82.35	88.12
S6	29.50	60	83.9
S7	32.5	86.66	88.4
S8	40.67	80	89.84
avg. acc.			83.75
baseline acc. (do nothing)			76.56

The results of this method can be found in Table 7 that were performed by 8 in-house participants. The precision indicator refers to the number of the correct rotation commands that were issued (i.e. detected during the period that the subject was performing the movement) divided by the total rotation commands. The recall measurement refers to the number of correct rotation commands divided by the total number of movements that were performed during the experiment. Finally, the accuracy metric represents the percentage of the seconds that the classifier behaved according to the user's will. To calculate this accuracy metric we used Equation 4 where n_{sec} refers to the number of seconds of the testing set, n_{move} were the number of times that the participant was asked to perform a movement by the experiment interface, TP was the number of rotations that were detected by the system and were during the time the participant was performing a mental movement and FP was the number of rotations detected by the system during the time the participant was not performing any action.

$$acc = \frac{n_{sec} - 5 \cdot (n_{move} - TP) - FP}{n_{sec}} \tag{4}$$

In short, this accuracy metric calculates the number of seconds that the system was performing according to the intention of the user and places it on the numerator. The total number of seconds is placed on the denominator, thus an accuracy of 80% for an 100 second experiment means that the system was performing correctly for a total of 80 seconds. A baseline accuracy of a classifier that does not detect any rotations is also reported on this table as 76.56%. An example of the performance of our classifier can be seen on Figure 20, where the blue pulses represent the intention of the user, i.e. the times the user intended to perform the mental movement, the green lines are when the system detected a rotation command and the red lines are when the system did not detect any mental movement.

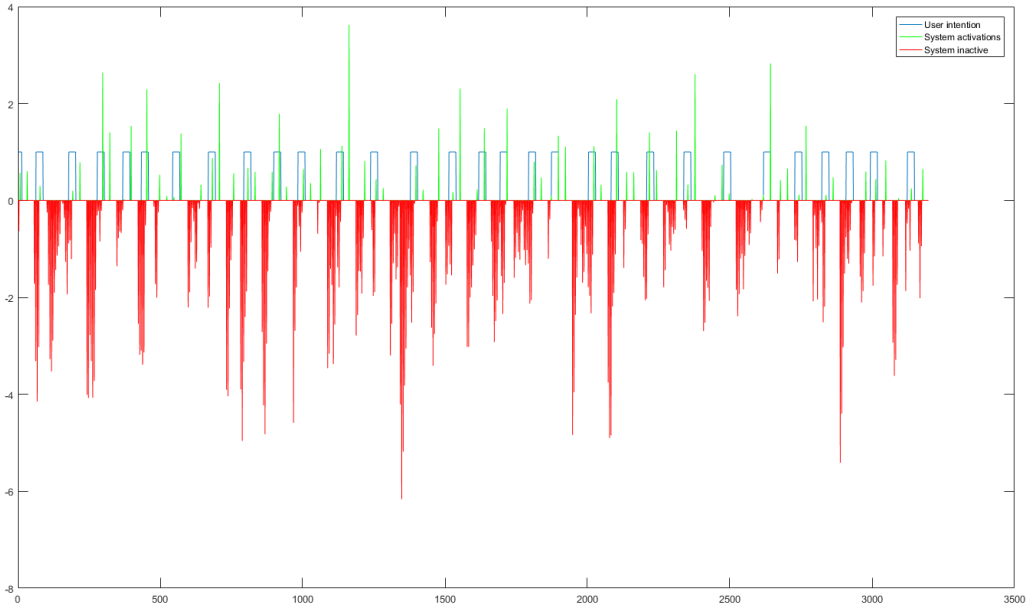


Figure 20: Subject 2 performing mental movement (blue lines) and the result of our classifier colored as green(rotation command) and red (no rotation command).

Online classification framework: For the online classification we used the same classifier as for the offline case trained with the whole calibration session instead of splitting the dataset in half. The EEG signal is again sampled with a moving sliding window which moves for 0.2 seconds each time and produces a classification score. After each second, the median classification scores of the past 5 windows is calculated and communicated to the Tetris game. Since the scores are negative if the window segment corresponds to the no movement class and positive for the movement class, this practically means that in order to have a movement outcome, at least 3 out of the 5 past windows must produce a positive classification score.

3.2 ErrPs

In this section, we investigate thoroughly the utilization of ErrPs in multimodal HCI systems, and more specifically in a gaze-based keyboard. Towards this goal, our first objective is to optimize the detection of ErrPs in an algorithmic fashion, by devising spatial filters that maximize directly the signal-to-noise ratio (SNR) of the EEG signals applicable to any ERP signal (ErrPs are a special case of ERPs as mentioned above). The proposed spatial filters are evaluated on existing and established ERP benchmarks (Section 3.2.1). Then, we analyze the EEG data with respect to the keyboard ErrPs from the Phase I Pilot Trials (Section 3.2.2). Based on the

results of this analysis, we optimize the design of the keyboard and its animations (e.g. letter selection preview, dwell animation, etc.). Finally, we propose a multi-modal algorithm for the more accurate detection of the eye-typing errors that takes into account information from both the EEG sensor and the eye-tracker (Section 3.2.3).

3.2.1 Spatial filtering for SNR maximization

Spatial filters can be separated in two distinct categories. On one hand, are those that are data-independent (i.e. the weights for each sensor are predetermined), such as the Laplacian filter and the common average re-reference technique [48]. These filters are mostly used in electroencephalography analysis to diminish volume conduction effects. On the other hand, are the filters which are data-dependent (i.e data are necessary for filter calculation). In particular for use within ERP-BCI paradigms, the xDAWN [62] and Canonical Correlation Analysis (CCA) related filters [67] appear as the most promising ones [63]. The former aims to maximize the Signal-to-Signal-plus-Noise Ratio (SSNR), while the latter the correlation between the single-trial signals and the average evoked response.

In this section, we present a new method for spatial filtering that resorts to the well-established temporal patterning of ERP responses and maximizes the separability among single-trial responses belonging to distinct brain states. The use of this concept had been initiated successfully in a BCI paradigm that relied on transient visual evoked responses [41]. Therein, a differential sensor montage was introduced based on the bipolar pattern emerged in the topographical distribution of the evoked response. Here, we elaborate on a generalized methodology for designing spatial filters based on the Fisher's discriminant analysis of single-trial temporal patterning. The main goal of this research is to provide a method that increases the SNR of the recovered response while enhancing the differences between responses of different type or brain state. We show here, that Fisher's separability criterion constitutes the natural extension of a standard SNR estimator suitable for multi-trial ERP responses, and can therefore naturally lead to spatial filters conforming to discriminant analysis. The introduced filters, named hereafter Discriminant Spatial Filters (DSF), offer a flexible framework and are characterized by computational efficiency. The approach is validated using two independent ERP datasets, one concerning ErrPs and the other P300 responses, in direct comparison with CCA and xDAWN methodologies.

3.2.1.1 Methods

An ERP signal is a recorded brain response that is the direct aftereffect of a specific event (e.g. the perception of an erroneous action). Due to low SNR of the recorded signals, several single trial responses (i.e. repetitions) can be aggregated in the so-called ensemble average waveform. Averaging, typically, ensures that the background noise (i.e. brain's electrical activity that is not related to the stimulus) is cancelled out and only a small fraction survives the averaging. Two assumptions must hold so that averaging will increase the SNR of the ERPs. Firstly, the signal of interest should consist of phase-locked responses with invariable latency and shape. Secondly, the background noise should follow a random Gaussian process of zero mean, uncorrelated between different recordings and not time-locked to the stimulus [43].

Typically, SNR is used in order to validate the credibility of one signal. However, the SNR hardly offers any information regarding the separability among groups in a classification problem, where distinct types of response need to be considered. We discuss next, how Fisher's separability criterion can blend SNR measurements from distinct groups into a single class discrimination score. Finally, we extend this idea to the case of multichannel recordings so as to formulate a suitable cost function for spatial filter design.

Preliminaries Considering that the average waveform is "practically" free of noise (due to a sufficiently large number of recorded responses), the calculation of SNR may proceed as follows.

Let us denote by $X^i = [s_1^i, s_2^i, \cdots, s_m^i]^T \in R^{m \times p}$, the i-th multichannel single trial response (i.e. a matrix that contains a single brain response recorded from multiple sites) with DC offset removed, where m denotes the number of sensors and p the number of samples. Hence, $\bar{s}_k = \frac{1}{N_x} \sum_{i=1}^{N_x} s_k^i$ represents the average response, which is a p-dimensional vector, of N_x single trial responses for the k-th sensor. Similarly, the average multichannel response is expressed as $\overline{X} = \frac{1}{N_x} \sum_{i=1}^{N_x} X^i = [\bar{s}_1, \bar{s}_2, \cdots, \bar{s}_m]^T$.

Consequently, the Signal Power (SP) and Noise Power (NP) of k-th sensor are defined as

$$SP_k = \frac{1}{p} \|\bar{s}_k\|_2^2 \tag{5}$$

$$NP_k = \frac{1}{p(N_x - 1)} \sum_{i=1}^{N_x} \|\bar{s}_k - s_k^i\|_2^2 = \frac{1}{p} c_{s_k} \tag{6}$$

where c_{s_k} can be interpreted as a measure of dispersion of the single-trials from the average waveform. Formally this quantity corresponds to the energy of noise. Equation 6 implies that every trial is an addition of the ERP signal and noise. Then the corresponding SNR of the k-th sensor is the ratio of SP_k over NP_k.

$$SNR_k = \frac{\|\bar{s}_k\|_2^2}{c_{s_k}} \tag{7}$$

Observing the above equation, we could easily conclude that in an ideal scenario we would expect identical ERPs with zero scatter among them.

In a typical ERP-based BCI we aim at distinguishing between two types of brain response, ERP vs nonERP, or equivalently to discriminate between two groups of single trial responses. One could easily separate the two groups if the temporal patterning is consistent within the groups and deviates across groups. In such a case, the corresponding averaged waveforms would differ significantly, while the dispersion for both groups will be low. The first condition can be formulated as high power for the differential averaged signal (i.e. the difference of the average waveforms). The second condition can be expressed as the sum of two terms, expressing the noise power of either group. These two condition ensure that indeed the two groups are distinguishable while the single-trial responses are identical within each group. By forming the corresponding ratio SP/NP and simplifying the expression, we end up with the Fisher's separability criterion used in standard discriminant analysis [19].

Considering two groups of single trial responses, X^i and Y^j with N_x and N_y trials each, the separability index of the k-th sensor becomes

$$J_k = \frac{\|\bar{s}_k(X) - \bar{s}_k(Y)\|_2^2}{c_{s_k}(X) + c_{s_k}(Y)} \tag{8}$$

Discriminant Spatial Filters Having defined all the necessary notations as previously, we are now in position to formulate the spatial filter design as an optimization problem. Although equation 8 could be used for sensor selection, we are looking for a spatial filter that captures this idea and extends it to multiple sensors. Since surface EEG measurements may reflect the neural response of interest in several electrodes placed over the scalp, and with a varying grade of credibility, we seek the linear combination (i.e. a weighted sum of sensor signals to create a "virtual" signal) that will maximize the Fisher's separability criterion.

Let $w = [w_1, w_2, \ldots, w_m] \in R^m$ be a spatial filter (weight vector). Denoting $C_X = \frac{1}{N_x - 1} \sum_{i=1}^{N_x} (X^i - \bar{X})(X^i - \bar{X})^T$, the average response and noise energy for the "Virtual Sensor" (VS) for the X^i group of trials can be written as

$$\bar{s}_{vs}(X) = w_1 \bar{s}_1 + \ldots w_m \bar{s}_m = w \begin{bmatrix} \bar{s}_1 \\ \vdots \\ \bar{s}_m \end{bmatrix} = w\bar{X} \tag{9}$$

$$c_{vs}(X) = \frac{1}{N_x - 1} \sum_{i=1}^{N_x} \|w\bar{X} - wX^i\|_2^2 = wC_X w^T \tag{10}$$

Similarly, for the Y group the corresponding average response and noise energy are calculated as

$$\bar{s}_{vs}(Y) = w\bar{Y} \tag{11}$$

$$c_{vs}(Y) = wC_Y w^T \tag{12}$$

following the similar denotation, as above, for the Y group where $C_Y = \frac{1}{N_x - 1} \sum_{i=1}^{N_x} (Y^i - \bar{Y})(Y^i - \bar{Y})^T$.

The Fisher's separability index of the "virtual sensor" can now be expressed as:

$$J_{vs} = \frac{\|\bar{s}_{vs}(X) - \bar{s}_{vs}(Y)\|_2^2}{c_{vs}(X) + c_{vs}(Y)} \tag{13}$$

Substituting equations 9-12 to 13, J_{vs} can be easily reformulated and hence maximized by:

$$w = \arg\max_w \frac{w(\bar{X} - \bar{Y})(\bar{X} - \bar{Y})^T w^T}{w(C_x + C_Y)w^T} \tag{14}$$

It is easy to notice, that the ratio in right hand side is a generalized Rayleigh quotient and hence the solution can be obtained by solving the generalized eigenvalue decomposition problem. The desired spatial filter corresponds to the eigenvector associated with the largest eigenvalue. We should note here that the above formula could be easily adapted to enhance the discriminability in problems that only one group of responses is available (i.e. ERP needs to be recovered from background activity) or problems that concern the simultaneous separation of more than two groups, following an approach similar to [61].

