Diss. ETH No. 25545

Exploring Motion-Based and Mid-Air Gestures in Mobile Web Contexts

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

Linda Di Geronimo

Laurea Magistrale in Informatica
Università degli Studi di Salerno, Italy

born June 7, 1989
Battipaglia (SA), Italy

accepted on the recommendation of

Prof. Dr. Moira C. Norrie, examiner
Prof. Dr. Otmar Hilliges, co-examiner
Prof. Dr. Aaron Quigley, co-examiner
Prof. Dr. Beat Signer, co-examiner

2018

Abstract

In the last decade, the use of mobile devices such as smartphones and tablets has increased dramatically. Portable devices are now capable of achieving a multitude of tasks and, therefore, have become an important tool in our everyday life. Moreover, users often own more than one device and, for this reason, cross-device applications have raised interest among researchers and companies. In this multi-device era, improving the overall user experience on mobile devices in single and cross-device scenarios is particularly critical. Despite this need, the form of interaction currently used on these devices relies mainly on touch. While touch gestures offer an intuitive way of interacting with smartphones and tablets, they can fail in certain situations. For instance, touch often requires the user to change their holding position to reach some locations on the screen. Moreover, to perform touch gestures users need to look at the screen and they are often required to held the device with both hands.

To overcome these issues, researchers have proposed alternative forms of interactions such as tilting and mid-air gestures. Tilting interactions are movements of the device in some direction and, given their nature, they are one-hand and eyes-free interactions. Alternatively, researchers have also exploited the use of the phones' cameras to perform mid-air poses to trigger actions on the device and, therefore, enlarge the interaction zone of the phone or tablet. However, this research has been applied mainly to native applications rather than on mobile web browsers.

On the web, the primary focus has been on the adaptation of content to the screen real estate of the clients rather than its modes of interactions. While responsive design solved some of the issues concerning the usability of web applications on mobile devices, it does not tackle any alternative forms of gestures.

In this thesis, we bridge this gap via a series of contributions aimed at improving the mobile user experience by enriching the set of possible interactions available on single and cross-device web applications. We start with an in-depth analysis of the literature on tilting gestures and, based on these, derived a set of API as Tilt-and-Tap (TAT), a JavaScript framework for the rapid development of motion gestures on the web. TAT offers a catalogue of previously proposed tilting interactions and allows developers to customise gestures with a number of different parameters. TAT has the goal of encouraging developers to use tilting interactions in their web applications by hiding implementation details and supporting coherent interactions across different platforms and browsers.

While TAT targeted developers, we also wanted to allow end-users with little or no experience in web technologies to experiment with motion gestures. To reach this goal, we used the drag and drop UI paradigm to add gestures to a web application. To test our approach, we developed WP-TAT, a WordPress extension that allows users to map tilting gestures to global actions via drag and drop interactions. Global actions are functions which affect the entire website. For instance, a rapid movement of the device to the left or to the right could redirect the user to the previous or next posts. Users can define these mappings by using drag and drop interactions on our visual interface and, therefore, they are not required to write any code.

Concerning cross-device scenarios, a preliminary user study output a series of requirements to improve the user experience, as well as the development process, of cross-device applications that exploit tilting interactions. Inferred by these requirements we developed Cross-Tilt-and-Tap (CTAT), a set of high-level APIs for the rapid prototyping of motion interactions in cross-device environments. After the developer has attached to the desired gesture (i.e. tilt right), the sender of the message and its receivers, the framework will be responsible for recognising the devices involved in the communication, detect the motion interaction and sending the messages.

Finally, we continued our work on cross-device applications by employing the use of mid-air gestures to share data across devices. Via an elicitation study, we asked users to suggest a series of gestures to target co-located and remote devices. The proposed gestures, as well as the feedback received from participants, informed the design of MyoShare, a system that allows users to copy web data from and to their desktop computers and mobile devices.

To evaluate our frameworks and tools, we have carried out a number of user and developer studies as well as proposing various example applications.

Abstract

Nell'ultimo decennio, l'utilizzo di dispositivi mobili come smartphone e tablet è aumentato drasticamente. I dispositivi portatili sono ora in grado di raggiungere una moltitudine di compiti e, pertanto, sono diventati uno strumento importante nella nostra vita quotidiana. Inoltre, gli utenti spesso possiedono più di un dispositivo e, per questo motivo, le applicazioni cross-device hanno suscitato gran interesse tra ricercatori e aziende. In questa *multi-device era*, il miglioramento dell'esperienza utente su dispositivi mobili in scenari single e cross-device è particolarmente critica. Nonostante questa necessitá, la principale forma di interazione attualmente utilizzata è il touch. Mentre le gesture touch offrono un modo intuitivo di interagire con smartphone e tablet, possono fallire in determinate situazioni. Ad esempio, il touch spesso richiede all'utente di modificare la posizione della loro mano per raggiungere alcune zone sullo schermo. Inoltre, per eseguire touch gesture, gli utenti devono guardare lo schermo e spesso sono tenuti a tenere il dispositivo con entrambe le mani.

Per superare questi problemi, i ricercatori hanno proposto forme alternative di interazione come tilting e mid-air gesture. Le interazioni di tilting sono movimenti del dispositivo in qualche direzione e, data la loro natura, possono essere eseguite con una sola mano e senza necessariamente guardare il dispositivo. In alternativa, i ricercatori hanno anche sfruttato l'uso delle fotocamere dei telefoni per mid-air gesture per eseguire azioni sul dispositivo e, quindi, ingrandire la zona di interazione del telefono o del tablet. Tuttavia, questa ricerca è stata applicata principalmente alle applicazioni native piuttosto che ai browser web.

In questa tesi, colmiamo questa lacuna proponendo una serie di contributi scientifici volti ad ampliare l'insieme delle possibili interazioni su applicazioni web. Dopo un'analisi approfondita della letteratura riguardo le interazioni di tilting, abbiamo sviluppato Tilt-and-Tap (TAT), un framework JavaScript per lo sviluppo rapido di motion gestures sul web. TAT offre un catalogo di interazioni di tilting proposte in letteratura e consente agli sviluppatori di personalizzare le gesture modificando una serie numerosa di parametri customizzabili. TAT ha l'obiettivo di incoraggiare gli sviluppatori ad utilizzare le interazioni di tilting nelle loro applicazioni web, nascondendo i dettagli implementativi delle gesture supportate e mantenendone la coerenza tra piattaforme, nonostante le lore differenze.

Mentre TAT mira ad aiutare gli sviluppatori, con WP-TAT, coinvolgiamo anche gli utenti finali con poca o nessuna esperienza in tecnologie web ad sperimentare con le motion gestures. WP-TAT è un'estensione di WordPress che consente agli utenti di abbinare le interazioni di tilting a delle azioni globali. Le azioni globali sono funzioni che, quando eseguite, hanno effetto sull'intero sito web. Ad esempio, un rapido movimento del dispositivo verso sinistra o verso destra potrebbe reindirizzare l'utente al post precedente o successivo. Gli utenti possono definire questi abbinamenti tra azioni e gesture, attraverso delle semplici operazioni di drag e drop dalla nostra interfaccia grafica e, di conseguenza, senza la necessitá di inserire nessuna riga di codice.

Per quanto riguarda gli scenari cross-device, uno studio utente preliminare ha prodotto una serie di requisiti per migliorare l'user experience e il processo di sviluppo di applicazioni cross-device che sfruttano le interazioni di tilting. In base a questi requisiti, abbiamo sviluppato Cross-Tilt-and-Tap (CTAT), un set di API per la prototipazione rapida di interazioni di tilting in scenari cross-device. Con CTAT, lo sviluppatore deve solo definire la gesture desiderata, il mittente e il destinatario dellinterazione, a questo punto, il framework sarà responsabile del riconoscimento dei dispositivi coinvolti nella comunicazione, rilevare la gesture e inviare i messaggi ai giusti destinatari.

Infine, abbiamo continuato la nostra ricerca utilizzando mid-air gesture per la condivisione di dati tra dispositivi. Tramite un elicitation study, abbiamo chiesto agli utenti di suggerire una serie di gesture per identificare dispositivi co-localizzati e remoti. Le gesture proposte, ed i feedback ricevuti dai partecipanti, hanno influenzato il design di MyoShare, un sistema che consente agli utenti di copiare dati web da e verso i loro computer desktop e dispositivi mobili.

Per studiare e valutare i nostri tool e framework, abbiamo condotto una serie di studi utente, inoltre, abbiamo implementato varie applicazioni desempio implementate utilizzando le API e i sistemi proposti in questa tesi.

To Nonno Bruno, Nonno Giulio and Paolo.

*"It is possible to commit no mistakes and still lose.
That is not weakness, that is life."*

Jean-Luc Picard

Table of Contents

1 Introduction

More than ten years have passed since the advent of the first iPhone. On January 9th 2007 at the MacWorld keynote[1], Steve Jobs presented a device that would drastically change the mobile phone world and, as a consequence, our everyday lives. Back then, Blackberries and other similar devices already offered some of the key features present in the first iPhone. For example, they were capable of sending and receiving emails, and could play music, manage calendars and connect to Bluetooth devices such as wireless headphones. Most of these devices offered a physical QWERTY keyboard to allow a typing technique similar to the one people were accustomed to on desktop and laptop machines.

This generation zero of smartphones were often used for work-related tasks, but they were big and cumbersome to use and carry. In this setting, the multi touch capabilities of the iPhone, its innovative design, its futuristic user interfaces and the newly proposed gesture set felt like a fresh new start.

Given the enormous success of the iPhone, the technology around these devices evolved rapidly. In a single decade, our smartphones became smarter, faster and thinner. As a consequence, a large proportion of the world's population now use and perceive mobile phones in a completely different way to the early years of the 21st century. Nowadays, phones are multi-functional devices that serve a multitude of tasks, with calling and messaging people only two of the many functions that they offer.

One key factor that makes these devices smart relies on their sensing capabilities. Touch-enabled displays allow intuitive gestures to be used to interact with the device. Ambient light sensors are capable of detecting the current amount of light so the brightness of the screen can be adjusted accordingly. Near-field communication (NFC) sensors allow two physically close electronic devices to initiate a communication, and accelerometers and gyroscopes can detect the speed and orientation of the device.

Given the number of sensing techniques available, relying only on touch to interact

[1]Introduction of the first iPhone - Presentation by Steve Jobs (2007): https://youtu.be/9hUIxyE2Ns8. Accessed on November 2017.

with the smartphone seems limiting. Moreover, in the last years, the size of smartphones has increased causing users to often change their holding position to reach some locations on the screen and hold the device with both hands. For these reason, researchers have exploited the use of the camera, motion and other sensors to propose alternative means of interaction with the final goal of improving the overall user experience and fix some of the issues of touch interactions.