3.2.1.2 Experiments

In this section, we present the classification accuracies obtained by the proposed method and we compare the results to the ones obtained with xDAWN and CCA. These two methods not only represent the current trends in spatial filters but also share the same character with ours, that is of a data-dependent filter. The used datasets cover two widely used ERP categories in BCIs, those of P300 and ErrPs, and constitute a proper basis for the desired evaluation. A common processing, classification and evaluation framework was employed in each dataset for all of the compared methods.

In all cases, the signals were bandpass filtered between 1-16Hz using a zero-phase FIR filter and all the channels were used for the calculation of spatial filters. The single-trial patterns were classified using Linear Discriminant Analysis (LDA) [57]. The classification performance was estimated based on a 10-fold cross validation. To avoid any bias associated with the randomized partition into folds, the overall procedure was repeated 100 times.

Error-Related Potentials Dataset For the detection of ErrPs we used the data introduced in [68]. Nine subjects had to use a c-VEP speller and if the BCI detected the wrong letter, an ErrP was elicited by the erroneous feedback. EEG was recorded from 32 electrodes with a sampling rate of 600Hz. To remove the eye artefacts EOG-regression method was used. Approximately 1300 single-trial segments, of one second duration each, were extracted, for each subject, starting from the appearance of the presented letter. There were two types of response, the ErrP related one and the evoked response associated with the presentation of a correct letter.

The mean value of accuracy, associated with each one of the ten folds, was computed as an aggregate measure of performance. Table 8 includes the accuracy level as averaged over the

100 repetitions (of the 10-fold cross validation). The overall scheme was repeated using the spatial filters from the CCA and xDAWN approaches. On average (across 9 different subjects), DSF approach led to the best performance. Using the one-tailed Wilcoxon signed-rank test, we verified that this trend was statistically significant ($p<0.001$).

Subject	xDAWN	CCA	DSF
S01	96.32%	96.62%	97.34%
S02	92.92%	87.36%	94.71%
S03	97.92%	96.85%	98.40%
S04	98.55%	98.78%	98.60%
S05	89.42%	86.71%	93.69%
S06	96.28%	94.10%	96.37%
S07	98.93%	98.98%	99.07%
S08	83.46%	82.36%	88.91%
S09	96.90%	95.97%	97.39%
AVERAGE	94.52%	93.08%	96.05%

Table 8: Accuracy for the Error-Related Potentials dataset: This table contains the average accuracy for 100 different 10-fold cross validations splits per subject.

Figure 21 depicts the average error-minus-correct waveform for both the Cz electrode, which is the most prominent for the ErrP components, and the DSF-derived traces. It becomes evident that in the virtual sensor the differences among erroneous and correct responses are intensified. In more quantitative terms, we report here the Fisher criterion level for the virtual sensor and the Cz electrode, which was 0.1262 and 0.0675 respectively.

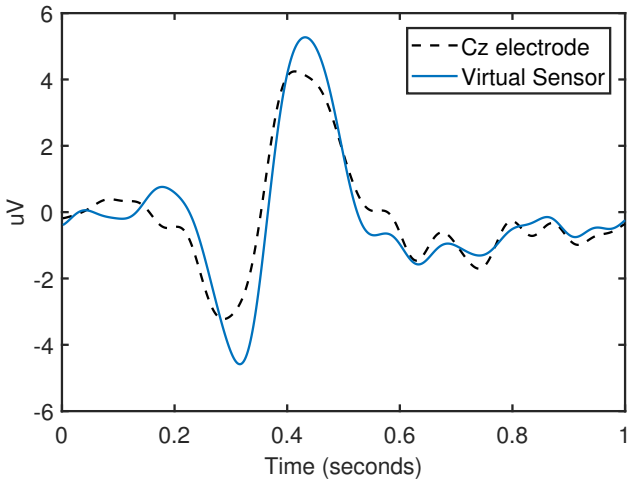


Figure 21: Grand average of error-minus-correct waveforms across all participants at Cz electrode and the corresponding temporal traces produced by DSF (after L2-norm normalization of the corresponding weight vectors).

Auditory Oddball Paradigm (P300) Dataset Auditory oddball paradigm dataset was acquired from two healthy participants [64]. The experiment was conducted in two different conditions, active and passive. In the former condition, the participants were instructed to listen to a tone stream and count the number of odd tones. In the latter condition, counting was not was required. The EEG sampling rate was 512Hz.

In this dataset, we report results from our attempt to distinguish the responses to frequent from the responses to rare auditory stimuli, separately for the two states, and in a subject-specific manner. Single-trial segments of one second duration were extracted based on the auditory stimulus onset. The cognitive response is known to present a clear positive deflection, namely P300, around 300 ms after the delivery of the rare tone only.

Table 9 presents the results obtained by the compared methodologies. Discriminant spatial filter outperforms the two other methods for both subjects and both conditions. The shown improvement was further justified statistically ($p<0.001$), using one-tailed Wilcoxon signed test.

Subject	xDAWN	CCA	DSF
S01Active	86.42%	86.00%	89.49%
S01Passive	92.82%	94.64%	96.15%
S02Active	91.48%	91.36%	94.20%
S02Passive	86.56%	86.31%	89.80%
AVERAGE	87.66%	89.58%	92.41%

Table 9: Accuracy for the Auditory Oddball paradigm dataset: Average accuracy per condition and subject. First two rows correspond to the first subject and the rest for the second.

An illustrative example of the derived spatial filters, obtained for one subject in the active recording condition is provided in Fig 22. The included scalp topographies display the contribution of each sensor in the spatial filter. All three methods tend to emphasize on activity from fronto-central brain areas, with our filter being more bilateral.

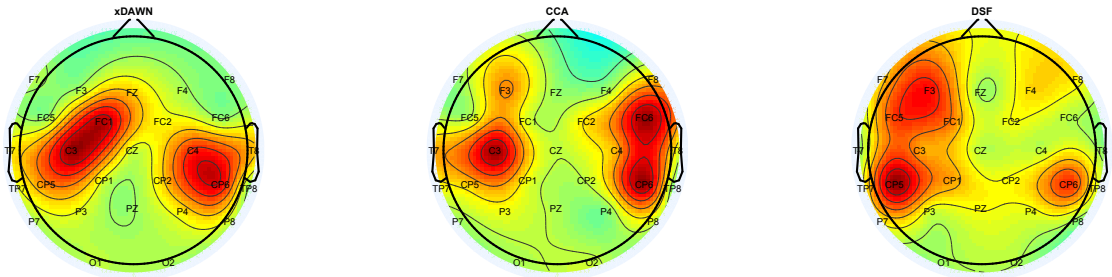


Figure 22: Topographical representation of the absolute values of the weights in the three spatial filters derived from the active condition of subject S01 in the auditory oddball dataset.

Attention Detection Task To examine the potential of the proposed method in a more challenging task, we further utilized the dataset from [64] in the context of detecting attention (i.e. to distinguish between attentive and passive responses to rare stimuli). To this end, we utilized the P300 single-trial responses from both attentive and passive condition and attempt to design a spatial filter that could enhance the separability between the two types of homologous brain responses [55]. In order to calculate the spatial filters we used segments of 0.4 seconds from stimulus onset, to isolate the response of interest, while the rest of the processing and evaluation procedures were as described before. Table 10 presents the obtained results. The increased performance of the DSF method is statistically significant ($p<0.001$; one-tail Wilcoxon signed rank test). We must note here that the xDAWN algorithm can only consider ERP against nonERP problems and therefore is not suitable for the particular task.

Figure 23, presents the averaged waveforms for the virtual sensor produced by each one of the three spatial filtering methods. It is evident that DSF trace enhances the differences in the temporal patterning of P300 response, and that the influence of attention appears as a faster risen response.

Discussion and Conclusion The main idea behind DSF is to find a linear combination of sensors that increases the SNR of single trial responses. This is achieved through the maximization of the Fisher's separability criterion. In the same context, xDAWN aims into maximizing the SSNR. Both ideas lie on a good estimation of the average ERP response, which of course

Subject	xDAWN	CCA	DSF
S01	59.53%	67.12%	71.75%
S02	55.02%	54.54%	70.64%
AVERAGE	57.28%	60.83%	71.20%

Table 10: Accuracy for the Attention Test dataset: Average accuracy for 100 repetitions of 10-fold cross validation splits per subject. The task concerns the separation of active and passive condition by the corresponding P300 waveforms.

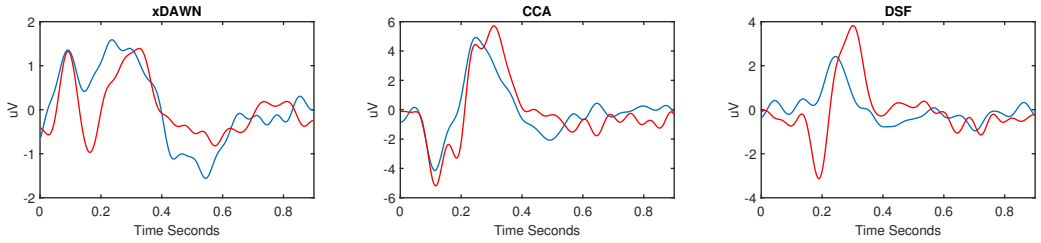


Figure 23: The average P300 waveforms produced by the three spatial filters in both active and passive conditions. The time-courses correspond to the first subject and weight vectors have been normalized based on L2 norm.

requires a large amount of single trial ERP responses cleaned from artefacts and specific assumptions to hold. However, data come in a continuous manner and therefore DSF weights need to be regularly updated. The similarities that DSF holds with Fisher's linear discriminant allow us to get inspiration from the incremental LDA algorithm [23] for overcoming such difficulties. An incremental DSF approach could also be considered as an adaptive procedure for alleviating the problem relating to slightly variable electrodes placement between different recording sessions.

DSFs, in their introduced form, implement only a linear combination of sensors and this of course acts as a limitation to information enhancement. The kernel trick seems an advantageous direction for extending this work. Generalized discriminant analysis [8] appears a readily exploitable framework for designing nonlinear DSFs. Finally, future work should address the temporal window where temporal patterning is addressed, and possibly invoke multiple multiscale variants in cases where the brain response evolves in distinct stages, diverging both in space and time.

3.2.2 Pilot Phase I trials analysis

In this section, our objective is to analyze the data collected through the phase I trials of MAMEM (cf. D6.4) during the gaze-based typing experiment with ErrP-designed keyboard. The subjects that participated correspond to 18 patient participants (6 with Parkinson's disease, 6 with neuromuscular disease and 6 with spinal cord injury), along with their able-bodied control cohorts. The subjects that did not manage to complete the designed ErrP experiment properly were excluded from this analysis. During this experiment, subjects had to type several sentences while their brain and gaze activity was recorded. The subjects stopped typing when they thought they had completed the typing correctly, since they were also allowed to press the "backspace" button. During the dwell time the gazed key was magnified and filled in with a white colour bubble (dwell animation) indicating and making clear the letter the participant was looking to. This allowed the subjects to move their gaze away when they perceived that the system was misinterpreting their gaze in order to avoid an error. The result of this experimental setting was to produce non-time-locked event related potentials associated with the error perceptions since there was no indication for such an event.

It is known that ErrPs lie within the low frequency components of brain activity. Therefore following the rationale described in the DSF section the signals where band-pass filtered between

1-16Hz. Then the signals were segmented into epochs starting from key selection, also referred to as stimulus onset, and lasting 0.5 seconds. The filtering was applied prior to segmentation in order to avoid edge effects. Since the number of instances for the correct and the erroneous responses was unequally distributed and this would lead to a classifier that would classify everything as correct we had to augment the minority class. This was performed using the SMOTE algorithm (a detailed description of SMOTE is included in the following section). The augmented epochs were then split into training and testing sets by means of Monte-Carlo cross validation. Since the DSFs seem to offer a significant improvement in the classification of the ERPs they were employed as a standard procedure in the processing of the ErrPs obtained from phase I trials. The spatial filters were obtained from the training set and were applied to both the training and testing sets. The former was used to train a linear SVM classifier while the latter to validate its performance.

Table 11 tabulates the results obtained from the classification process averaged after 5 Monte Carlo cross-validation repetitions. The performance of the employed classifier indicates the absence of, or more gently expressed as weak presence, of ErrPs.

Subject ID	Sensitivity	Specificity
PH1	60.0	39.23
PH2	92.0	08.57
PH3	100.0	01.54
PH4	12.0	83.45
PP1	76.00	31.43
PP2	100.0	00.00
PP5	60.0	47.24
NH2	72.00	84.86
NH3	100.0	00.00
NH4	88.0	03.64
NH6	36.00	86.81
NP4	80.0	87.46
SH4	84.00	02.11
SH5	96.00	04.44
SH6	36.00	65.56
SP1	100.0	04.71
SP2	44.00	74.21
SP3	84.00	22.40
SP5	100.0	00.00
SP6	100.0	00.00
Average	**76**	**32.38**

Table 11: Tabulating the average results after 5 Monte-Carlo cross validation repetitions for each subject separately. The results correspond to classification task using a linear SVM.

This fact becomes more evident by figure 24 where no phase-locked brain activity is present in the prominent electrode. Observing the grand average, across all subjects, we can see that although the classifier achieves a decent sensitivity value, the fail in terms of specificity is apparent. This means that in on-line setting, that would operate under the presented sensitivity and specificity values, the most correctly typed letters will be deleted which is expected to cause high frustration to users.

The obtained results from the phase 1 trials motivated us in order to create a new experimental setting. The main conceptual flaw that is addressed in the new setting concerns the design of a new interface that produces phase-locked brain activity related to the perception of an error. This is achieved by removing the gazing indication (dwell animation) during the letter selection. In the new setting the subjects are not aware of the letter that is being pressed

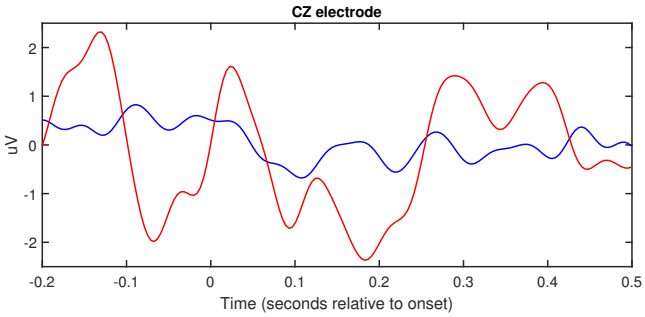


Figure 24: Brain activations accompanied with the eye movement speed for both correct, in blue, and erroneous, in red, responses. The presented refers to the average obtained from one subject.

until it is registered. This is in order to make sure that the subject realizes the error on the key registration (i.e. stimulus onset) and not arbitrarily slightly earlier. This new setting is presented in detail in the next section, in addition to the proposed multimodal methodology of combining data from the EEG and the eye-tracker to detect errors more accurately.

3.2.3 Multimodal typing error detection in gaze-based keyboards

A multimodal, also referred to as hybrid, BCI (MM-BCI or hBCI) constitutes from a BCI and another system, which might be another BCI that is based on different kind of brain patterns or a system based on other physiological signals, such as eye gaze, electrocardiography or electromyography, suitable for performing certain actions more successfully than a conventional BCI [4].

The vast majority of BCI applications makes use of electroencephalogram (EEG) that is designed to reflect the electric potential produced by the synchronous action of neurons with parallel geometric orientation [13]. However, EEG also captures electric activity from sites other than the brain. Therefore, it is difficult for EEG to be effectively combined with other modalities that involve ocular or muscular movement, such as an eye-tracking device, because the EEG signal will be contaminated by artefacts. Despite the difficulties that eye movement introduces in the EEG processing, these two modalities have been successfully combined during the last years. Actually, one of the uses concerns the identification and removal of ocular artefacts from EEG signal with the aid of an eye-tracker [59]. Additionally, EEG and eye tracking have been used to predict the relevant, among displayed, items for the user. Neural activity can be associated with the displayed items by tracking the eye movement and isolating the neural signal around the gaze fixation onsets [22]. A parallel exploit of eye-tracking and EEG concerns the study of eye-fixation-related potentials [7]

In this section, we are investigating whether EEG can be used in conjunction with an eye-tracker in order to create a high-speed gaze-based keyboard. This study unfolds in two distinct directions. Firstly, we demonstrate that a specific neurophysiological event associated with error perception is elicited during the visual mistyping perception, which is not contaminated by eye movement artefacts. Secondly, we provide evidence that this event, which is a special case of an ErrP, can serve as the basis for an automated Error Correction System (ECS) that originates directly from the users brain responses complemented by the eye movement information and is able to improve the overall typing speed in a gaze-based keyboard.

3.2.3.1 Data Acquisition Protocol

The experimental protocol relies on a gaze-based visual keyboard paradigm that was operated exclusively by an eye-tracker. Initially, the experimenter shortly described the experiment and its purpose to the participants. The objective of this experiment was to capture the brain activation and gazing coordinates of each participant while typing and identify the neural activity of unintentional letter typing. The experimenter stressed to the participants that these

experiments were conducted in order to explore their activity during erroneous actions and not to test their ability in typing tasks. For this, a challenging keyboard was used which locks faster on the letters.

The letters were selected after a dwell time (i.e. the time required for the eyes to stay focused on the letter before selected) of 0.5 seconds and then there was a preview of the selected letter by turning its colour to white for another 0.5 seconds. In order to ensure that the brain activity concerning the perception of an error was time-locked with the letter visual press, there was no dwell indication regarding the users gaze location prior to the letter selection. The participants perceived the gazed letter only after it was typed, which of course increased the number of mistyped letters but proved to be a necessity according to the experiments on the Phase I Pilot trial data presented in Section 3.1.2.

It was made clear to the participants that unintentionally typed letters should be ignored and the participant in case of a mistype had to retry in order to type the correct letter until the whole sentence is correctly completed in a letter by letter aspect. Moreover, subjects were asked to refrain their movement in order to avoid artefacts (such as jaw clenches, hand/feet movements etc.) that contaminate the EEG signal since this experiment is extremely sensitive to noise.

Twenty sentences 12 were shown iteratively to the subjects in order to be typed using the designed keyboard. During the experimental procedure, the ongoing typing sentence was not accessible so subjects had to memorize it first. This choice was made towards a more natural way of typing since subjects were able to type almost spontaneously. All sentences should be written using lowercase letters with a full stop at the end. Each session, which consisted of typing one sentence, was followed by short-time breaks. At the beginning of the experimental procedure and after the eye-tracker was calibrated in a subject specific manner so as to decrease the gaze prediction error to a bare minimum, participants performed a short session during which they typed some foo sentences in order to become familiar with the gaze-typing procedure.

	Sentences used for typing
1.	my favorite hat is blue.
2.	their dog is brown.
3.	the summer is hot.
4.	the bride looked beautiful.
5.	i can swim well.
6.	flowers smell good.
7.	the rat is in the tub.
8.	i see a big snowman.
9.	roses are red and violets are blue.
10.	my bike is red.
11.	the fox can jump.
12.	earth has one moon.
13.	the lion is up in the tree.
14.	zebras like to eat grass.
15.	seven is an odd number.
16.	big bugs are scary.
17.	i wish i had a million dollars.
18.	nature is a good journal.
19.	i have a yellow watch.
20.	this will never end.

Table 12: The twenty sentences that were used in order to obtain the ErrP dataset.

In the experiments 12 subjects participated of age ranging from 20 to 45 (mean 32 $\pm$ 4; 8 male, 4 female). All participants joined voluntarily the experiments, had no known prior or current pathological and neurological condition and their vision was normal or corrected to

normal. All participants were familiar with physical keyboard devices and therefore were aware of the letter positioning on the screen.

In order to capture the brains electrical activity the EBNeuro EEG device was used, which offers 64 wet electrodes placed according to the 10-10 international system. For the gaze information, the SMI myGaze eye-tracker was used. The sampling frequency was 256Hz and 30Hz for the EEG and eye-tracker devices respectively. The Lab Streaming Layer software performed the synchronization of the streams with a sub-millisecond accuracy.

3.2.3.2 ErrP detection system

Gazing Features The information that the eye-tracker provides concerns the location of gaze in terms of on-screen coordinates. However, coordinates can hardly provide any information regarding the users intentions and whether they managed to type the correct letter or not. Therefore, from the gazing coordinates we calculated the on-screen-distance time series the eyes travelled in both horizontal and vertical directions. The produced distance time series, constituted the basis for the classification of the gazing information.

Hjorth descriptors [27], served as the final descriptors for the aforementioned time series. These descriptors are used to indicate the statistical properties of a signal in the time domain and consist of three components. Let us denote by x(t) the initial signal, then the corresponding Hjorth descriptors are: $activity(x(t)) = var(x(t))$, $mobility(x(t)) = \sqrt{\frac{var(\frac{\partial x(t)}{\partial t})}{var(x(t))}}$ and $complexity(x(t)) = \frac{mobility(\frac{\partial x(t)}{\partial t})}{mobility(x(t))}$. These three components describe not only distance properties but also capture speed and acceleration (first and second order derivatives of distance) in a more sophisticated way.

Calibration and Classification It is known that ERPs are embedded into low frequency components, so zero-phase bandpass filtering of 1-16Hz was applied to the EEG recordings. Then the recordings, both from EEG and eye-tracker devices, were segmented into epochs that were aligned with the visual button presses. Each epoch (single trial response) contained the 0.5 seconds of the multichannel EEG signal and the gaze descriptors for the corresponding gazing coordinates signal after each visual keypress.

Since the number of erroneous visual keypresses is much lower than the number of correct ones we had to deal with a classification problem of imbalanced classes. This could lead to a classifier that classifies all observations as correct. Instead of keeping only a subset of the dominant class, a procedure often called majority subsampling, we followed the opposite tactic. From the minority class we generated new observations, referred to as minority oversampling, until the number of observations in both classes was almost equal. To perform this approach we employed the well-known Synthetic Minority Oversampling Technique (SMOTE) [12]. According to the SMOTE algorithm, in order to create synthetic observations one has to interpolate a new sample randomly between neighbours of the same class. Assuming that N new samples have to be generated, we traverse the initial set of observations and for each one we calculate its k-nearest neighbours (k is treated as a parameter). Then for each initial observation, we choose a random neighbour and between the selected neighbour and the corresponding initial observation, a new synthetic observation is interpolated. If the first traverse is not sufficient to generate N new observations, this procedure can be repeated. We must note that the SMOTE algorithm was performed on the initial data, after the frequency filtering, in case of the EEG and on the distance time series in case of eye-tracker.

In order to increase the detectability of the ErrPs and enhance their SNR we adopt the Discriminant Spatial Filter (DSF) approach, a method that increases the SNR of ERP recordings based on Fishers separability criterion. We first remove bad sensors (e.g. sensors that were bridged with ground electrode and did not record any brain activity). The remaining electrodes are used in order to calculate the weight vectors that enable the spatial filtering. Only the first component was kept and was treated as a spatial filter and so only one virtual

sensor was produced (a weighted sum of the initial sensors). For the gaze signals, each epoch was described by the Hjorth descriptors for the on-screen distances. The choice for Hjorth descriptors is based on the assumption that when the user manages to type a letter correctly his gazing will shift fast towards the next desired letter. In the opposite case, the user will slightly adjust his gazing in order to type the intended.

It is quite common that ERP components are distinguishable by linear classifiers, so for the classification task we made use of the well-known Support Vector Machines (SVMs) with linear kernel. Initially the data, after the filtering and epoching procedures, were split into train and test sets. The train set was used in order to create the augmented train set (with synthetic data). Then the augmented train set was used in order to calculate the DSF weights and the classifier was trained by the corresponding virtual signal. Figure 25 schematically represents the processing pipeline that was employed for the ErrP detection and classification. The eye-tracker features and the EEG were combined in a late aspect. That means that two separate SVMs were trained, one for each modality and their output was combined in order to form the final decision. This happened in order to avoid early fusion issues that stem from combining features with large numerical differences. The test set consists of ten erroneous responses and a specific amount of correct responses so as maintain the initial ratio of correct over incorrect visual keypresses. In order to evaluate the classifier, a monte-carlo cross validation approach was performed where the splitting and testing procedures were repeated 100 times.

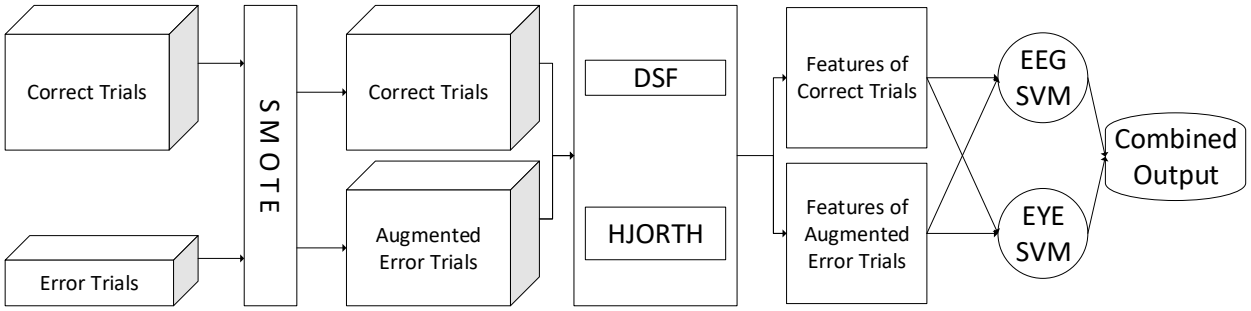


Figure 25: This figure outlines the methodology that was employed for the classification and calibration of the error-detection system.