For example, tilting gestures have been proposed as a possible alternative to touch interactions [179, 15, 92, 108]. These motion gestures allow the user to interact with the device by simply moving it in some direction in a fast or continuous manner. Through tilting interactions, users can perform gestures without touching the screen, and therefore, without changing their holding position. Moreover, via tilting interactions, users can perform gestures without looking at the screen and interact with the device also when wearing gloves or have dirty fingers while cooking or eating.

Alternatively, mid-air gestures have also been used as another possible form of interaction [149, 9]. By mimicking some gestures or posing hands or fingers in a specific way, users can pause videos or go to the next song without the need of carrying an additional remote in their hands. The interaction is usually recognised by a camera or by wearable devices worn by the user.

While many researchers have experimented with these alternative modes of interaction in native applications, the primary focus on the web has remained on the organisation of content rather than modes of interactions [152].

In the early days of the first iPhones and Android devices, accessing web content on a mobile phone required the users to zoom in and out repeatedly. The fat finger problem caused users to inadvertently tap incorrect links or buttons in a web page because of the small size of the touch areas [44]. For these reasons, reading an article or merely browsing a list of products was perceived as painful tasks. Only later, did the concept of responsive web design became prominent in web development. A responsive website is capable of adapting to the screen where it is rendered according the type and dimension of the device itself. This was possible with the introduction of CSS media queries in 2009 which allow developers to define rules that will be applied only if certain conditions are satisfied. For example, developers can write rules that will be rendered only if the width of the device viewport is smaller or larger than a threshold. These breakpoints allow better coordination of the elements in the page when dealing with different screen real estates making it easier for developers to build their responsive website.

Thanks to media queries and, overall, to responsive design, web applications gradually became easier to browse on mobile devices. However, many years had to pass until most websites would be smoothly accessible in a wide variety of display sizes.

In June 2014, the number of internet accesses from mobile devices surpassed desktop computers[2] with an estimated 60% of the time users spent on their daily digital media consumption made from a mobile device. However, in 2015, eight years after Steve Jobs presented the first iPhone, as reported by the *The Economist*, 40% of the leading websites were not yet mobile friendly[3].

[2]The U.S. mobile app Report (2014): https://goo.gl/qse47h. Accessed on November 2017.

[3]Google Mobilegeddon. The world's biggest search engine shakes up its algorithms (2015): goo.gl/19nM63. Accessed on November 2017.

The Google algorithm update of April 21st 2015, known as the *Mobilegeddon*, had the goal of reducing this number by de-ranking pages that failed to pass a mobile-responsiveness test[4]. As a result, many were forced to implement a mobile strategy for their website to not be affected by the de-ranking[5].

However, although more and more websites are now easily accessed from mobile devices, users still prefer to access digital media from native apps rather than from web browsers. As reported by ComScore[6], out of the total time users spend on smartphones on a daily basis, on average 80% of this time is consumed on apps. One of the reasons could be how users still perceive the web. Overall, apps could seem more finger friendly and reactive than mobile web pages. For this reason, while responsive design was an essential first step to a more adaptive web, it still does not solve all of the issues. A responsive website re-arranges and changes the dimensions of elements to cater for a specific screen real estate but it does not address different interaction techniques that could potentially improve its usability.

This lack of attention towards alternative modes of touch interaction on the web could be due to the poor support available for developers, especially given the compatibility problems among browsers. While most browsers offer APIs to access raw data of almost all sensors available on smartphones, their implementations often do not follow the W3C[7] standards, resulting in significant differences among devices and browsers. Also, for this reason, there is a lack of high-level APIs that support developers in experimenting with alternative forms of interaction in their web applications.

With jQMultiTouch, Nebeling and Norrie [151] propose a framework for the rapid development of multi touch gestures in web applications. jQMultiTouch was one of the first works that went beyond the classic set of touch interactions while fixing cross-browser compatibility problems. The framework had the overall goal of supporting investigations into new forms of interaction on the web and therefore, to easily allow further research in the area. Despite these first steps, there is still remains a lot of work to be done on investigating how to enlarge the set of possible interactions used in web applications, including tilting and mid-air gestures.

1.1 Motivation

The primary goal of this thesis is to improve the user experience of mobile web browsing by enriching the set of possible interactions available. To reach this goal, we investigate how to best support developers in expanding the set of possible interactions used in web applications.

Since the first generation of smartphones, developers were capable of accessing raw data from various sensors in their native apps and had more power in manipulating and using this information compared to mobile browsers. This might be one of the reasons why alternative gestures have not yet been adequately studied in the context of web applications. However, nowadays, browsers incorporate plenty of new features, and they propose a valid alternative to more native solutions.

[4]Google Mobile-friendly test: https://goo.gl/wm49AR. Accessed on November 2017.
[5]Google's Mobilegeddon aftermath (2016): https://goo.gl/lzZqGd. Accessed on November 2017.
[6]The Mobile Report (2017): https://goo.gl/EbJ5xD. Accessed on January 2018.
[7]W3C website: https://www.w3.org/. Accessed on January 2018.

Currently, many APIs are available to offer a broader set of functionalities. Web applications can now push notifications to the clients, can be launched in full screen, and can also access raw data of the various sensors installed on the mobile device. For example, web developers can make use of the camera of the phone; can locate the user via the GPS and can also get information about the current orientation of the phone via motion sensors, such as accelerometers and gyroscopes.

Thanks to this set of hardware and software capabilities, the web can move forward and support richer applications as well as allowing alternative interaction techniques that go beyond touch gestures. However, supporting tilting or mid-air gestures can be challenging. Given the lack of support, developers are often forced to work directly with the raw data given by the sensors. This requires particular knowledge of how, for example, the accelerometer and gyroscope work, what data they return and their limitations and boundaries. This knowledge is usually hard to obtain and, often, it requires various trial and error experiments. Acquiring these skills requires time and resources that are not always available when developing applications. Especially on the web, these problems can be particularly challenging to manage for two main reasons: the weak compatibility across browsers and the often limited technical background of web developers.

Despite the improvements made over the years, compatibility across browsers remains one of the main issues that web developers have to face. When a new standard is introduced, it is not clear if all browsers will support it eventually and, even if they do differences are often found in the implementations. On this matter, websites such as caniuse.com[8] allow developers to check if a specific API is supported by browsers and at which level. Given this information, developers are then forced to create alternative implementations to cater for a specific set of browsers. Researchers have also proposed specific IDEs to alleviate this problem [144].

Access to data sensors, such as accelerometers and gyroscopes, is particularly affected by cross-browser compatibility problems. Every device is equipped with different sensors that can be more or less accurate and, therefore, return data with different granularities. On top of the hardware differences, iOS and Android devices tend to deal with sensors in contrasting ways despite standard specifications.

Another critical factor that slows down experimentation with alternative forms of interaction regards the technical skills of developers that work with web technologies. Often web developers have little or no formal education in computer science. Many are self-taught and rely on tools or online platforms to create their website [160]. In such environments, Content Management Systems (CMS) have become increasingly popular over the years. For example, 60 million websites are supported by WordPress[9], which is currently the most famous and widely used CMS[10].

Thanks to CMSs, users can create their websites in a few clicks. They can pick and customise themes that will define the look and feel of their web application and add features via plugins without requiring any particular coding skills. However, most of these systems mainly focus on the customisation of the layout of a website and rarely support any additional interaction technique. Moreover, CMS mainly focus on

[8]caniuse website: https://caniuse.com/. Accessed on November 2017.
[9]WordPress website: https://it.wordpress.com/. Accessed on November 2017.
[10]CMS Usage Statistics (2017): https://goo.gl/wuuvav. Accessed on November 2017.

single device scenarios while there is an increasing interest in cross-device applications [13, 52, 80].

Cross-device applications have the goal of benefiting from the differences among devices when used in conjuction [188]. For example, the input capabilities of a mobile phone could allow users to quickly search and select the next video to watch, while a larger display, such as a TV, can show the desired media. In this scenario, the smartphone provides a comfortable alternative to typing, while a bigger screen allows a more enjoyable experience for watching a movie.

In cross-device environments, tilting interactions have also been experimented with for pairing and sharing data among devices [220] and also to remotely control public screens [23, 45]. A fast tilt of a smartphone to the right or the left could allow users to browse a gallery of media shown on a bigger display, or send content towards other devices that share the same connection. Similarly, mid-air gestures can be used to control other devices as well as share data among them [200, 37].

This means that the challenges of detecting alternative interaction techniques becomes spread across multiple devices. Each client has to manage the recognition of the gesture, and once the user performs the interaction, the client has to send the message to the desired set of connected devices.

Given this environment, the lack of applications that exploit alternative gestures does not come as a surprise. Despite the current software and hardware opportunities, touch remains the primary form of interaction in single and cross-device web applications.

We hope that, when supported with the right tools and applications, both developers and end-users can go beyond the classic set of gestures on mobile web browsing and experiment with alternative forms of interaction. We also hope that with our support, the overall interest in motion and mid-air gestures on the web might inspire further research in the area.

The main research questions of this thesis can be summarised as follows:

- **RQ 1**: *How can we support both developers and end-users in building single and cross-device web applications that use tilting interactions?*

- **RQ 2**: *Which single and cross-device applications can benefit from tilting or mid-air gestures?*

1.2 Challenges

In summary, we can identify two main challenges that arise when supporting and experimenting with alternative interaction techniques in single and cross-device web browsing. First, while many responsive web design frameworks and general guidelines are currently available online, their focus is mainly on how to best adapt UI elements to different screen real estates rather than on different interaction techniques. For this reason, end-users can usually rely on a more finger-friendly mobile experience. However, touch has remained their primary mode of interaction although alternative gestures, such as tilting and mid-air gestures, could bring improvements of the overall user experience [92, 23]. The lack of support available is a challenge that developers have to face when experimenting with alternative gestures.

Building applications that propose new forms of interaction requires additional skills and resources. Working with sensor data, managing the right thresholds to detect gestures as well as dealing with software and hardware compatibility problems can be challenging. A multitude of parameters could also influence each gesture. For example, researchers have proposed tilting gestures in several variants [205, 23, 162], also in combination with touch gestures [92] as well as tactile or visual feedback [162]. Moreover, in cross-device scenarios, gestures have first to be detected by each device involved in the communication, and then messages need to be exchanged in real time. In such environments, devices of various sizes, different platforms and browsers should communicate smoothly and in an unobtrusive way. Interactions should give the impression that there are no hardware or software boundaries among the different clients involved. Achieving these goals requires particular attention to designing and implementing gestures that are consistent among devices, despite their differences.