3.2.3.3 Experiments

System Evaluation In order to quantify the performance of the BCI we employed the Utility metric, which offers a more user-centered perspective [16]. This metric is designed to match the concept of speller and also enables the gain quantification of an automated ECS. The Utility in a simple BCI system (without an integrated ECS) is calculated by $U = \frac{(2p-1)\log_2(N-1)}{c}$, while in error-aware system is calculated by $U_{ECS} = \frac{(pr_C+(1-p)r_E+p-1)\log_2(N-1)}{c}$ where N is the number of possible selections (i.e. symbols on the keyboard), p the accuracy of the system (i.e chance that the eye-tracker will interpret correctly the users gaze), C the required time in order to perform a keypress (i.e the dwell time in the visual keyboard case), r_E the sensitivity (recall) value of the erroneous class and r_C the sensitivity value for the correct class. It is of course assumed that r_E and r_C are letter independent and that p remains constant. Finally the gain of integrating an ECS in the visual keyboard application can be defined by the ratio of utility in the error-aware system over the utility of the simple, $g = \frac{pr_C+(1-p)r_E+p-1}{(2p-1)}$. When the gain value is over one then the ECS actually improves the overall performance of the BCI.

Physiological Findings Among the main concerns of this study was to investigate the neuronal responses associated with the perception of an error during the gaze-based typing procedure. We isolated the brain and eye activity after each letter preview. Figure 26 depicts the average responses of one subject for both brain activity and eye movement. The main

component of the average erroneous brain response is a negative peak 300ms with a frontocentral scalp distribution after stimulus onset (i.e. preview of the selected letter) followed by a positive peak 100 ms later with a centro-parietal scalp distribution. The presented results seem to agree with the ones met in the literature [11] differing only in latency.

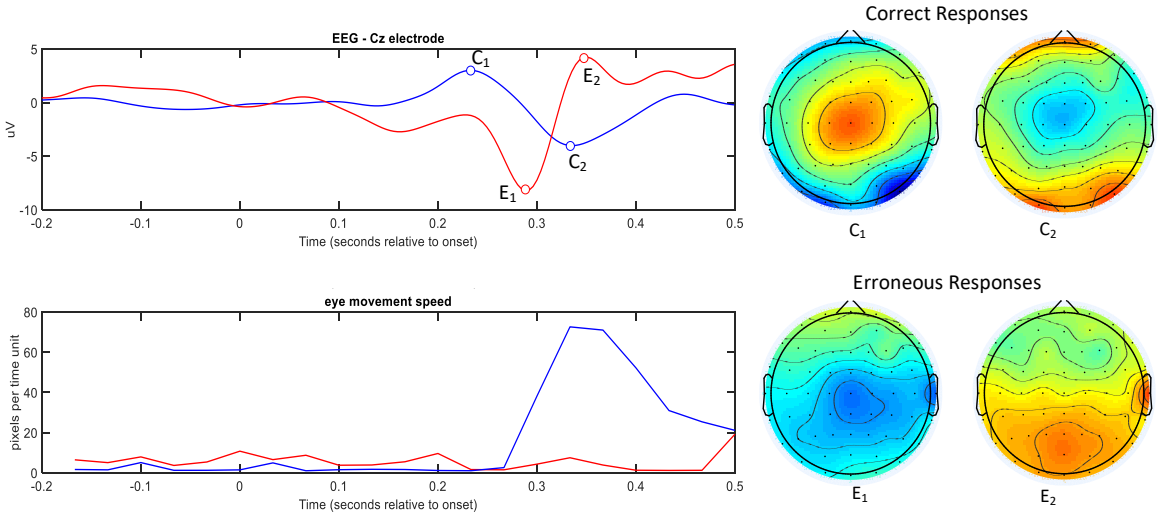


Figure 26: Brain activations accompanied with the respective scalp distributions and eye movement speed for both correct, in blue, and erroneous, in red, responses. The upper scalp potentials, C_1 and C_2 correspond to the positive and negative peak of the blue line while E_1 and E_2 to the negative and positive of the red line respectively. The presented refers to the average obtained from one subject.

On the other hand one would expect the average correct response would be flat and no signal to survive the averaging. However, we notice a negative peak at 300 ms followed by a positive peak at 400 ms relative to the stimulus onset. Both components, according to the presented scalp topographies, have a centroparietal scalp distribution. As a matter of fact, we see that these potentials come in conjunction with significant eye movement that accompanies only the correct responses and the corresponding peaks of eye speed and brain activity are aligned in time. Actually after an erroneous response the subjects slightly adjust their gaze since their intention is marginally misinterpreted while in the opposite occasion one has to type the next letter which is probably located far on the screen from the previous (however two nearby consecutive correct letters are not an impossible scenario e.g. in typing the word "was"). The corresponding scalp distributions indicate that recorded brain activity is not the outcome of the eye-movement artefacts but rather from the brain region that seems to be directly related to the eye movement [49].

To further investigate the association of EEG and the corresponding eye movement we grouped similar eye movements and computed the respective average brain waveforms. For each eye movement epoch we calculated the total movement and we performed clustering using the k-means clustering. Hoping that it will manage to group up, down, left and right eye movement we set the desired number of clusters to 4. Figure 27 shows the grouping performed by k-mean in the eye movement of correct responses for S02 and the corresponding average EEG waveforms. As a matter of fact, the k-means algorithm managed to performed a meaningful, in terms of directions of the eye movement, grouping. The respective average EEG waveforms seem to follow to eye movement speed without any polarity alternation, as it would be expected for ocular artefacts [40], among opposing movements (e.g. left, coloured as red, versus right, coloured as blue).

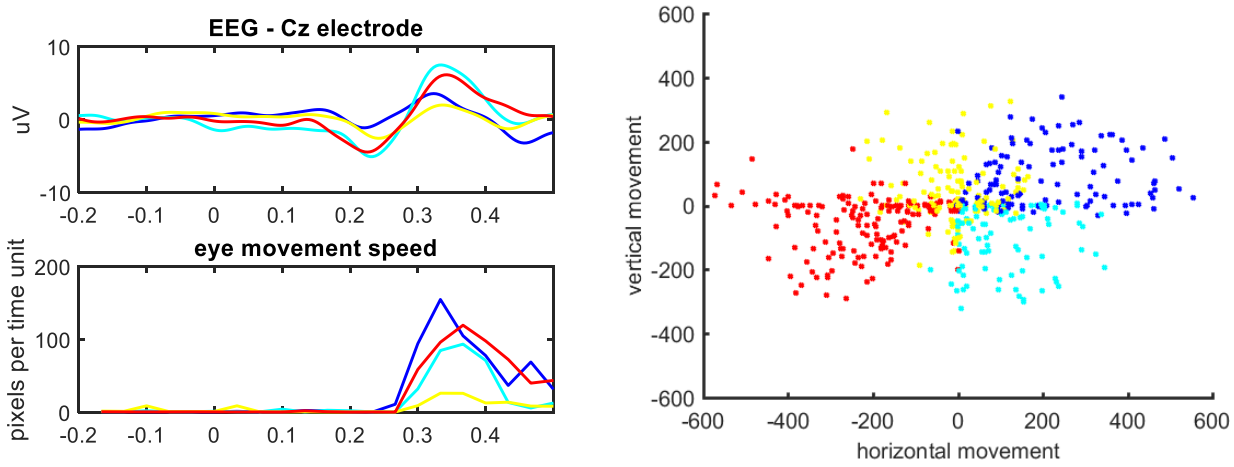


Figure 27: Average brain activations grouped according to the respective eye movement in each epoch using the k-means algorithm obtained from one subject. The origin point is relativeto the starting gazing point of each epoch.

Classification of Errors A crucial step in our research concerns the discrimination of correct from erroneous letter types using both EEG and eye movement signals independently or combined. For this task we employed the Support Vector Machines with linear kernel. The results tabulated in Table 13 were obtained after 100 repetitions of Monte-Carlo cross validation avoid any bias associated with the randomized partitions. Since the classes are imbalanced accuracy cannot serve as a reliable measure of evaluation. Therefore we report sensitivity, the proportion of errors that are correctly identified as such, and specificity, the proportion of corrects that are correctly identified as such. As Table 13 indicates, EEG based classification tends to emphasize sensitivity while eye movement specificity. Classification that is based on a mixture of features seems to achieve equal results in both terms of sensitivity and specificity without any significant trade off over the other. However deciding whether sensitivity or specificity is more crucial in an error aware requires more advanced metrics that are task-oriented. Figure 28 depicts the proposed BCI system that realizes an error aware visual keyboard.

	EEG		eye		early fusion		late fusion	
Subject ID	Sens.	Spec.	Sens.	Spec.	Sens.	Spec.	Sens.	Spec.
S01	73.4	83.197	97.8	51.395	78.9	85.961	91.5	78.000
S02	75	87.974	90.1	71.948	80.4	89.345	85.5	87.009
S03	85.5	93.204	89.3	82.830	87.4	94.000	89.5	93.136
S04	74.7	85.184	87	70.263	80.1	85.184	83.8	81.921
S05	76.3	85.903	88	75.718	82.5	89.835	85.2	89.961
S06	76.6	80.095	95.6	63.952	88.9	75.825	92.6	73.698
S07	61.6	78.797	83.9	67.891	71.7	77.797	79.3	75.578
S10	85	83.234	90.7	50.702	84.9	85.851	89.1	76.766
S11	69.3	83.765	94.2	71.357	89.8	85.306	90.2	83.765
S12	73.9	80.941	75.5	77.012	75.3	81.188	76.9	81.859
S13	67.3	82.362	87.7	77.184	78.6	83.941	82.7	83.704
S15	70.4	81.419	80.7	80.565	79.3	80.806	79.7	82.694
Average	**74.08**	**83.84**	**88.38**	**70.07**	**81.48**	**84.59**	**85.50**	**82.34**

Table 13: Tabulating the average results after 100 Monte-Carlo cross validation repetitions for each subject separately. Four classification scenarios are presented using only EEG recordings, eye movement information and combined in both early and late aspects.

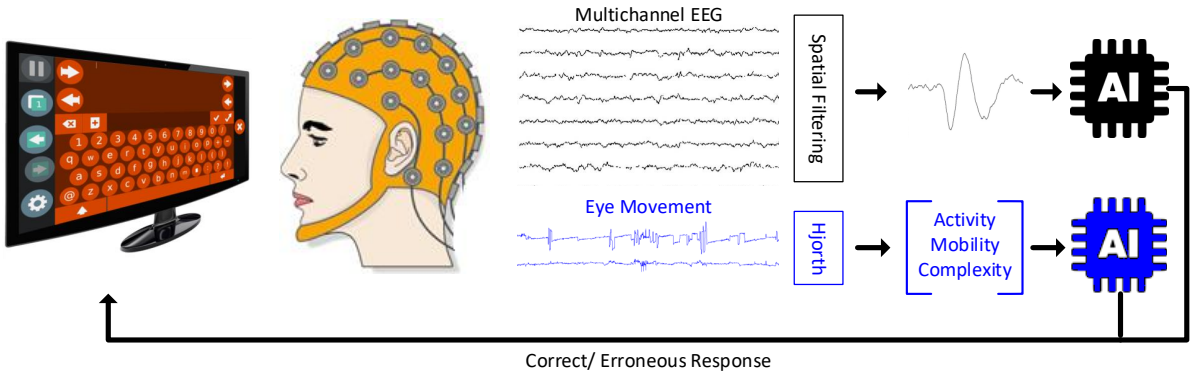
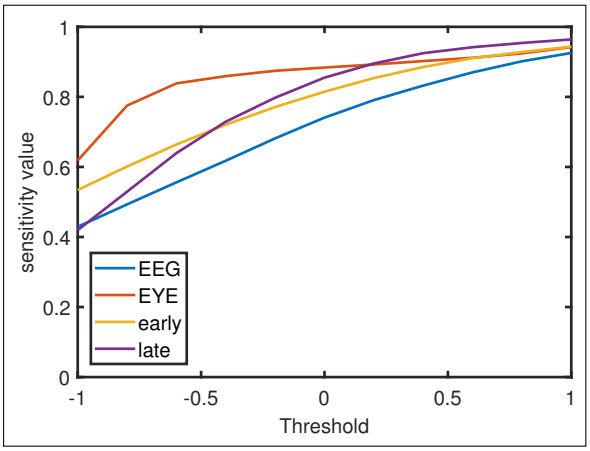


Figure 28: This figure outlines the methodology that was employed for the classification and calibration of the error-detection system.

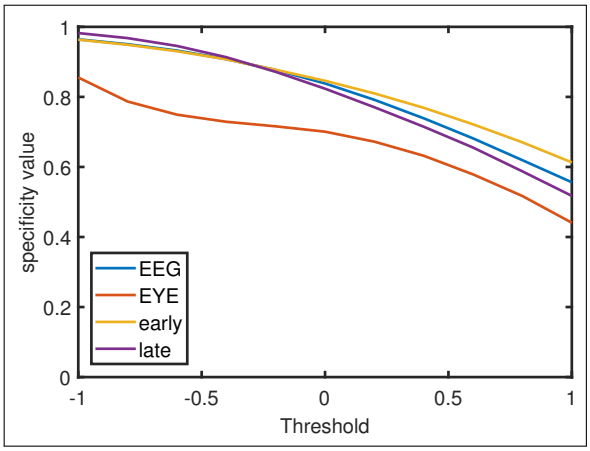
Exploitation in the visual Keyboard Classification of erroneous responses is a crucial task in order to create an error-aware keyboard and therefore accuracy alone cannot provide the necessary information for evaluation, especially in a problem with unequally distributed samples over classes. The Utility metric offers a natural way to decide sensitivity over specificity and ascribe the corresponding importance weights. Then, the utility metric itself captures the gain from detecting and auto-correcting the mistypes in terms of average time required in order to complete a correct letter type.

An SVM classifier calculates a hyperplane and in order to proceed with the classification task it computes the sign of the projection for each observation from the aforementioned hyperplane. Interpreting arbitrarily the SVM classifier, the hyperplane is computed in order to achieve maximum classification accuracy. As already stated, this measurement is not sufficient for evaluating the classifier for the desired task. Therefore we sought for a threshold (i.e. a value that is added in the calculated normalized projection of observations to the hyperplane and can be thought of as moving the hyperplane towards one class or another) that maximizes the utility metric. According to the utility metric the most crucial parameters for detecting and auto-correcting erroneous responses are sensitivity and specificity modulated by the chance the eye-tracker will interpret the user's intentions falsely. Having that in mind, specificity is far more important than sensitivity.

It becomes apparent from figure 29 that specificity is maximized for negative values of threshold. Among these values, the one provided by late fusion classifier manages to achieve maximum average specificity. As a matter of fact, it is also the maximum gain achieved by all compared classification methods. Table 14 contains the results obtained at the threshold that achieves the maximum utility gain in both terms of early and late feature fusion, which happens to coincide. As table 14 indicates the late feature fusion classifier is the one that systematically produces better results compared to early and single-modal gains. A fact that also needs to be noted here concerns the results of EEG and eye movement classifiers. As a matter of fact, EEG almost in all cases is able to enhance the user's experience in terms of time efficiency contrary to the eye movement information. Finally, we must note that all obtained accuracies are above random chance.



(a) Sensitivity with respect to threshold.



(b) Specificity with respect to threshold.

Figure 29: The grand average sensitivity and specificity values, after 100 Monte-Carlo cross validation repetitions, with respect to threshold moving within the normalized SVM margins.

Subject ID	misstype probability	Gain EEG	Gain eye	Gain early	Gain late	Spec.	Sens.	Acc.
S01	11.60	1.02	0.69	1.05	1.05	98.64	45.60	92.48
S02	7.92	1.01	0.93	1.02	1.03	98.55	45.40	94.33
S03	6.36	1.04	0.99	1.04	1.04	99.28	71.00	97.48
S04	11.6	1.04	0.93	1.04	1.05	98.53	46.50	92.48
S05	8.87	1.02	0.91	1.04	1.04	98.94	50.60	94.66
S06	13.67	1.03	0.92	1.03	1.03	96.68	39.20	88.81
S07	13.50	0.99	0.92	0.99	1.02	97.97	24.10	87.99
S10	17.50	1.10	0.93	1.12	1.12	96.87	59.50	90.32
S11	9.24	1.00	0.94	1.03	1.02	98.89	28.70	92.39
S12	10.53	0.99	0.96	0.99	1.01	97.84	26.60	90.34
S13	6.16	0.98	0.91	0.99	1.00	97.66	35.40	93.82
S15	13.84	1.03	1.01	1.05	1.05	98.84	31.20	89.44
Average	**10.90**	**1.02**	**0.92**	**1.03**	**1.04**	**98.22**	**41.98**	**92.05**

Table 14: Chance that the eye-tracker will interpret user's intentions falsely, utility gain using EEG, eye movement as well as early and late feature fusion by moving the separation hyperplane so as to achieve maximum gain. Recall values for error and correct class and accuracy correspond at the respective classification threshold. The results correspond to the average of 100 Monte-Carlo cross validation repetitions.

4 GSR-based interaction and analysis

In this section, we present the algorithm adopted by MAMEM for detecting the stress level of a user based on GSR recordings. The complete pipeline contains various preprocessing steps for cleaning the GSR signals, the automatic training of the stress thresholds per user and the detection of stress levels in continuous recordings. More specifically, the preprocessing steps include the downsampling, filtering and interpolation techniques applied to the raw signal. For the automatic training algorithm, we have relied on the algorithm presented in D3.2 that detects the stress level thresholds in an unsupervised manner for each subject independently. Finally, in order to showcase the application of the proposed pipeline we present in the following a step-by-step analysis of the GSR signals obtained through the Phase I Trials of MAMEM. The following results correspond to one subject and serve as an example. The outcome of the analysis for all subjects has been included in D6.4.

The GSR data were collected during every stage of the Phase I experiments and were stored

in a separate file. For the purpose of our analysis we excluded data that were recorded during the lightweight configuration since we did not expect them to offer additional insights. Based on the stress detection algorithm that was described in D3.2, we concatenated the GSR data that were recorded during the stages of a) GTW training, b) ErrP experiment, c) SMR experiment and d) Dictated Tasks and used the whole duration for calibrating the algorithm. Specifically, from each file generated for a partipant namely a) *Organization_ParticipantID*_GTW b) *Organization_ParticipantID*_Heavy_ErrP c) *Organization_ParticipantID*_Heavy_SMR and d) *OrganizationParticipantID*_Dictated we extracted the GSR signals and concatenated them based on the aforementioned order. The concatenated signal was downsampled to 32Hz and cleaned by passing it through a low pass zero-phased FIR filter with a cut-off frequency of 5Hz, as well as by repairing segments of the signals that were interrupted by contact loss, as it has been described more extensively in D2.3. An example of a cleaned signal can be seen in Figure 30. Note here that the y-axis corresponds to the skin resistance (kOhm), thus higher skin resistance corresponds to less sweat and as a result to less stress.

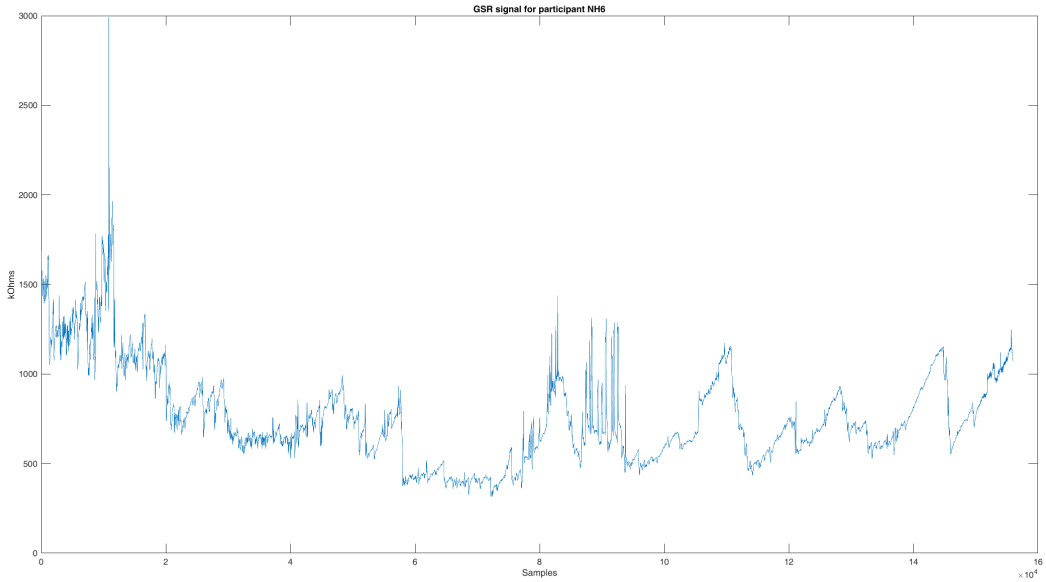


Figure 30: GSR signal for participant NH6 after cleaning

The resulting cleaned GSR signal was then analyzed by the stress detection algorithm described in D3.1 resulting in the definition of the stress level thresholds, which can be seen in Figure 31. In this figure, 4 coloured lines categorize the segments of the signal into 5 stress levels starting from level 1 (no stress) up to level 5 (very stressed). More specifically, the area above the green line determines the no stressed segments (level 1) and below each of the coloured lines the stress level is augmented by 1 up to level 5, which corresponds to the parts below the red line.

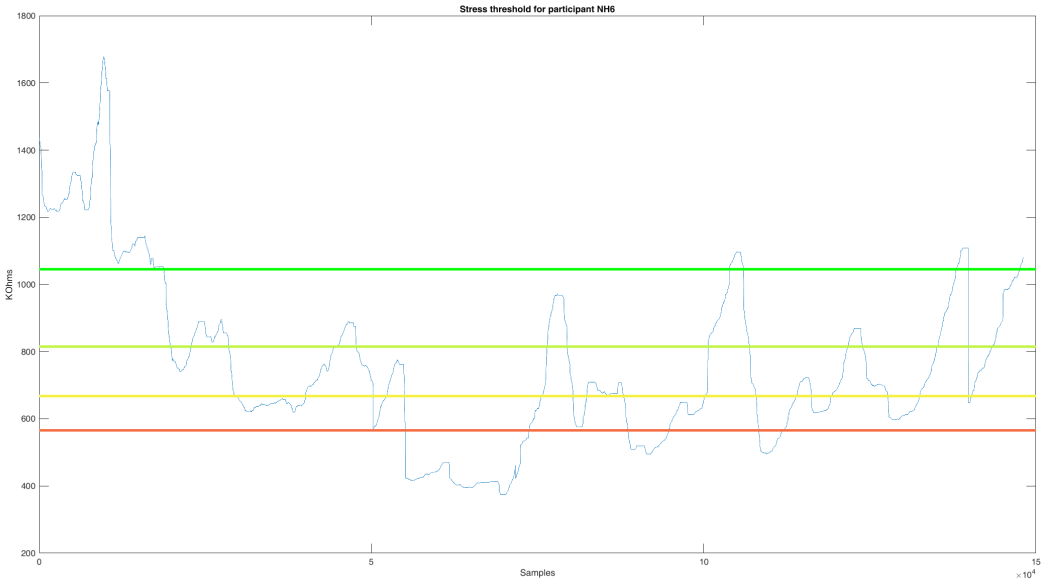


Figure 31: Stress indicator levels and thresholds (coloured lines)

The goal of this analysis was to compare either the patients to their corresponding control participants or between different cohorts with respect to the related sub-tasks they had to perform so as to generate any meaningful insights. In order to achieve this, we had to aggregate the data based on the average stress that was generated between each sub-task. These sub-tasks was selected based on the event data that were recorded and thanks to LSL timestamping we were able to synchronize them with the GSR data. Specifically, the sub-tasks that were defined for the analysis were the following:

- For the GazeTheWeb training task, the duration between the beginning and the completion of each training level, e.g. Basic 1, Basic 2, Intermediate 1, etc.
- For the ErrP experiment, the duration between the start and the end of a new sentence, as well as the two resting periods if available for the participant.
- For the SMR experiment, the duration between the start and the end of each of the 8 sessions
- For the Dictated task, the duration between the beginning and completion of each task, e.g. E-mail, Photo edit, Youtube and Twitter.

The result of this analysis was a mean stress value for each of the 27 subtasks and for each participant. This can be seen for example on Figure 32, where we aggregated data between all healthy and all patient participants. In this figure, each data point represents the average stress for all healthy participants (blue line) and patient participants (red line) during a specific task. Based on the presented algorithmic procedure, the stress-related comparison figures found in deliverable D6.4 have been generated.