These issues become particularly problematic when practitioners with little or no coding skills want to experiment with alternative means of interaction. While CMSs allow end-users to create their websites without requiring any particular technical background, they focus on the look & feel of the final product and offer no support for novel modes of interactions.

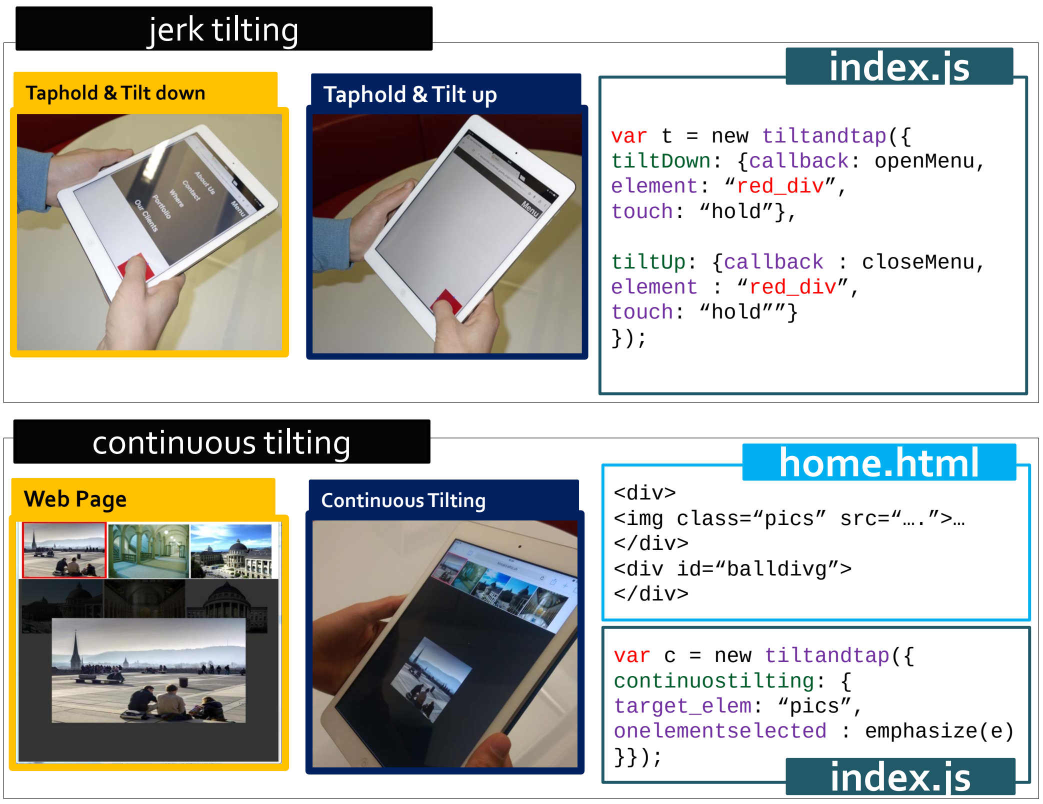
Finally, a second challenge is the question of where and how to apply new gesture sets. As a consequence of the lack of support for non-standard interactions, there is a lack of applications that exploit alternative gestures for mobile web browsing. Therefore, there is still a need for investigation to understand in which scenarios new gestures are best suited and if they could improve the overall user experience when compared to traditional interactions.

1.3 Contribution

This thesis makes seven contributions to the overall goal of enlarging the set of possible interactions used for mobile web browsing and supporting rapid experimentation with new gesture sets.

Our **first contribution** relies on a detailed analysis of motion gestures, their implementation and possible variants as suggested by previous research in the field. These interactions were then included in Tilt-and-Tap (TAT), a JavaScript framework that supports the development of two main forms of motion interactions, namely jerk and continuous tilting gestures (see Figure 1.1).

We identified jerk tilting interactions as rapid movements of the device in ten different directions (up, down, left, right, south-east, south-west, north-east, north-west, clock and counterclockwise). In contrast, continuous tilting gestures are slow movements of the device to, for example, scroll a series of pictures in a gallery. Both jerk and continuous interactions can be combined with touch gestures. For example, users can tap or hold tap on a specific element in the page while performing the tilting interaction. Moreover, visual and audio feedback can be triggered once the motion gesture has been recognised. Developers can simply specify these parameters, as well as the callback functions to be called once the gestures have been recognised, in the form of strings to the TAT instance, thereby avoiding the implementation details of the supported interactions. Behind the scenes, the framework also deals with differences among



```
var t = new tiltandtap({
tiltDown: {callback: openMenu,
element: “red_div”,
touch: “hold”},

tiltUp: {callback : closeMenu,
element : “red_div”,
touch: “hold””}
});
```

```
<div>
<img class=“pics” src=“….”>…
</div>
<div id=“balldivg”>
</div>
```

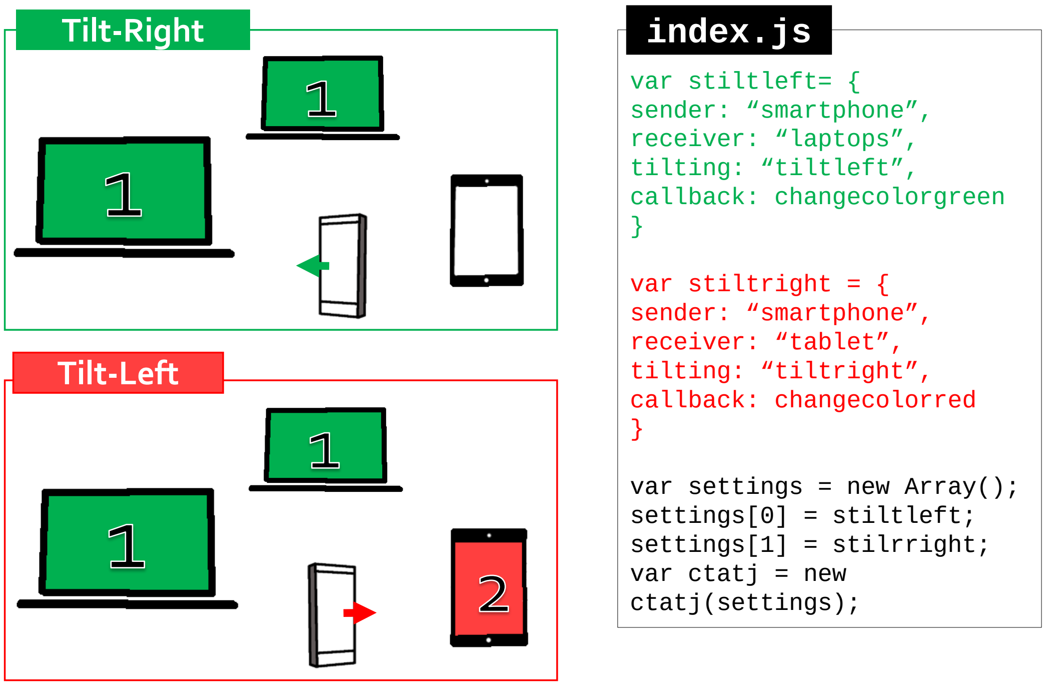
```
var c = new tiltandtap({
continuostilting: {
target_elem: “pics”,
onelementselected : emphasize(e)
}});
```

Figure 1.1: Jerk and continuous tilting examples and their corresponding implementation. In the first example, users can open and close a menu by holding the red box on the bottom right side of the page and tilting the device up and down. In the second example, users can browse a list of pictures by slowly moving the device to the right and the left.

browsers and hardware sensors allowing the developers to freely experiment with the gestures without having to deal with compatibility issues. In a second stage, we extended Tilt-and-Tap to support more variants of continuous tilting gestures and developed TAT 2.0.

With TAT and TAT 2.0, we encourage developers to experiment with motion interactions in their web applications, however both frameworks target users with some technical background in web technologies. With our **second contribution**, we also involve end-users by exploiting the drag and drop UI paradigm to add tilting gestures to web applications. To test our proposed approach, we developed WP-TAT, a WordPress extension for the rapid prototyping of tilting gestures. Via drag and drop interactions, users can associate jerk tilting gestures with a number of actions, such as go to next post or page, search the current selected text on Google or Google maps, click buttons and so on. Users are not required to write any code and can extend web applications with alternative forms of interaction through a simple to use GUI.

With these first two contributions, we tackled RQ1 and studied how to best support developers and end-users in building single device web applications that exploit tilting gestures. With our **third contribution**, we address RQ2 by listing a series of design guidelines on how to best apply motion gestures in web applications and in which scenarios these interactions can be beneficial on the web. These design observations



```
var stiltleft= {
sender: “smartphone”,
receiver: “laptops”,
tilting: “tiltleft”,
callback: changecolorgreen
}

var stiltright = {
sender: “smartphone”,
receiver: “tablet”,
tilting: “tiltright”,
callback: changecolorred
}

var settings = new Array();
settings[0] = stiltleft;
settings[1] = stilrright;
var ctatj = new
ctatj(settings);
```

Figure 1.2: Examples of cross-device jerk tilting interactions and their implementation with CTAT. Once the user performs a tilt left with a smartphone, the background colour of laptops involved in the communication will change to green. Instead, if the system recognises a tilt right, the background colour of the tablet will change to red.

have been inferred by the experience we gathered during development of TAT, TAT 2.0, WP-TAT and the feedback received from participants of our developer and user studies on the above mentioned tools and frameworks.

As a **fourth contribution**, we exploited TAT in cross-device scenarios and carried out a preliminary user study. The goals of the study were to first show the advantages of motion gestures in cross-device environments, and second, list a series of requirements to improve the development process of cross-device applications that use tilting interactions. Informed by these requirements, we extended TAT to also work across different devices via Cross-device Tilt-and-Tap (CTAT), a JavaScript framework for the rapid development of motion gestures across multiple devices. CTAT allows developers to simply define for each tilting interaction its desired sender(s) and the receiver(s), as well as the actions to execute once the gesture has been performed (see Figure 1.2). CTAT deals with the exchange of messages between clients, and it is capable of identifying the right subsets of devices that are involved in the interaction. Without CTAT, developers who would like to use motion gestures in their cross-device applications, would have to manage the recognition of tilting interactions on each client, recognise the right receivers of the trigger and send messages back and forth to achieve the desired result. Although technologies such as Node.js[11] and Socket.IO[12] help developers to create cross-device applications they often require a ping-pong exchange of messages between clients and the server for each request. CTAT reduces this effort while sup-

[11]Node.js website: https://nodejs.org. Accessed on December 2017.
[12]Socket.IO website: https://socket.io/. Accessed on December 2017.