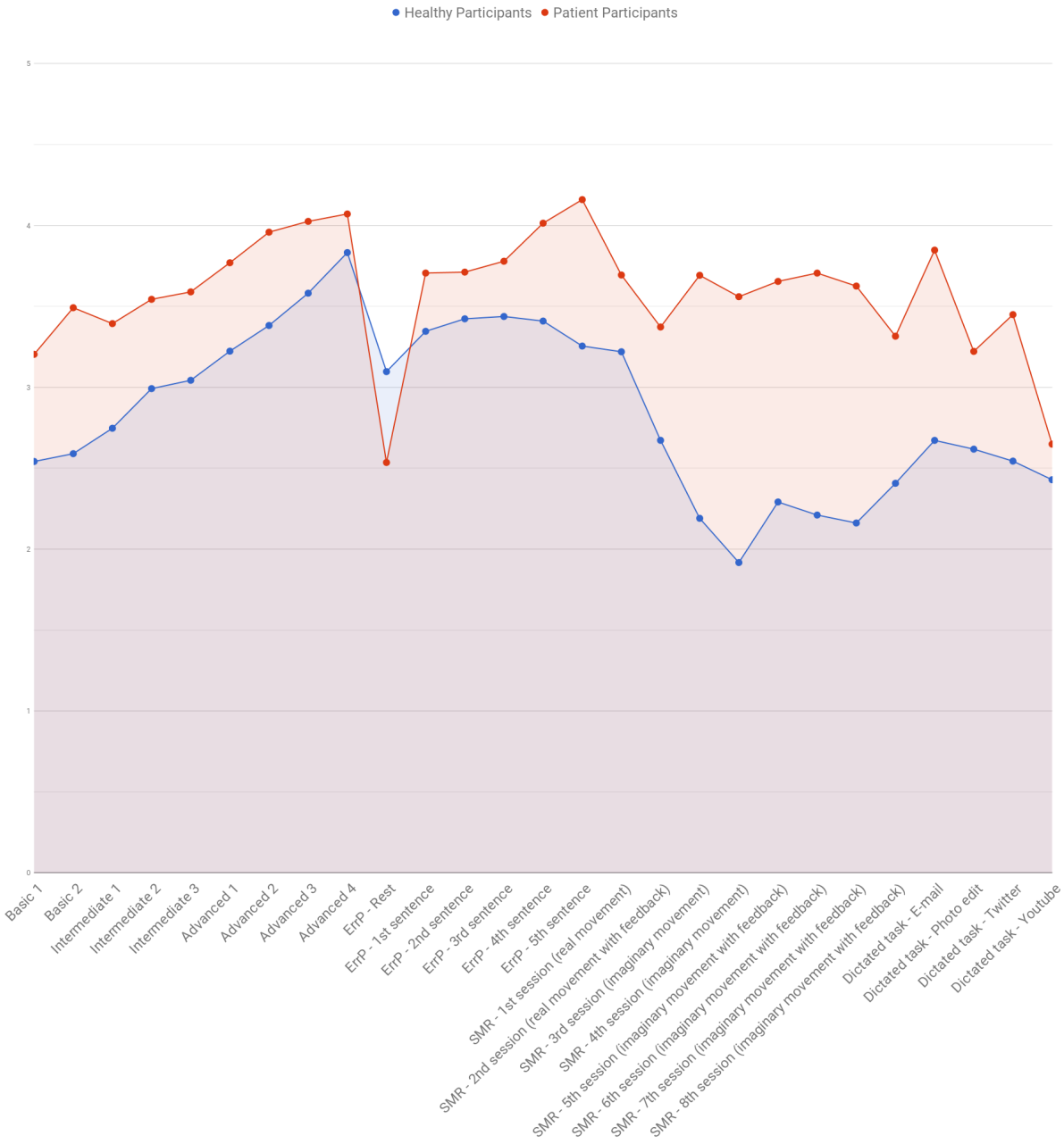


Figure 32: Average stress levels across healthy (blue) vs patient (red) participants, aggregated for each task.

5 Multimodal interaction

Having defined the necessary components and algorithms of the MAMEM system, in this section we propose the multimodal interaction paradigms that will be available to the end users of the MAMEM technology. More specifically, the rationale behind the MAMEM system is to rely on the more advanced gaze-based interaction through GazeTheWeb and complement it with EEG and GSR-based functionalities in order to alleviate its shortcomings. GazeTheWeb will be used as the interacting interface between the end users and the web and will encapsulate three multimodal interaction paradigms; a) the error aware keyboard, b) the reading and selection mode switch and c) the MM-Tetris game.

5.1 Error-aware keyboard

In this paradigm, the standard keyboard of GazeTheWeb will be replaced with the ErrP-designed keyboard that does not have a dwell animation (Section 3.2.3) in order to elicit phase-locked ErrPs. The errors will be detected based on the combination of EEG (i.e. ErrPs) and gaze-based features (Section 3.2.3). Once an error is detected, the multimodal keyboard will automatically delete the erroneous letter. In this way the typing speed of the user is expected to increase significantly.

5.2 MM-TETRIS

The proposed adaption of the popular Tetris game incorporates the utilization of all the three sensors (eye-tracker, EEG and GSR). The objective of this game is twofold; first to train the user to better control their SMR signals in a gamified environment and second to provide to the end users an interesting game to play in line with the creative pack that was planned in D8.3. The basic design principles have been presented in Section 3.1.3, while the main challenge that we focused to overcome was the transformation of the typical cue-based SMR detection algorithms in self-paced ones. The exact design elements of MM-Tetris will be presented in the upcoming updated version of D5.3, that will release the final MAMEM multimodal interfaces. It is worth noting though, that this game incorporates all the three sensors, i.e. eye-tracking, EEG and GSR as active input controllers of the game, i.e. horizontal move, rotation and speed respectively.

5.3 GazeTheWeb mode switch

Once the user is trained to control their SMR signals with sufficient accuracy, the asynchronous self-paced mode switch through SMR will be enabled. The user will be able to change between the two designed modes (i.e. reading and selection mode) by performing an imaginary movement. The reading mode will deactivate the click-able objects within the viewed web-page so that the user can read without activating every link they look at (Midas Touch problem). On the other hand, when GazeTheWeb is at selection mode all links are activated and the user can click them by fixating on them.

6 Conclusions

Deliverable 3.1 has proposed the initial direction that the individual modalities will be investigated and implemented in unison to bring forth a Multimodal Framework for MAMEM. D3.2 analyzed these modalities in details to try to bring them together in GazeTheWeb concept. In this deliverable D3.3 we presented the final implementations and analysis of algorithms that can enhance the Web accessibility of motor impaired people. We described the method and implementation of GazeTheWeb that follows the optimization guidelines by adapting the interaction in Web pages for gaze-control. This primary mode of Web interaction by gaze-control is effectively supported by additional modalities. For example, erroneous clicks are expected due to the known shortcomings of gaze based interaction (e.g. Midas Touch problem). We enhance the gaze-based interaction through the incorporation of brain signals, transforming in this way GazeTheWeb in a multi-modal interface that alleviates these shortcomings. Towards this goal and considering that typing is one of the most intensive interactions, an error-aware keyboard that automatically corrects the gaze-based typing errors was presented. Another multi-modal concept is demonstrated through MM-Tetris game, where we have utilized all three sensors (eye-tracking for controlling tetrimino horizontal movements, EEG through SMR for rotating the tetriminos and GSR through stress detection for adjusting tetrimino movement speed). We also presented the use case of switching between reading and selection mode of GazeTheWeb by performing an imaginary movement through SMR.

The algorithms and prototypes presented in this deliverable forms the core of MAMEM application framework, i.e., to be used in phase II trials, and future exploitations.

A Documentation for the EEG & GSR

The software accompanying this deliverable for the EEG and GSR data analysis is available on github[14]. It is a toolbox in Matlab and the extensive documentation for it can be found in D3.2. In this deliverable, we provide documentation only for the new elements that were presented previously, i.e. the Hjorth descriptors, the SMOTE algorithm, the proposed spatial filters (DSF), the Utility metric and the algorithm for fusing the EEG and eye-tracking data for error detection.

A.1 Hjorth Descriptors

Listing 1: Hjorth Descriptors

```
Produces Hjorth's Descriptors, Activity, Mobility and Complexity,
for each trial and each sensor.

Input:

- Trial_Matrix: a matrix s x t, where s = # sensors,
t = # samples containing the EEG samples for all sensors
for a trial

Output:

- Hjorth_desc: a s x 3 row vector where each row describes
Activity, Mobility and Complexity for a sensor
```

[14] https://github.com/MAMEM/eeg-processing-toolbox

A.2 SMOTE

Listing 2: SMOTE

```
Produces new data samples from a given set of instances
according to the Synthetic Minority Oversampling Technique.

Input:

- data: n x p data matrix where n = number of instances
and p = dimensionality of the feature space

- N: the number of synthetic samples that correspond to each
initial instance

- k: the number of neighbors used for linear interpolation.

Output:

- new_samples: n*N x p data matrix where n*N = number of
augmented instances and p = dimensionality of the feature space
```

A.3 Digital Spatial Filters

Listing 3: spatialSNR

```
Calculates a projection matrix A that maximizes the Fisher's
score (equivalent to SNR) for binary classification problems
only.

Input:

-trials: n x c x t data-matrix where n = number of trials,
c = number of sensors and t number of samples containing the EEG
data.

-labels: vector containing the label for each trial (binary).

Output:

-A: the projection matrix c x c. The first column provides the
linear combination that maximizes the SNR.
```

A.4 Utility metric

Listing 4: utility_metric

```
Calculates the utility metric.

Input:

-N: number of possible options e.g. number of available
letters to press
-p: probability for the initial system to make a mistake
-rc: recall value of correct class
-re: recall value of error class
-c: time needed in order to perform one trial, e.g. type
one letter

Output:

-U: utility value, scalar
```

A.5 Error detection fusing EEG and eye-tracking data

In this section we provide the documentation for the analysis performed on both eye-tracker and EEG signals by means of error detection.

Listing 5: utility_metric

```
Calculates the detection rate of erroneous actions during a
visual typing task based on EEG signals and eye-tracking
information from synchronized recordings. Initially the streams
are segmented into trials and are labeled. The minority
observations are augmented using SMOTE algorithm. Then
the Hjorth descriptors and the spatial filters are calculated
for the eye-tracking and EEG respectively. Finally two SVMs are
are trained for classifying separately the two modalities that
are late fused so as to enable the multimodal error detection.
The whole procedure is performed in terms of monte-carlo cross
validation 100 times and the average detection rate is
calculated.

Input:

-eeg_data: a struct that should contain at least 3 fields namely
timeseries, timestamps and info.effective_srate concerning the
eeg data
 --eeg_data.timeseries: a c x t matrix containing the eeg values
where c the number of sensors and t the number of time samples.
 --eeg_data.timestamps: an 1 x t vector containing the
 timestamps for each eeg sample
 --eeg_data.info.effective_srate: the sampling rate for eeg
acquisition

-eye_tracker_data: a struct that should contain at least 3
fields  namely timeseries, timestamps and info.effective_srate
concerning the eye-tracking data
 --eye_tracker_data.timeseries: a 2 x t matrix containing the
eye-tracking values for horizontal and vertical axis, t is the
number of time samples.
 --eye_tracker_data.timestamps: an 1 x t vector containing the
timestamps for each eye-tracking sample
 --eye_tracker_data.info.effective_srate: the sampling rate for
 eye tracker data acquisition

-gtw_data: a struct that should contain at least 2 fields namely
timeseries and timestamps regarding the typed letters.
 --gtw_data.timeseries: an 1 x n cell array containing the
events corresponding to n typed letters.
 --gtw_data.timestamps: an 1 x n vector containing the
timestamps for each of the n typed letters.
-sentences: a string indicating the correct sentence that should
be typed.

Output:

-avg_acc: an 1 x 11 array containing the accuracy for error
detection for varying SVM thresholds, that is from -1 to 1
by a step of 0.2
```

B Documentation for GazeTheWeb browser

In this section we provide the documentation for GazeTheWeb browser framework (the software available on MAMEM GitHub[15]). In GazeTheWeb, we are using C++ programming language and the OpenGL API as it works on many platforms and supports high performance execution. The proposed framework is divided into two major components (Figure 33), *Visual Browser* which manages the window; creates, updates and draws the gaze-controlled interface; connects with eye trackers; handles tabs; and stores user data like history and bookmarks. The other component *CEF Implementation*, is an implementation of the Chromium Embedded Framework (CEF) that is used for webpage rendering and interaction while supporting modern HTML standards. The Visual Browser and CEF Implementation communicate with each other over a single *Mediator* class that passes the rendered webpages and DOM[16] element data from CEF to the Visual Browser, on the other way it transfers the user commands (e.g. loading a new URL, emulating a mouse click etc.) from the Visual Browser to CEF. Visual Browser and the core structures of CEF run in a single process, while CEF internally handles the execution of sub-processes.

B.1 Visual Browser

Visual Browser contains the classes that endeavor the displayed user interface, and their organization is tightly coupled to the menu structures of the application. The *Master* owns a single *Web* instance, which holds all open *Tabs*. A Tab can execute different *Pipelines* of *Actions* on the represented webpage, as it also controls the Web view rendering. The Master is listening to the connected eye tracking device and updates the interfaces and interactions by sharing the gaze data with delegated the classes.

Rendering

In GazeTheWeb, rendering of the user interface is implemented by either the direct OpenGL API calls, or by encompassing eyeGUI library (described in the next section). Calls to OpenGL API enable low-level control over graphics while running on all the target platforms, which provides a high performance rendering mechanism for user interaction. For example, the pixels of the rendered webpage are received from CEF via an `onPaint` callback and are stored in an OpenGL texture object. This texture is drawn onto a quad that represents the Web view in the center of the interface. The shader program used for this texture rendering can be manipulated in order to scale and move the texture on the quad. We used this feature for the continuous zooming implementation at click emulation. Especially, since this zooming effect does not depend on any webpage magnifying on CEF side, a smooth rendering is ensured.

eyeGUI

We proposed the eyeGUI library in MAMEM deliverable D3.2 for eye based interfaces to handle gaze-controlled elements, which is specifically relevant for GazeTheWeb framework. eyeGUI supports rendering and interaction with buttons, keyboard layouts, and elementary text editing. It represents a single header file which serves as an interface for all programming side commands like adding layouts, dynamically setting of content in text blocks or registering a callback class to react at button activation. eyeGUI layouts are defined in XML files, where elements are assigned to unique string IDs that are stored at parsing of layouts and then used to connect logic on the programming side with elements in the layout.