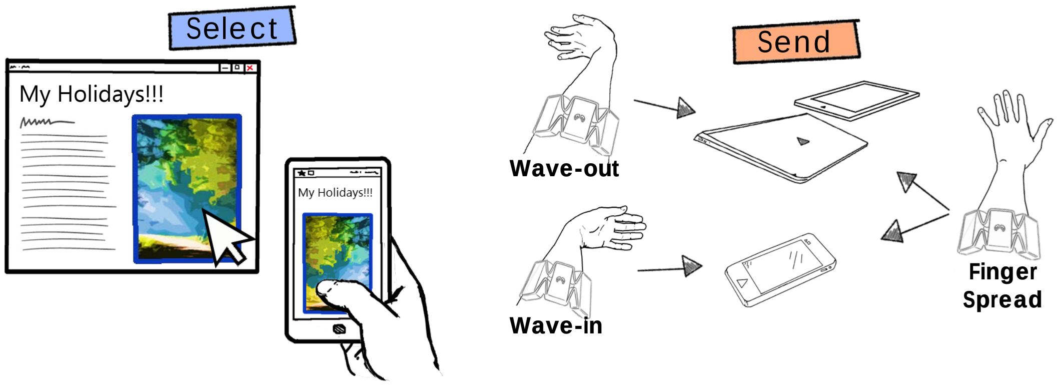


Figure 1.3: An example scenario of MyoShare. Users select from their mobile device or desktop computer, an image that they would like to share. Once the selection is made, users can perform different gestures to target specific sets of devices.

porting all the main features offered by its single device version. Moreover, to further improve the usability of the framework, a tool has been developed to automatically generate the code for the desired cross-devices tilting interactions. Thanks to this GUI, developers can add devices such as laptops, tablets, mobile phones and smartwatches and create *tilting links* among them. These links define the motion gestures between sets of clients involved in the communication. As shown in Figure 1.2, tilting links can be created by simply using JavaScript objects and defining the desired source of the gesture (in figure a smartphone) and possible targets (laptops and tablets).

In our **fifth contribution**, we combine the cross-device features offered by CTAT and the new extended version of TAT 2.0, to study usability performance of different variants of continuous tilting interactions to remotely control a cursor on a public screen. Via a user study, we evaluated different motion gestures previously proposed by researchers but never compared in cross-device environments [205, 23, 162]. Moreover, as similarly done for single device scenarios, based on the results of the study, we infer possible use cases of how best to apply tilting gestures in cross-device applications. The fourth and fifth contributions of this thesis tackle RQ1 and RQ2 when tilting gestures are used in cross-device environments.

To bring our research forward, with our **sixth contribution**, we explored the use of mid-air gestures to extend the set of possible interactions available on the web. While mid-air gestures provide an intuitive way of interacting across different devices, they have mainly been exploited when all the clients are co-located and easily accessible [69, 12, 37]. However, users do not always carry all their devices with them [52]. For this reason, we were interested in exploring how best to apply such gestures in remote cases. We conducted an elicitation study that defined a set of gestures that could work when devices are not physically near to the user. The study gave us the chance to understand how users mentally associate mid-air gestures to devices that are not co-located. Moreover, the study results and the user feedback on our scenario, informed the implementation of the proposed gestures of MyoShare, a system that allows users to share data across devices via mid-air gestures. With our system, users can select an element they would like to share from their mobile device or desktop computer and then send it by performing a mid-air gesture. Gestures are detected via the Myo armband[13]

As can be seen in Figure 1.3, the interaction performed will infer the right receiver of the message. In this case a wave-in interaction will send the data to the smartphone of the user, while a wave-out to their secondary sets of devices such as a laptop and a tablet. Meanwhile, a finger spread will broadcast the data selected to all the registered devices.

Via its Chrome extension, MyoShare is also capable of understanding the nature of the selected web content. If the user selects a phone number and then shares it, it can be accessed and manipulated on the target device via an Android application. By clicking on the item, the system will automatically copy the number in the phone dialogue of the device. Similarly, if users share plain text, it can be copied to the clipboard, while if images or videos are sent, they can be saved in the gallery.

With MyoShare, we were also able to compare different interaction techniques for the scenario we envisioned via a user study that represents the **seventh and last contribution** of this thesis. As result, we found that mid-air gestures are as fast as other more common interaction techniques (shortcuts and UI menus) and users enjoyed experimenting with these gestures to send data across devices. The elicitation and user study gave us the chance to address RQ2 when mid-air gestures are used in cross-device scenarios.

Thanks to these works, we were able to experiment with a number of single and cross-device web applications that could benefit from alternative interactions such as tilting or mid-air gestures. In this thesis, we will present applications for each of the proposed contributions. We will also report on a variety of users and developer studies that have been carried out to evaluate the power of the proposed approaches also when compared to more classic interactions.

The research methodology of this thesis follows the DSRM process model studied by Peffers et al. [170]. After defining the problems, motivations and goals for each contribution proposed, we design and developed artefacts to demonstrate and evaluate our suggested approaches.

1.4 Thesis Overview

The thesis is organised as follows: Chapter 2 discusses related work, as well as current applications that exploit the use of tilting and mid-air gestures. We analyse gaps found in the literature as well as gathering knowledge from previous research on the use of alternative forms of interaction to list possible requirements for our frameworks, tools and applications.

In Chapter 3, we discuss our proposed solutions for single device scenarios. Tilt-and-Tap, its extended version TAT 2.0 and WP-TAT are presented. The architecture and implementation details of the proposed approaches will be discussed together with a number of applications built using these frameworks and tools. Developer and user studies are also presented along with related discussions including possible future improvements. We also discuss compatibilities and performance issues related to tilting interactions on web applications. Moreover, inferred by user feedback as well as our

[13]Myo website: https://www.myo.com. Accessed on December 2017.

experience in the topic, a list of guidelines on how and where to apply tilting gestures is presented at the end of the chapter.

Chapter 4 moves beyond single device scenarios where CTAT is presented along with details of its architecture and implementation as well as its supported APIs. Showcase applications built with CTAT as well as two user studies are reported: one with the aim of gathering general feedback on the approach when multiple users perform tilting interactions in cross-device applications, and another to compare different motion gestures for interacting with a public screen. Moreover, we will discuss the visual tool to generate CTAT instances.

MyoShare is presented in Chapter 5. We report on our envisioned scenario on sharing data via mid-air gestures as well as discussing our proposed solution. We present the elicitation study conducted to explore possible mid-air gestures to share data across co-located and remote devices. Together with the general architecture and implementation of MyoShare, we also report on the user studies conducted to evaluate mid-air gestures in our cross-device scenario.

Finally, in Chapter 6, we conclude the thesis with a discussion on the contributions that outlines limitations of our approaches as well as possible future work.

2 Background

The research community has explored intensely the use of alternative interaction techniques on mobile devices. However, despite the current possibilities available on mobile web browsers, these alternative gestures have yet to be deeply exploited on the web in both single and cross-device scenarios. In Section 2.1, we first analyse the state of the art of single and cross device web applications, their evolution over the past decade as well as discussing the challenges that arise when adding new forms of interaction. It is important to note that, in this thesis, we focus on mobile web applications rather than web applications in general. In contrast, in Section 2.2, we present gestures suggested by researchers that go beyond the classic set of touch gestures in mobile applications. Following that, we discuss cross device applications in more detail in Section 2.3 by describing their potential in everyday scenarios. In this part, we focus on related work that studies interactions in applications that run across multiple devices. Finally, we outline gaps found in the works presented and discuss conclusions in Section 2.4.

2.1 Mobile Web Applications

In recent years, many developers have opted to build mobile web applications for their IT solution. However, the primary form of interaction available on mobile browsers relies only on touch gestures. One of the reasons could be related to the difficulty of extending the set of possible interactions on the web for both single and cross-device scenarios. We decided to investigate how both developers and end-users could be encouraged to experiment with alternative interaction modalities. With the support of visual tools and frameworks, we help users in exploring a broader set of forms of interaction on their websites. Given the increasing popularity of cross-device applications [13, 52, 80], we also explored alternative interactions when more heterogeneous devices are involved in the communication. We start by discussing why building mobile web apps as opposed to native applications can be beneficial in Section 2.1.1. Then in Section 2.1.2, we study how developers and end-users can currently build their web applications and what challenges they have to face when adding new interaction techniques on the web.

In this thesis we use the terms *website* and *web application* (also called *web apps*) as synonymous to refer to IT solutions can be accessed from browsers. While no formal definition differentiates the two terms clearly, most IT professionals see a difference between websites and web apps. Overall, websites can be seen as more informational-based web pages while web applications as more complex IT solutions that usually involve more interactions from the users. A typical example is a portfolio website of a freelancer in contrast to an e-commerce application such as Amazon[1]. The first website is mainly used to gather information, in contrast, the second allows the users to do more complex operations and actively participate in the interaction. On the other hand, this definition is limited and can fail to differentiate more borderline cases such as news websites, where the interaction from there is not much interaction from the user however, the architecture behind the scene might be complex. While we recognise the difference between websites and web apps, as mentioned above, we use both terms as synonymous.

2.1.1 Native versus Web Apps

Currently, there are 2.8 and 2.2 million apps on the Android and iOS stores respectively[2]. On a monthly basis, more than 60 thousand apps are added to the Google App Store alone[3]. Given these numbers, it is clear that there is an app for almost everything: from entertainment and e-commerce applications to social media and beer simulators[4].

Currently, **mobile native applications** are winning against web solutions. The majority of digital media access is performed on apps rather than browsers. On the other hand, users do not like to download new apps, and if they do, they tend to use only a small subset of the ones available on their device[5]. On average, users install from 60 to 90 apps on their smartphones but use only nine on a daily basis. Moreover, among these nine applications around five are usually already pre-installed on the device. Users feel overwhelmed by apps and the term *App Fatigue* became popular in the IT sector. If users do not quickly see value in an app, they will easily forget about it. In such an environment, developing a native app for an IT solution is not always the best alternative.

Browsers, such as Safari or Chrome, are one of the five applications already used on a daily basis. They are pre-installed on mobile devices, and therefore, they can be easily accessed. From the perspective of developers, building web applications is particularly beneficial since developers are not required to rebuild their IT solution for each platform. A web application has the potential to work on different combinations of devices and operating systems. For all these reasons, implementing a web solution as opposed to a native app is becoming increasingly popular among developers.