[15] `https://github.com/MAMEM/GazeTheWeb/tree/master/Browse`
[16] `https://www.w3schools.com/js/js_htmldom.asp`

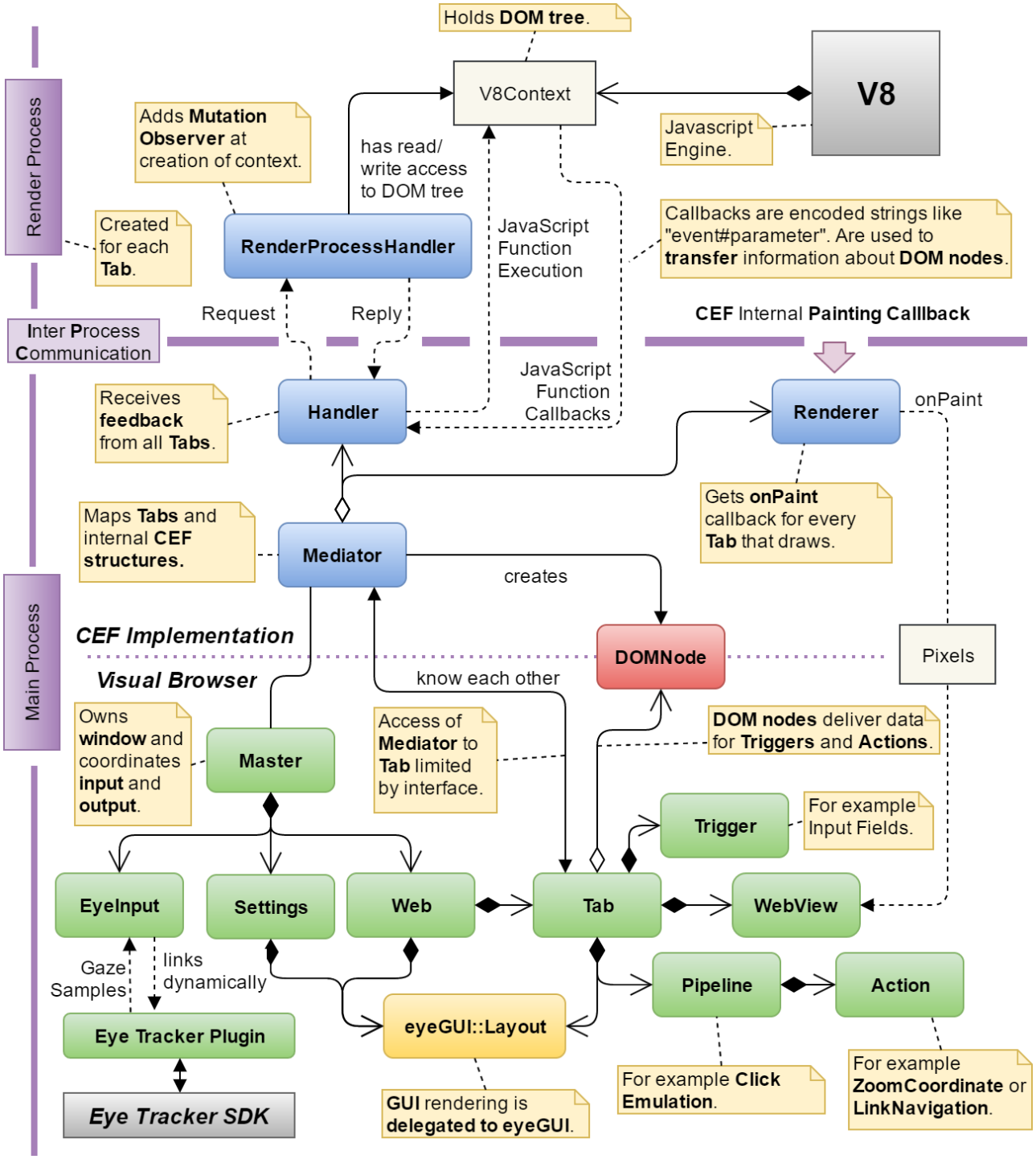


Figure 33: Architecture of the framework. The complex structure of CEF is hidden behind the Mediator in order to simplify interaction implementation.

Pipeline System

We have implied different actions for gaze driven events that are applicable in various interaction scenarios, e.g., the continuous zooming is suitable for both click emulation and text selection. Hence, we propose a pipeline system that consists of independent modules, called actions. Here are some examples of implemented actions:

- **Keyboard** inclusive text display with simple editing features. It optionally takes an initial text as input and outputs the collected text consisting of input and typed text. It successfully finishes if the user either hits the "ok" or the "submit" button.
- **ZoomCoordinate** holds the behavior described for continuous zooming in click emulation. Output is a coordinate on the webpage. Actions have limited control over Web view rendering, so this action does highlight the links while activation. This supports the selection process of the user by providing a visual feedback about clickable content.
- **LinkNavigation** takes a coordinate on the webpage and checks, whether the coordinate is upon a link. If false, the distance to the extracted DOM links is measured. When one's distance is lower than a threshold, the coordinate is moved to the center of this link.. Finally a left mouse button click is emulated at the coordinate.

The proposed pipeline is a linear combination of one or more of these above-mentioned actions. When a pipeline is enabled, the first action in the pipeline is activated and executed, and once the action is finished and deactivated, the next action get activated and so forth. Inherently the pipeline can be considered as a data stream that collects input data from the user while executing the desired action. For example the click emulation consists of first the ZoomCoordinate and second the LinkNavigation action.

DOM Node Driven Interface

The CEF Implementation serves the Visual Browser with extracted DOM nodes, which are used to enable and improve eye-based interaction in various ways:

- **Text Input Fields** are detected in order to place a gaze-controlled button on top of them.
- **Links** are used to highlight their bounding boxes while click emulation executes continuous zooming and to correct click coordinates which are not on but nearby a link.
- **Fixed Elements** are used in automatic scrolling mode. They are applied to display static page elements like navigation bars. If a user fixates these, any global scrolling of the page would represent incorrect interaction.
- **Overflowing Elements** are often necessary to scroll in order to reveal the complete information. We detect these elements and employ an automatic scrolling approach that centers the content beneath the gaze within the bounding box of the element.

There is a huge potential for further usage scenarios of extracted DOM nodes, i.e., the general availability of DOM nodes within the same programming unit is not only relevant for eye gaze input, but it provides a possibility of in-depth integration of next generation input devices (e.g., brain computer interfaces) in Web interaction.

Eye Tracker Integration

There are various eye tracking devices available in the market. In GazeTheWeb, we have implemented both SMI iViewX SDK and Tobii EyeX SDK interfaces. Further models and brands can be easily added, as long as they provide a C++ programming interface. Eye tracker SDKs are similar in the manner they provide the gaze samples. They offer the implementation of a callback function, in which they provide the gaze sample as input parameter. After connection with the eye tracker, this function is automatically called when a new gaze sample is available. Connection and disconnection are also similar in the different SDKs, however it is difficult to define a common abstraction for calibration. Tobii offers no custom calibration feature[17] and SMI is limited in terms of calibration sample positioning.

We implemented the eye tracker connections as dynamic libraries with an abstract interface for connection, disconnection and gaze retrieval. The libraries are linked at run-time, and hence the execution on systems without an installed eye tracker software is possible (no direct linking of the framework to a SDK is performed). When the dynamic linking fails, the library is not loaded and another one can be tried. Therefore we can provide a single binary for systems with no eye tracker, SMI or Tobii system. Both the SDKs are currently available only for Windows, hence there is no eye tracker support on other operating systems.

B.2 CEF Implementation

The CEF implementation provides functions to interact with a webpage and provides a list of DOM nodes. These are received by the Tab class of the Visual Browser and used to display overlays on webpage elements, i.e., representation of text input fields, to improve link navigation, or to prohibit automatic scrolling when the user's gaze is on a fixed DOM element.

Chromium Embedded Framework

The Chromium Embedded Framework is based on the Chromium project, which is the foundation for the popular Google Chrome[18] browser. CEF works on Windows, macOS and Linux, that makes it a suitable choice as host of Web technologies on desktop systems. With CEF, the GazeTheWeb application can control resource loading, navigation, context menus and more, while taking advantage of the same performance and HTML5 technologies available in the Google Chrome Web browser. CEF provides C and C++ Chromium implementations and additional interfaces, which can be easily extended without deep knowledge about Chromium's core application. CEF encapsulates the following main components:

- **Chromium** provides general functionality that enables a modern Web browser, like communication between different processes.
- **Blink** is the utilized HTML rendering engine.
- **V8** is a high performance JavaScript engine.

These components are running on different processes, which are mainly divided into the *Main Process* and multiple *Render Processes*. The Main Process contains the execution of the Visual Browser and the Mediator, including its delegated classes. Additionally one Render Process containing an instance of V8 and Blink engine is started for each tab and controlled via the implementation of the *RenderProcessHandler*. Communication between the single Main Process and multiple Render Processes is handled via *inter-process communication* (IPC), a mechanism provided by the Chromium project.

[17] `http://developer.tobii.com/community/forums/topic/eyex-custom-calibration`
[18] `https://www.google.de/chrome/browser/desktop`

Communication of C++ and JavaScript

There are multiple channels for communication between an open webpage including its DOM tree and the framework:

- **Input Emulation** is directly executed in the Handler on the Main Process. This enables emulation of standard input devices like mouse or keyboard.
- **IPC Messages** are used to request DOM data extraction in the Render Process and to transport extracted data to the Main Process. Requests are formulated by the Handler on the Main Process and asynchronously answered by the RenderProcessHandler on the Render Process. The RenderProcessHandler can retrieve the requested data by accessing the webpage's V8 context, which holds its DOM tree.
- **JavaScript** can be executed asynchronously by injecting JavaScript code from either processes. An example of this is the insertion of typed text to input fields. The JavaScript code might respond with a callback over IPC into a predefined C++ function. Such a response is encoded into a string, where the single parts are split by a separator symbol. The received string has to be parsed on C++ side in order to extract the transferred information.
- **Renderer** implementation receives the pixels of a rendered webpage on the Main Process. These are filled into an OpenGL texture and displayed as the Web view.

One of the major requirements of GazeTheWeb is to extract the DOM nodes from the displayed webpages in order to improve eye-based interaction. In the real time Web application, it is critical when to fetch DOM related data from JavaScript to keep an updated overlay and interactions. A straight forward approach would be to use the `OnLoadEnd` method in the Handler, which is triggered when a page has completed loading. The DOM tree could be parsed at this point of execution and nodes of interest can be extracted. However, no DOM nodes are available before this point and this approach does not account for on-going changes in the DOM tree while browsing. Therefore, we imply a mechanism to induce C++ callbacks when JavaScript code perceives changes in the DOM tree. This is realized with direct JavaScript callbacks utilizing the provided *Message Router* interface. An arbitrary string can be passed from the JavaScript context into a predefined method in the Main Process of the framework.

Mutation Observer

We discussed the method to induce C++ method callbacks from JavaScript, consecutively it is possible to mirror relevant parts of the DOM tree while it is being loaded. This can be achieved by implementing a `MutationObserver`[19] within JavaScript, such that it observes the DOM tree's root node and its children. The code that implements the creation of this observer class, is injected and executed on every V8 context's creation in the RenderProcessHandler. Each time changes take place within the DOM tree, a defined mutation function is called that receives a list of `MutationRecord` objects, which contains the information about the changes. A possible mutation is the adding of child nodes to any node included in the DOM tree. The observer classifies each added node by using predefined criteria. For example, the gathering of all text input fields requires a specific investigation of associated nodes. A text field is represented by a node of class `Element`, whose attribute `tagName` is set to `"INPUT"` and `type` contains `"text"`. This way, the node classification is performed for each tab in its Render Process itself and only relevant nodes have to be transferred between Render and Main Process. So, it is possible that all current tabs analyze their webpage's node structure simultaneously. Pointers to the DOM nodes of interest are stored in global lists within the V8 context, which are determined by the observer's classification result for this node.

[19] `https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver`

The direct JavaScript callbacks via Message Router are used to inform the Handler on the Main Process about occurring DOM node events (`add`, `update`, `remove`). To reduce complexity of encoding with a string, the callback transfers only minimal information like the node's id, its type or class and its changed attribute to the Handler. This information is sufficient in order to perform a data update of the extracted DOM node. An IPC request containing the identification of the DOM node is sent to the RenderProcessHandler, which reads the requested values from the global list of collected DOM nodes within the V8 context. Resulting values are then sent back over IPC to the Handler, where a custom DOM node structure is either created, updated or deleted. These DOM node structures on C++ side are finally shared with the Tab class of Visual Browser, which utilizes it for DOM node driven interfaces.

7 References

References

[1] Optikey: Type, click, speak. https://github.com/OptiKey/OptiKey/wiki.

[2] System usability scale (sus). http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html, 1986. Accessed: 27th April 2017.

[3] Kiyohiko Abe, Kosuke Owada, Shoichi Ohi, and Minoru Ohyama. A system for web browsing by eye-gaze input. *Electronics and Communications in Japan*, 91(5):11–18, 2008.

[4] Setare Amiri, Reza Fazel-Rezai, and Vahid Asadpour. A review of hybrid brain-computer interface systems. *Advances in Human-Computer Interaction*, 2013:1, 2013.