Initially, **hybrid mobile apps** were proposed as a good compromise between the cross-platform compatibility features of web applications and the visibility offered by app stores. These applications are usually built using web technologies, such as HTML,

[1]Amazon website: https://www.amazon.com. Accessed on April 2018.
[2]Number of apps available (2017): https://goo.gl/tCnPXW. Accessed on March 2018.
[3]App Stores Start to Mature (2016): https://goo.gl/UCBBkv. Accessed on March 2018.
[4]iBeer: https://goo.gl/1LTIlz. Accessed on March 2018.
[5]Countries Leading in App Usage (2017): https://goo.gl/6GntlC. Accessed on March 2018.

CSS and JavaScript, but they are later incorporated in Web Views: browsers wrapped in native apps. Similarly, tools such as Apache Cordova[6] and PhoneGap[7] allow developers to build native apps using only web technologies. In this way, developers do not need to implement their solution for all platforms, but they can build their application only once thereby reducing costs and time. While this solution offered several advantages, performance issues, as well as a number of other drawbacks, required a new set of smarter applications, namely **Progressive Web Apps** (PWA)[8]. PWAs offer the same features of common apps but they can be browsed as websites and do not need to be downloaded from the store.

Frances Berriman and Alex Russel proposed this new set of applications as a possible alternative to hybrid solutions. PWAs are not available through app stores, but once visited from the browser, they can be added to the home screen app or app launcher. Thanks to features newly added to most common browsers, Progressive Web Apps can support app-like functionality while preserving the advantages that the web offers. In fact, web applications can now push notifications, run in full-screen mode as well as be added to the home screen of the device (see Figure 2.1).

Berriman and Russel also defined a set of attributes of this new class of web applications. Among those, we have *responsiveness*, *connectivity independent* and *linkable*. *Progressive Web Apps* should adapt to the screen real estate of the client. Thanks to caching, they should be accessible if offline and finally, they should be easily shared via URLs while avoiding the additional overhead of being installed from stores.

Overall, Progressive Web Apps are faster and lighter than native apps; always up to date, they can work offline and on different platforms and finally, they are easy to be accessed and distributed. These features could drastically increase the overall engagement of users on a website. For instance, Twitter Lite is a successful example of a Progressive Web App. In their PWA solution, Twitter found a 75% increase of tweets sent by users, while requiring less than 3% of storage space than its native solution[9]. Similarly, the global media company Forbes[10] was able to double the time that users spend on average on their website thanks to a Progressive Web App[11]. Also eCommerce websites, such as AliExpress[12] and 5miles[13] are experiencing improvements with PWA solutions.

In **cross-device scenarios** when different and heterogeneous devices are used simultaneously, the multi-platform capability of web applications plays a key role [97]. A native solution requires the developer to first build an app for each possible platform and then, secondly, allow the communication among them. Since these platforms can differ significantly, building a native solution for cross-device scenarios can be particularly challenging. A web application can overcome these issues. As a consequence, PWA solutions and web applications in general, represent a good alternative to native apps from both user and developer perspectives for single as well as cross-device scenarios.

[6]Cordova website: https://cordova.apache.org. Accessed on March 2018.

[7]PhoneGap website: https://phonegap.com. Accessed on March 2018.

[8]Progressive Web Apps (2015): https://goo.gl/KIZydg. Accessed on March 2018.

[9]Twitter Lite PWA improvements: https://goo.gl/iRWyWu. Accessed on March 2018.

[10]Forbes website: https://www.forbes.com. Accessed on March 2018.

[11]Forbes PWA solution improvements: https://goo.gl/BcOjPg. Accessed on March 2018.

[12]AliExpress website: https://www.aliexpress.com. Accessed on March 2018.

[13]5miles website: https://www.5miles.com. Accessed on March 2018

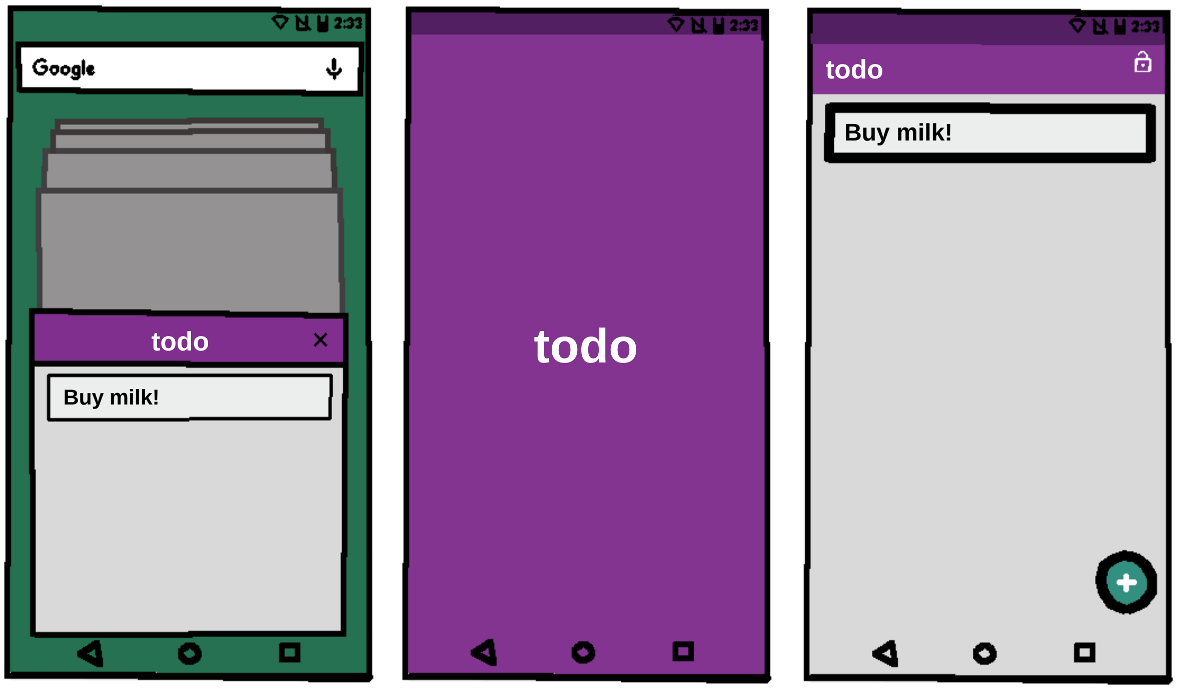


Figure 2.1: Example of a Progressive Web App for a to-do list scenario. Although the application is developed with web technologies, once added on the home screen the user can browse the app among all open applications and it will be displayed in a full-screen mode.

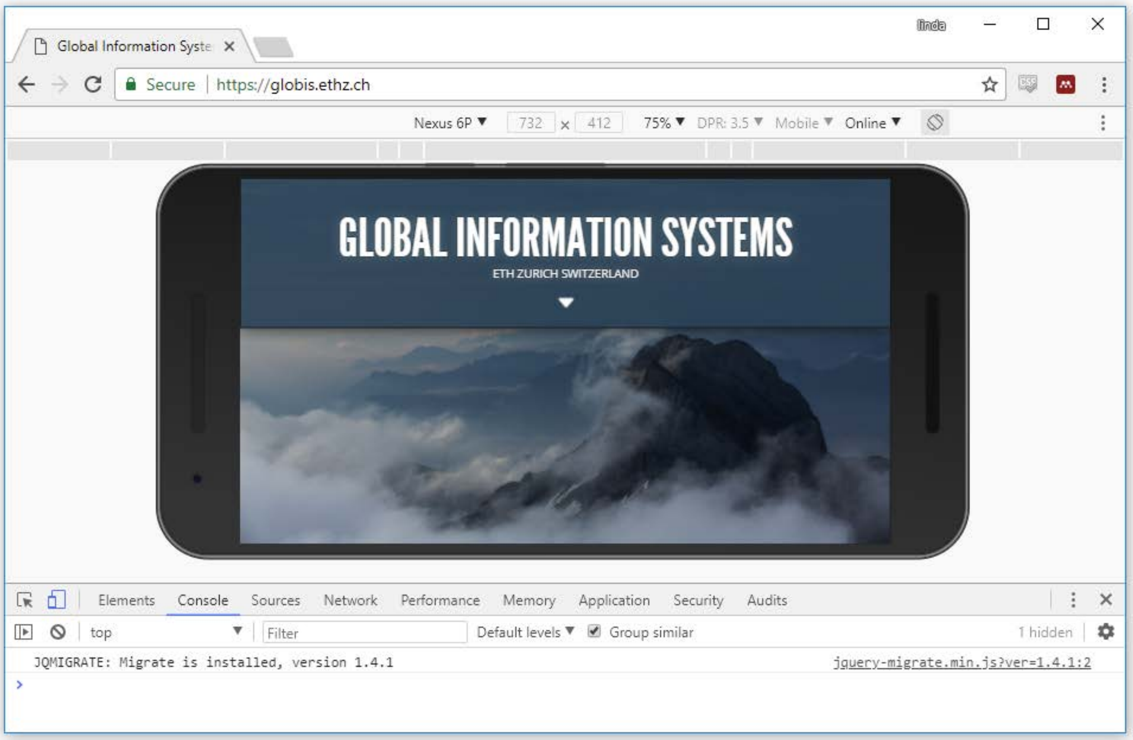
2.1.2 Development Tools

To better support developers and end-users, both researchers, and IT companies, have proposed a number of systems to facilitate the creation of web applications. IDEs and frameworks can improve the productivity of developers while allowing high-level APIs to experiment with new technologies. On the other hand, end-user development tools (EUD) can allow users to create their website without requiring any particular technical skill [128]. With the increased popularity of cross-device applications, researchers have also proposed frameworks and EUD systems to speed up the development process and make ubiquitous scenarios more prominent. Despite this growth of tools, the development of web applications still raises many challenges, especially when experimenting with alternative forms of interaction. In this section, we discuss these challenges while presenting systems that support the development of single and cross-device web applications. We start our discussion by first analysing frameworks and tools that target developers; before giving an outline of GUI and EUD systems for end-users.

Tools for Developers

From the developer perspective, most common browsers such as Chrome, Firefox and Safari have been extended with developer tools to inspect the DOM, run code in the console and simulate the responsiveness of a website (see Figure 2.2).

Media queries help developers to build responsive web applications. They allow the definition of styles depending on a boolean condition. Often, media queries are



```
JQMIGRATE: Migrate is installed, version 1.4.1        jquery-migrate.min.js?ver=1.4.1:2
```

Figure 2.2: Screenshot of the Google Chrome development tools.

used to define breakpoints on the width of the client and therefore, arrange the layout of the web page depending on the size of the device displaying the content. Recently, responsiveness was made even easier with the advent of Flexbox[14], the new layout mode available with CSS3. The overall goal of Flexbox is to improve the alignment of items on the page, their order and directions depending on the dimension of the viewport of the client. At the same time, developers now have also access to a multitude of frameworks to build their mobile-friendly web solutions. A typical example is Bootstrap[15], a front-end library built in CSS and JavaScript that can provide a good starting point for developers building a responsive web application.

Furthermore, editors have also been extended to improve productivity. IDEs such as Brackets[16], allow developers to push changes on the browser without refreshing the corresponding page; they also incorporate version control systems such as GitHub[17]. In literature, researchers have improved IDEs to open documentation pages, or code examples found online [190, 141, 126]. The goal of these projects is to avoid the back-and-forth between browsers and editors that usually developers do when building applications.

Despite the number of tools and frameworks available, building a web application is not free from issues. A significant issue concerns **cross-browser incompatibilities**. While many features have been added natively on most browsers, they are not all implemented in the same way by different platforms, or are only partially supported

[14]CSS Flexbox: https://goo.gl/VyX92u. Accessed on March 2018.
[15]Bootstrap website:https://getbootstrap.com. Accessed on March 2018.
[16]Brackets website:http://brackets.io. Accessed on March 2018.
[17]GitHub website:https://github.com. Accessed on March 2018.

[144]. As an example, Flexbox itself is not yet fully available for Internet Explorer[18]. Similarly, many media query conditions are not yet allowed by all browsers. The query `@media (pointer: type)` used to specify the presence and accuracy of the pointer of the device, it is not yet supported by Internet Explorer, Firefox and Opera[19]. Many other features are partially developed or in conflict with other implementations.

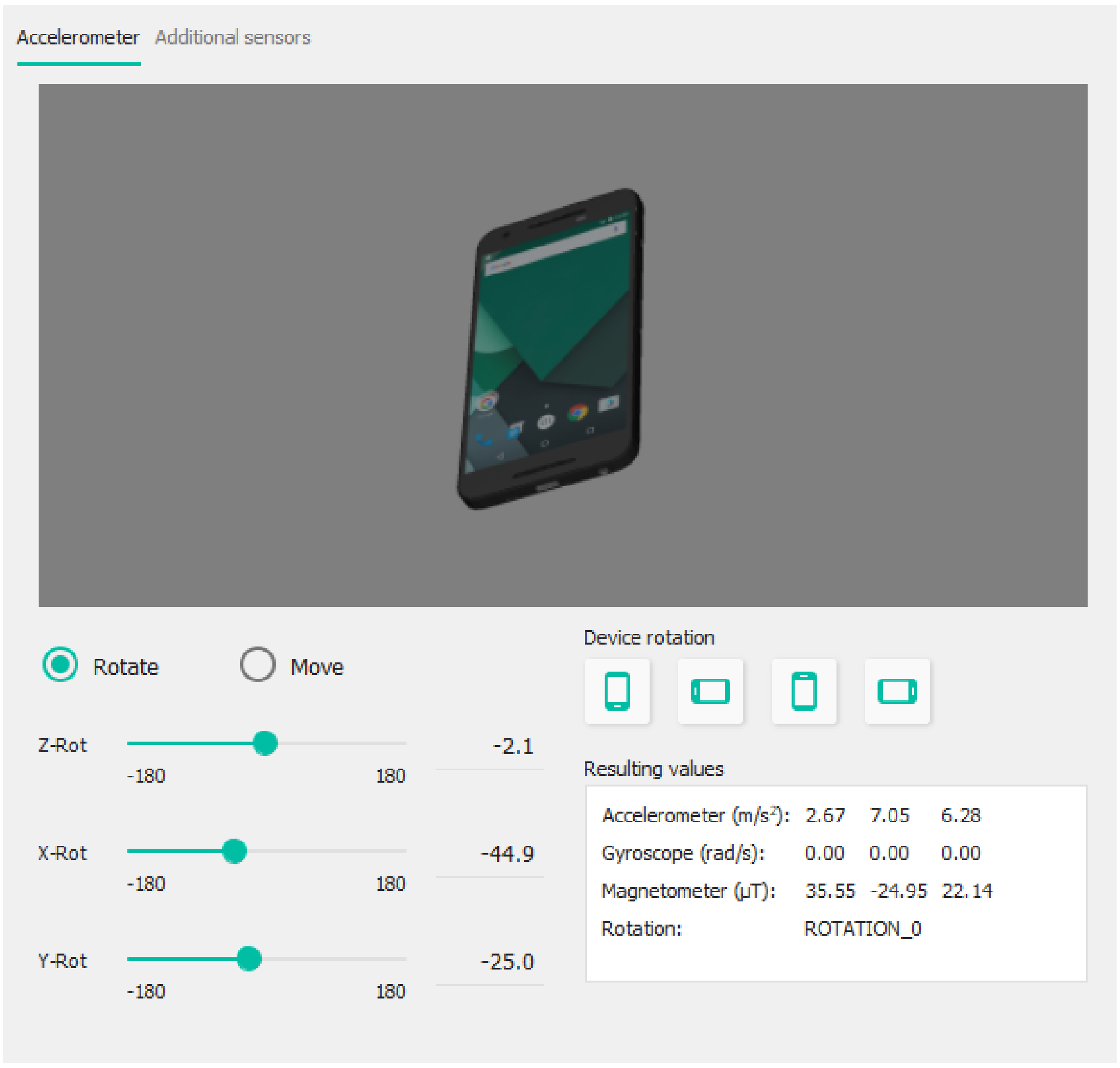
Concerning **interaction techniques**, cross browsers compatibility problems are even more drastic. Although touch is the primary form of interaction on mobile devices, browsers have different implementations of its event listener functions. For example, Internet Explorer employs *Pointer Events* instead of touch events and Edge requires developers to enable touch events by using a specific flag[20]. Moreover, some browsers are now improving performance over touch scroll events with *passive* event listeners. However, these changes were not made backwards compatible, causing problems with old websites[21].

Furthermore, motion events also suffer from browser compatibility problems. In detail, `DeviceOrientationChange` and `DeviceMotionChange` both return, the three absolute orientation angles (alpha, beta and gamma), and the acceleration of the device. With these raw data values, developers can experiment with motion interactions and allow users to perform gestures by simply moving their device. However, the only browser that fully supports them is Edge[22]. In the other browsers, the two events give different outputs. For example, the range of the orientation angles are treated differently on iOS and Android platforms. In browsers that run on iOS devices, the beta angle ranges from -180 to 180 degrees, while on Android it ranges from -90 to 90. In contrast, the alpha angle ranges from -180 to 180 on Android and -90 to 90 on iOS. Many other differences can be found among browsers and platforms. While developer tools to test and debug responsiveness are generally available on most browsers such as Chrome and Safari, they do not offer tools to check orientation events. In contrast, this feature is available for the development of native apps such as in Android Studio (see Figure 2.3).

To help developers deal with compatibility issues, researchers have proposed IDE extensions that check if the code will work among browsers and to which extent [144]. jQuery or JavaScript frameworks such as jQMultiTouch [151], Hammer.js[23] and jQuery-Mobile[24], also help the developer to use touch and multi-touch interactions without having to deal with implementation differences among browsers. Other plugins, such as Gyro.js[25], also support developers in dealing with browsers differences for motion events. However, while Hammer.js and jQueryMobile support high-level APIs for touch gestures, frameworks for motion events do not offer gestures but rather return raw data more consistently. Developers therefore are responsible for using the events fired by the sensors to implement motion interactions by themselves.

We also note that researchers have proposed systems for detecting compatibility is-

[18]Can I Use Flexbox: https://caniuse.com/#feat=flexbox. Accessed on March 2018.
[19]Can I Use @media pointer: https://caniuse.com/#search=pointer. Accessed on March 2018.
[20]Can I Use touch events: https://caniuse.com/#search=touch. Accessed on March 2018.
[21]Forum discussions on passive scroll events: https://goo.gl/kts9yT. Accessed on March 2018.
[22]Can I Use orientation: https://caniuse.com/#search=orientation. Accessed on March 2018.
[23]Hammer.js website: https://hammerjs.github.io. Accessed on March 2018.
[24]jQuery mobile website: https://jquerymobile.com. Accessed on March 2018.
[25]Gyro.js website: https://goo.gl/GW6ncB. Accessed on March 2018.



Rotate [✓] Move

Z-Rot -2.1

X-Rot -44.9

Y-Rot -25.0

Accelerometer (m/s²):	2.67	7.05	6.28
Gyroscope (rad/s):	0.00	0.00	0.00
Magnetometer (µT):	35.55	-24.95	22.14
Rotation:	ROTATION_0		

Figure 2.3: Screenshot of the orientation tools in Android Studio.

sues by comparing either the DOM of the page or how they visually appear on different browsers [184, 41, 42]. Similarly, BrowserStack[26] and Browsera[27] are commercial services that run the web applications on different combinations of browsers, devices and platforms to see if there are any differences in the appearance of the page. However, it is clear that these tools cannot easily check differences in the interaction techniques supported by the device. Instead, systems such as Selenium[28] allow the record and replay of mouse, keyboard and touch events for testing purposes. Hesenius et al. [89] go a step further and allow the automation of events on mobile devices for complex high-level touch gestures for native apps. For motion interactions, Hartmann et al. [84] propose a system that enables the authoring of motion gestures by demonstration. With Exemplar, developers can demonstrate the gesture by first performing it and then editing the resulting raw data from the sensor via a visual tool. Finally, the user can repeat the

[26]BrowserStack website: https://www.browserstack.com. Accessed on March 2018.
[27]Browsera website:http://www.browsera.com. Accessed on March 2018.
[28]Selenium website: https://www.seleniumhq.org. Accessed on March 2018.

action to test the gesture performed. While Exemplar could help developers in building tilting interactions, they are still required to work in a low-level fashion by studying raw data fired by the sensor.

On the web, researchers have also promoted **model-driven approaches**for supporting the development of web applications [65, 1, 32]. With model-driven techniques, developers define the model underneath the application and, from this model, the systems will automatically generate the websites. For instance, with WebRatio[29], users can define data using the entity-relationship model and define the functional requirements of their web application with WebML [32], a specific graphical language, is used to define the functional requirements of their web application. Model-driven approaches have also been extended to work on smaller devices and support responsive web design [33]. Although the research community has spent a lot of effort in developing and studying model-driven approaches, they did not become widely used among developers. Moreover, this technique also fails in addressing alternative forms of interactions on the web.