[5] K. K. Ang, Z.Y Chin, C. Wang, C. Guan, and H. Zhang. Filter bank common spatial pattern algorithm on bci competition iv datasets 2a and 2b. *Frontiers in Neuroscience*, 6:39, 2012.

[6] Hjorth B. The physical significance of time domain descriptors in eeg analysis. *Electroencephalography and clinical neurophysiology*, 34(3):321 – 25, 1973.

[7] Thierry Baccino and Yves Manunta. Eye-fixation-related potentials: Insight into parafoveal processing. *Journal of Psychophysiology*, 19(3):204–215, 2005.

[8] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Comput.*, 12(10):2385–2404, October 2000.

[9] Wolfgang Beinhauer. A widget library for gaze-based interaction elements. In *Proceedings of the 2006 Symposium on Eye Tracking Research &Amp; Applications*, ETRA '06, pages 53–53, New York, NY, USA, 2006. ACM.

[10] Guangyu Bin, Xiaorong Gao, Yijun Wang, Yun Li, Bo Hong, and Shangkai Gao. A high-speed bci based on code modulation vep. *Journal of neural engineering*, 8(2):025015, 2011.

[11] Ricardo Chavarriaga, Aleksander Sobolewski, and José del R Millán. Errare machinale est: the use of error-related potentials in brain-machine interfaces. *Frontiers in neuroscience*, 8, 2014.

[12] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[13] Mike X Cohen. *Analyzing neural time series data: theory and practice*. MIT Press, 2014.

[14] Adrien Combaz, Nikolay Chumerin, Nikolay V Manyakov, Arne Robben, Johan AK Suykens, and Marc M Van Hulle. Towards the detection of error-related potentials and its integration in the context of a p300 speller brain–computer interface. *Neurocomputing*, 80:73–82, 2012.

[15] Bernardo Dal Seno, Matteo Matteucci, and Luca Mainardi. Online detection of p300 and error potentials in a bci speller. *Computational intelligence and neuroscience*, 2010:11, 2010.

[16] Bernardo Dal Seno, Matteo Matteucci, and Luca T Mainardi. The utility metric: a novel method to assess the overall performance of discrete brain–computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(1):20–28, 2010.

[17] Daniel K Davies, Steven E Stock, and Michael L Wehmeyer. 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*, pages 107–113, 2001.

[18] Antonio Diaz-Tula and Carlos H. Morimoto. 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, pages 3533–3544, New York, NY, USA, 2016. ACM.

[19] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[20] Manuel JA Eugster, Tuukka Ruotsalo, Michiel M Spapé, Oswald Barral, Niklas Ravaja, Giulio Jacucci, and Samuel Kaski. Natural brain-information interfaces: Recommending information by relevance inferred from human brain signals. *Scientific reports*, 6:38580, 2016.

[21] Anna Maria Feit, Shane Williams, Arturo Toledo, Ann Paradiso, Harish Kulkarni, Shaun Kane, and Meredith Ringel Morris. 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*, pages 1118–1130. ACM, 2017.

[22] Andrea Finke, Kai Essig, Giuseppe Marchioro, and Helge Ritter. Toward frp-based brain-machine interfacessingle-trial classification of fixation-related potentials. *PloS one*, 11(1):e0146848, 2016.

[23] Youness Aliyari Ghassabeh, Frank Rudzicz, and Hamid Abrishami Moghaddam. Fast incremental lda feature extraction. *Pattern Recognition*, 48(6):1999 – 2012, 2015.

[24] Aryeh Gregor, Ms2ger, Alex Russell, Robin Berjon, and Anne van Kesteren. W3C dom4. W3C recommendation, W3C, November 2015. http://www.w3.org/TR/2015/REC-dom-20151119/.

[25] Human Performance Research Group. Nasa task load index (tlx): Paper and pencil package. `http://humansystems.arc.nasa.gov/groups/tlx/downloads/TLX_pappen_manual.pdf`, 1988. Accessed: 2nd May 2016.

[26] John Paulin Hansen, Hákon Lund, Hirotaka Aoki, and Kenji Itoh. Gaze communication systems for people with als. In *ALS Workshop, in conjunction with the 17th International Symposium on ALS/MND, Yokohama, Japan*, 2006.

[27] Bo Hjorth. Eeg analysis based on time domain properties. *Electroencephalography and clinical neurophysiology*, 29(3):306–310, 1970.

[28] Howell Istance, Richard Bates, Aulikki Hyrskykari, and Stephen Vickers. Snap clutch, a moded approach to solving the midas touch problem. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*, ETRA '08, pages 221–228, New York, NY, USA, 2008. ACM.

[29] Rob Jacob and Sophie Stellmach. What you look at is what you get: Gaze-based user interfaces. *interactions*, 23(5):62–65, August 2016.

[30] Rob Jacob and Sophie Stellmach. What you look at is what you get: Gaze-based user interfaces. *interactions*, 23(5):62–65, August 2016.

[31] Robert J. K. Jacob. 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, pages 11–18, New York, NY, USA, 1990. ACM.

[32] Melanie Kellar, Carolyn Watters, and Michael Shepherd. The impact of task on the usage of web browser navigation mechanisms. In *Proceedings of Graphics Interface 2006*, GI '06, pages 235–242, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[33] C. Kumar, R. Menges, and S. Staab. Eye-controlled interfaces for multimedia interaction. *IEEE MultiMedia*, 23(4):6–13, Oct 2016.

[34] Chandan Kumar, Raphael Menges, Daniel Müller, and Steffen Staab. 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, pages 219–223, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[35] Chandan Kumar, Raphael Menges, and Steffen Staab. Assessing the usability of gaze-adapted interface against conventional eye-based input emulation. 2017.

[36] Manu Kumar. *Gaze-enhanced user interface design*. PhD thesis, Citeseer, 2007.

[37] Manu Kumar, Jeff Klingner, Rohan Puranik, Terry Winograd, and Andreas Paepcke. Improving the accuracy of gaze input for interaction. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*, ETRA '08, pages 65–68, New York, NY, USA, 2008. ACM.

[38] Manu Kumar and Terry Winograd. Gaze-enhanced scrolling techniques. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 213–216, New York, NY, USA, 2007. ACM.

[39] Chris Lankford. Effective eye-gaze input into windows. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, ETRA '00, pages 23–27, New York, NY, USA, 2000. ACM.

[40] D Liparas, SI Dimitriadis, NA Laskaris, A Tzelepi, K Charalambous, and L Angelis. Exploiting the temporal patterning of transient vep signals: A statistical single-trial methodology with implications to brain–computer interfaces (bcis). *Journal of neuroscience methods*, 232:189–198, 2014.

[41] D. Liparas, S.I. Dimitriadis, N.A. Laskaris, A. Tzelepi, K. Charalambous, and L. Angelis. Exploiting the temporal patterning of transient vep signals: A statistical single-trial methodology with implications to braincomputer interfaces (bcis). *Journal of Neuroscience Methods*, 232(Supplement C):189 – 198, 2014.

[42] F. Lotte and C. Guan. Regularizing common spatial patterns to improve bci designs: Unified theory and new algorithms. *IEEE Transactions on Biomedical Engineering*, 58(2):355–362, Feb 2011.

[43] Steven J Luck. *An Introduction to the Event-Related Potential Technique*. MIT Press, 2014.

[44] Christof Lutteroth, Moiz Penkar, and Gerald 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*, UIST '15, pages 385–394, New York, NY, USA, 2015. ACM.

[45] Päivi Majaranta. *Text entry by eye gaze*. Tampereen yliopisto, 2009.

[46] Päivi Majaranta. *Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies: Advances in Assistive Technologies*. IGI Global, 2011.

[47] Kaira Matsuzawa and Chiharu Ishii. Control of an electric wheelchair with a brain-computer interface headset. In *Advanced Mechatronic Systems (ICAMechS), 2016 International Conference on*, pages 504–509. IEEE, 2016.

[48] Dennis J. McFarland, Lynn M. McCane, Stephen V. David, and Jonathan R. Wolpaw. Spatial filter selection for eeg-based communication. *Electroencephalography and Clinical Neurophysiology*, 103(3):386 – 394, 1997.

[49] W Pieter Medendorp, Herbert C Goltz, Tutis Vilis, and J Douglas Crawford. Gaze-centered updating of visual space in human parietal cortex. *Journal of neuroscience*, 23(15):6209–6214, 2003.

[50] Jianjun Meng, Shuying Zhang, Angeliki Bekyo, Jaron Olsoe, Bryan Baxter, and Bin He. Noninvasive electroencephalogram based control of a robotic arm for reach and grasp tasks. *Scientific reports*, 6, 2016.

[51] Raphael Menges, Chandan Kumar, Daniel Müller, and Korok Sengupta. Gazetheweb: A gaze-controlled web browser. In *Proceedings of the 14th Web for All Conference*, W4A '17. ACM, 2017.

[52] Darius Miniotas, Oleg Špakov, and I. Scott MacKenzie. Eye gaze interaction with expanding targets. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, pages 1255–1258, New York, NY, USA, 2004. ACM.

[53] Spiros Nikolopoulos, Kostas Georgiadis, Fotis Kalaganis, Georgios Liaros, Ioulietta Lazarou, Katerina Adam, Papazoglou-Chalikias Anastasios, Elisavet Chatzilari, P. Vangelis Oikonomou, C. Panagiotis Petrantonakis, I. Kompatsiaris, Chandan Kumar, Raphael Menges, Steffen Staab, Daniel Müller, Korok Sengupta, Sevasti Bostantjopoulou, Zoe Katsarou, Gabi Zeilig, Meir Plotnin, Amihai Gottlieb, Sofia Fountoukidou, Jaap Ham, Dimitrios Athanasiou, Agnes Mariakaki, Dario Comanducci, Eduardo Sabatini, Walter Nistico, and Markus Plank. *The MAMEM Project - A dataset for multimodal human-computer interaction using biosignals and eye tracking information*, July 2017.

[54] Takaaki Numajiri, Akio Nakamura, and Yoshinori Kuno. Speed browser controlled by eye movements. In *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo, ICME 2002, Lausanne, Switzerland. August 26-29, 2002. Volume I*, pages 741–744, 2002.

[55] LB Oknina, IuS Tolochko, EV Sharova, EL Masherov, and IuM Koptelov. Characteristics of the spatio-temporal organization of the p300 component of aep during the" active" and" passive" stimulus perception in healthy subjects. *Zhurnal vysshei nervnoi deiatelnosti imeni IP Pavlova*, 51(2):149–157, 2001.

[56] Balestrassi P. P., Paiva A. P., Zambroni de Souza A. C., Turrioni J. B., and Popova E. A multivariate descriptor method for change-point detection in nonlinear time series. *Journal of Applied Statistics*, 38(2):327 – 42, 2011.

[57] Lucas C. Parra, Clay D. Spence, Adam D. Gerson, and Paul Sajda. Recipes for the linear analysis of eeg. *NeuroImage*, 28(2):326 – 341, 2005.

[58] Abdul Moiz Penkar, Christof Lutteroth, and Gerald Weber. *Eyes Only: Navigating Hypertext with Gaze*, pages 153–169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[59] Michael Plöchl, José P Ossandón, and Peter König. Combining eeg and eye tracking: identification, characterization, and correction of eye movement artifacts in electroencephalographic data. *Frontiers in human neuroscience*, 6, 2012.

[60] Marco Porta and Alessia Ravelli. Weyeb, an eye-controlled web browser for hands-free navigation. In *Proceedings of the 2Nd Conference on Human System Interactions*, HSI'09, pages 207–212, Piscataway, NJ, USA, 2009. IEEE Press.

[61] C. Radhakrishna Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203, 1948.

[62] Bertrand Rivet, Antoine Souloumiac, Virginie Attina, and Guillaume Gibert. xdawn algorithm to enhance evoked potentials: application to brain–computer interface. *IEEE Transactions on Biomedical Engineering*, 56(8):2035–2043, 2009.

[63] Raphaëlle N Roy, Stéphane Bonnet, Sylvie Charbonnier, Pierre Jallon, and Aurélie Campagne. A comparison of erp spatial filtering methods for optimal mental workload estimation. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 7254–7257. IEEE, 2015.

[64] Andrea Kbler Ruben G.L. Real. Auditory oddball paradigm during hypnosis. `http://bnci-horizon-2020.eu/database/data-sets`. Accessed: 2017-10-30.

[65] Korok Sengupta, Chandan Kumar, and Steffen Staab. Usability heuristics for eye-controlled user interface. In *The 2017 COGAIN Symposium: Communication by Gaze Interaction*, 2016.

[66] Martin Spüler, Wolfgang Rosenstiel, and Martin Bogdan. Online adaptation of a c-vep brain-computer interface (bci) based on error-related potentials and unsupervised learning. *PloS one*, 7(12):e51077, 2012.

[67] Martin Spüler, Armin Walter, Wolfgang Rosenstiel, and Martin Bogdan. Spatial filtering based on canonical correlation analysis for classification of evoked or event-related potentials in eeg data. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(6):1097–1103, 2014.

[68] Martin Spler, Wolfgang Rosenstiel, and Martin Bogdan. Online adaptation of a c-vep brain-computer interface(bci) based on error-related potentials and unsupervised learning. *PLOS ONE*, 7(12):1–11, 12 2012.