For **cross-device web applications**, technologies such as Socket.IO[30] allow developers to create real-time bi-directional communications among clients and a server using JavaScript. In contrast, technologies such as Peer.js[31] allow peer-to-peer connections among clients without the need of a central server with the goal of reducing latency. While building cross-device applications on the web can solve interoperability problems and easily allow developers to experiment on a large combination of platforms and devices, cross-device applications present additional challenges that are not present in single device web solutions [61]. For these reasons, researchers have proposed APIs and tools to improve all phases of the development process [100, 193, 13].

For instance, XD-MVC offers a set of APIs for easily sharing messages among clients easily [100]. XD-MVC includes a Node.js[32] server and uses Socket.IO to allow a client-server architecture and Peer.js to support, peer-to-peer communications when available. Without the guidance of XD-MVC, developers would be required to manage synchronisation, device pairing and device identification issues manually. In XD-MVC the state is kept in sync by updating the view of each client only when the model is changed. Devices can start communication with other clients by different supported pairing techniques, such as typing URLs or using QR codes. Moreover, developers can easily identify clients by an ID generated at the moment of the first connection.

Another critical aspect in cross-device applications is the **adaptation of UI elements** to clients. Researchers have studied different mechanism for deciding how to organise the UI in cross-device applications and therefore help developers in making these decisions. Some approaches require the assignment of roles to each device and arrange elements depending on this factor [70, 100, 97]. For instance, if a client was defined as the *controller* of the communication it will have additional UI elements to, for example, play the next video, or show the next picture, as well as add new devices to the communication. Alternatively, Part et al. [167] solve the assignment of elements to devices as an optimisation problem. Their system takes into consideration roles, user preferences and device characteristics (e.g. an element that requires text input

[29]WebRatio website: https://www.webratio.com/. Accessed on March 2018.
[30]Socket.IO website: https://socket.io. Accessed on March 2018.
[31]Peer.js website: http://peerjs.com. Accessed on March 2018.
[32]Node.js website: https://nodejs.org. Accessed on March 2018.

will be likely assigned to a device with a physical or software keyboard) to dynamically adapt content on different devices. Developers can change parameters of each element by using an IDE extension called AdaM.

Similarly to responsive design, these approaches support developers in how to best organise visual elements on different devices. However, they do not consider user interactions. Building gestures for single device web applications can be challenging, and, in cross-device scenarios, these issues became particularly problematic since interactions have to be managed, sent and received on a multitude of heterogeneous devices. To avoid these problems, researchers have built frameworks and visual tools to support the rapid development of **cross-device interaction techniques** [39, 16, 131, 196]. Chi et al. proposed Weave [39], a set of high-level web APIs to support touch and rotation events across devices. Via an authoring environment, developers can also easily edit, test and debug interactions and their outcomes on smartphones, tablets or smartwatches. Similarly, ATREUS offers APIs for touch and rotation events fired by handheld devices [16]. With ATREUS and Weave, developers do not need to manage the interactions on each device and then send data to the desired targets. The frameworks will be responsible for the recognition of the gestures as well as identifying the sender and receivers of the interactions. Although these works were of great inspiration, they do not offer a wide variety of tilting gestures or other forms of interaction.

Tools for End-users

In 2006, Lieberman et al. [128] stated that over the years, the goal of interaction systems would move from making systems easy to use to making systems easy to develop by end-users. To tackle this topic, authors suggested a new multidisciplinary paradigm called end-user development (EUD) and defined it as follows:

> *"EUD can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact".*

On the web, researchers have promoted **mashups** as a possible way of developing web applications without coding [224, 20, 2]. A web mashup is a web application generated by mixing and mashing UI elements, layouts, content or functionality of different web sources. A famous example of a mashup tool was Yahoo Pipes, a system that allowed end-users to aggregate and filter web feeds (*pipes*) and web pages via a user interface. For instance, users could create feeds on a specific argument such as tech news, sports or photography from a number of different websites that would act as the input of the aggregated web application. Yahoo Pipes was powerful and could be used for different purposes; however it still required some technical knowledge from the users. For this reasons, researchers have also experimented with more direct ways of manipulating and generating websites [74, 85]. Ghiani et al. [74] allow users to mix mashup components from arbitrary websites by direct GUI manipulations. Similarly, Hartmann et al. [85] proposed d.mix to support users to design by examples. The tool allows users to browse websites and select elements to sample that can later be edited and mixed with components gathered from other web applications.

Despite the fact that these methods can potentially allow a larger number of non-technical users to create their web applications, mashups had limited success over the

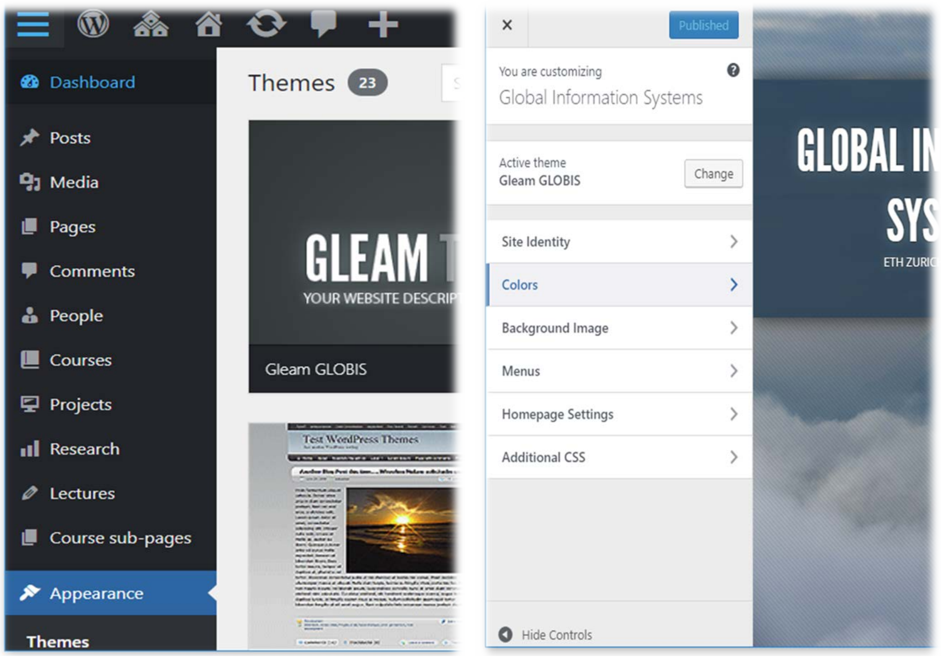


Figure 2.4: Screenshots of the WordPress Dashboard. On the left its main page where users can access posts, pages, plugins and also pick the desired themes. By clicking on the customise option in the appearance section, the page on the left will be shown. From here, users can change styling as the colour and picture of the background as well as modifying menus.

years. Nowadays, the vast majority of practitioners, as well as professionals, prefer to use **content management system (CMS)** platforms such as WordPress[33] to develop their websites. Currently, 30.2% of the top 10 million websites are developed using WordPress[34]. Overall, the latest version of WordPress was downloaded over 57 million times from wordpress.org[35] and many important companies, such as The New Yorker[36], Sony[37] and TechCrunch[38] have decided to build their websites on top of the CMS.

With WordPress, users can download and dynamically customise crowdsourced themes to meet their needs. A theme defines the look and feel of the web application, and it also allows users to customise its features via the dashboard, an administration visual interface available for any WordPress web application (see Figure 2.4). By default, most themes are responsive and adapt to the screen real estate of clients. Via plugins, users can further extend their web applications by adding functionality that is not already built into the theme. Users can currently download more than 54 thousands plugins available from the WordPress official page[39]. As found by a survey on web development practices that we carried out in our group [160], many developers that work with CMSs have little or no formal computer science education and are self-

[33]WordPress website: https://wordpress.org. Accessed on March 2018.
[34]W3C Technology survey: https://w3techs.com. Accessed on March 2018.
[35]WordPress Counter: https://wordpress.org/download/counter. Accessed on March 2018.
[36]The New Yorker website: https://www.newyorker.com. Accessed on March 2018.
[37]Sony music website: https://www.sonymusic.com. Accessed on March 2018.
[38]TechCrunch website: https://techcrunch.com. Accessed on March 2018.
[39]WordPress plugins repository: https://wordpress.org/plugins. Accessed on March 2018.

employed or work in small organisations. For these reasons, the flexibility and usability of systems like WordPress meet users needs. Given the popularity of WordPress, we developed a tool for the rapid development of web applications on top of the CMS platform [161, 51]. More in detail, the tool allows users to mix and match layout and functionality components of WordPress themes.

Recently, systems like Wix[40], Weebly[41] and Squarespace[42] have appeared and are now heavily advertised online. These tools allow users to create and customise their websites by editing or moving visual elements via drag and drop interactions.

Despite the fact that systems such as WordPress and Wix allow non-expert users to easily develop websites, they mainly focus on the layout and styling of web pages. The adaptation of web applications on phones or tablets strictly relies on the visual adaptation of content rather than its modes of interaction. Moreover, building a web application via tools such as WordPress and Wix can still be challenging for end-users. There is a trade-off between the amount of customisation possible on a theme and the complexity of applying them. While setting up a generic website is usually straightforward, personalizing content and adding additional features often requires the user to perform more complex and repetotove operations. For example, websites such as ThemeForest[43], allow users to buy and download thousands of highly customisable WordPress themes. Although these are advertised as easy to use, these themes have steep learning curve and they hard to personalise by users that have little or no technical knowledge of web development. Paradoxically, themes from ThemeForest can be more useful for developers to quickly build web applications for external clients rather than used by clients themselves.

On **cross-device web applications**, researchers have presented tools to support end-users in distributing user interfaces among different devices [46, 118, 154]. With XD-Studio, Nebeling et al. [154] present a GUI builder to support the development of cross-device web applications. Users can assign UI elements of an existing website to different profiles by drag and drop operations. Profiles can be defined depending on the device types and user roles. For example, in a meeting room scenario, users can split the user interfaces to give different information and controls to devices present in the room. A big screen will project the current slides, the audience can ask questions from their mobile device, and the presenters can control the slides and see questions from their phones.

Researchers have also applied mashup concept for the rapid design and development of cross-device applications [118, 101]. With MultiMasher, Husmann et al. [101] present a visual tool that allows the reuse of existing web applications and direct GUI manipulation to create a final mashup of web solution that runs and can be controlled across multiple devices. We also note, that the pipeline concepts has also been used for the rapid design of **cross-device interactions** [117]. Squidy offers a visual tool for the rapid design of interactions from heterogeneous devices such as touch, multi-touch, pens and laser point inputs. Also in this context, the tools available are usually limited and do not offer a broader set of motion interactions.

[40]Wix website: http://www.wix.com. Accessed on March 2018.
[41]Weebly website:http://www.weebly.com. Accessed on March 2018.
[42]Squarespace website: https://www.squarespace.com. Accessed on March 2018.
[43]ThemeForest website: https://themeforest.net. Accessed on April 2018.

2.2 Beyond Mobile Touch Interactions

Many researchers have studied ways of exploiting the hardware sensors integrated in mobile devices to go beyond simple touch-based interaction and enlarge the set of possible gestures on phones and tablets. With the same goals, researchers have also explored the use of wearable devices, such as wristwatches or armbands to interact with a smartphone. In this section, we give an overview of alternative means of interaction while comparing them to touch gestures. Given their advantages, we focus on two main forms of gestures: motion and mid-air interactions. We categorise motion gestures as interactions detected by the movement of devices such as phones or tablets and usually recognised by accelerometers and gyroscope sensors.

It is important to note that we use the terms *motion* and *tilting gestures* as synonymous to refer to interactions characterised by the orientation and speed of a device. Similarly, to avoid repetitions, the terms *interaction* and *gesture* are often used as interchangeable terms.

In Section 2.2.1, we present such interactions by discussing the most relevant works studied in research. We define mid-air interactions as gestures performed with the user's fingers, hands or arm in front of a camera or detected by a wearable device worn by the user. We discuss these gestures in Section 2.2.2.

2.2.1 Motion Gestures

Motion sensors on mobile devices have commonly been used to recognise if the device was in landscape or portrait mode to adapt the content displayed on the screen accordingly [93]. While it is easier to interact with a smartphone in portrait mode, holding the devices in landscape offers a better user experience when viewing movies or pictures. Although these adaptations might be one of the many reasons why the first iPhone became so popular, the potential of these sensors goes beyond the recognition of these two states.

Motion sensors, such as accelerometers and gyroscopes, measure the acceleration and rotational forces along the three-dimensional axes. When integrated into a mobile device, such as smartphones or tablets, they allow the recognition of the current orientation of the device as well as its applied acceleration (see Figure 2.5). Given these features, researchers and, more generally, developers have used motion sensors for a wide variety of scenarios on mobile devices.

For instance, we can find applications of this technology in many fields such as activity recognition systems, digital forensic investigations and interaction techniques. By studying the output of these sensors, we can infer the context of a device. Also in combination with GPS data, researchers have been to detect the current mode of transportation of users e.g., cars, buses or trains [88], if they are writing on a keyboard near to the device [134] or if they are running or walking [17, 25, 125]. Fit applications, such as FitBit[44] or Runnastic[45], exploit these sensors to better guide the users when doing physical activities. In a digital investigation, these contexts can be used to show evidence of a crime [145].

[44]FitBit website: https://www.fitbit.com. Accessed on February 2018.

[45]Runnastic website: https://www.runtastic.com. Accessed on February 2018.

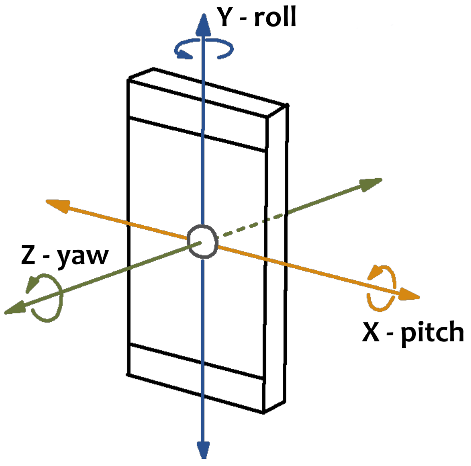


Figure 2.5: A mobile device in correspondence to the three-dimensional axis.

Accelerometer and gyroscope sensors have also been studied as an alternative way of interacting with mobile devices. Thanks to motion gestures, users can perform actions on a phone by simply moving it in some direction. In the last two decades, tilting interactions found popularity in the research community for the number of advantages they can offer. Given their nature, they are eyes-free one-hand gestures [15, 163, 156]. Moreover, users can easily move their device without changing the position of the hand holding the phone. These interactions can also provide an intuitive alternative when touch is not feasible due to gloves or dirty fingers [15, 18]. Furthermore, in contrast with touch, users can interact with the phone by means of tilting gesture without occluding the screen [221, 96].

In 1996, eleven years before the advent of the first iPhone, Rekimoto [179] was one of the first to suggest the use of motion gestures to improve the usability of PDA[46] (Personal Digital Assistant) phones, the ancestor of the mobile devices we use every day. In his prototype, he attached position and orientation sensors and two physical buttons to a small LCD screen to simulate a handheld device. He applied tilting interactions to navigate cylindrical or pie menus, a map and a 3D object. In the menu applications, users could start selecting items by first pressing the button on the top of the screen and then moving the device on the horizontal axis. When the desired item was in the centre of the screen, users could depress the button to select it. Similarly, by moving the device vertically or horizontally, users could browse a map or a 3D object.

Rekimoto also stated that several alternative designs could be proposed for each of the scenarios that he presented. With alternative implementations of tilting interactions in the same scenario offering different user experiences. Inspired by these first experiments with tilting gestures, and its many research possibilities, researchers have further explored the use of motion gestures on mobile devices.

By studying related works, we found that the most popular types of motion gestures are jerk and continuous interactions (see Figure 2.6). **Jerk tilting gestures** are fast movements of the device in some direction [15, 217, 186]. Continuous interactions, as studied by Rekimoto [179], allow users to, for example, scroll a list of items by slowly moving their phone [179, 40, 212].

[46]The history of Personal Digital Assistants: https://goo.gl/Le3dWJ. Accessed on February 2018.

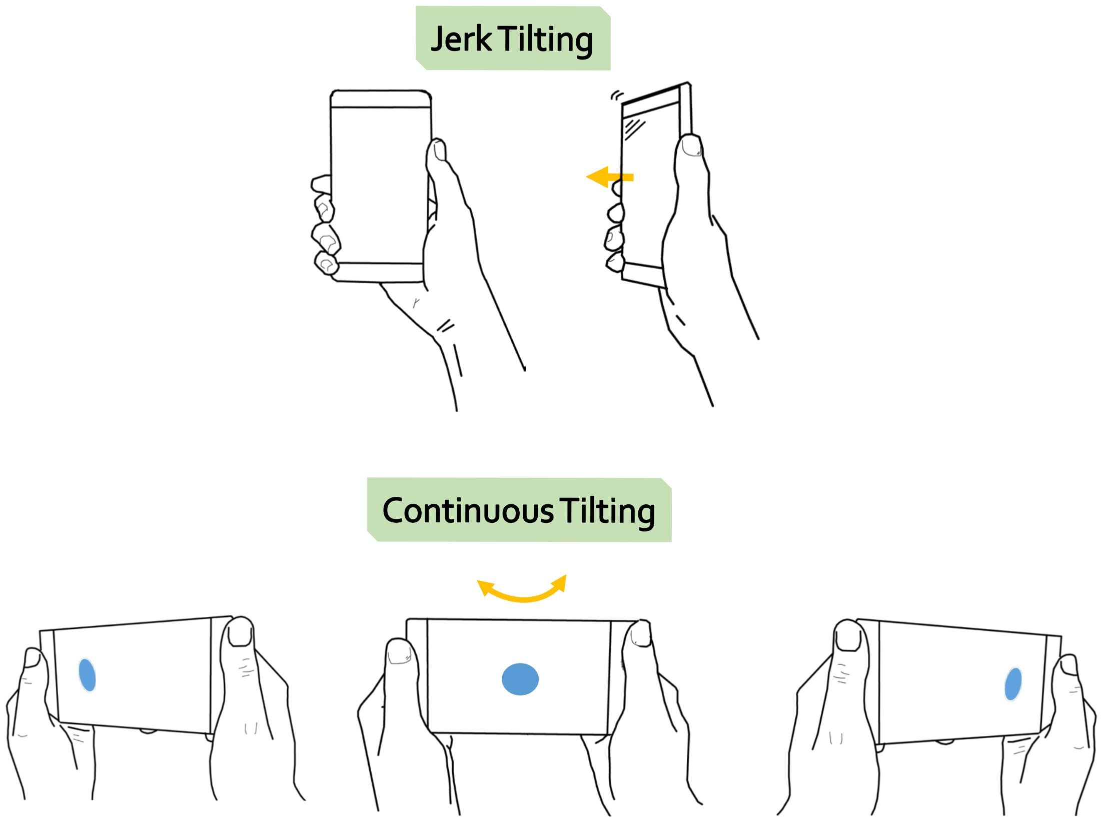


Figure 2.6: Abstract representation of jerk and continuous tilting gestures. By means of jerk interactions, users quickly tilt the device in some direction (to the left in figure) to perform the gesture. Instead, with continuous tilting interactions, the position of a pointer on the device (a blue circle in figure) can be controlled by continuously moving the device.

With JerkTilts, Baglioni et al. [15] propose new motion gestures that allow the user to interact with the device by quickly moving it back and forth on the pitch and roll axes. For example, the user can raise the volume of the device by tilting it up and then down, or stop playing the current song by performing a tilt to the left and then to the right. As stated by the authors, a positive characteristic of these gestures is the fact that they rely on the natural elasticity of the wrist. When the user tilts the device in some direction, a "recoil" factor will make the user perform a motion interaction in the opposite direction [79]. After a number of lab experiments, Baglioni et al. [15] concluded that their implementation of jerk tilting interactions was hard to produce inadvertently and they were as accurate as swipe gestures [15]. While jerk tilting interactions were slightly slower than touch, the authors discuss that motion gestures will not replace touch interactions entirely but rather enlarge the set of input capabilities of a mobile device.

Ruiz et al. [186] also came up with a similar conclusion: motion gestures can be used as an additional and alternative way to simplify some interactions, especially when touch is not feasible. They made this conclusion after discussing with participants of their elicitation study where the majority of users stated that they would like to use motion gestures in the future at least occasionally. In their experiment, participants came up with a set of tilting gestures to perform a series of everyday tasks they usually do on mobile phones. Among the gestures suggested by users, we can find many tilting interactions similar to jerk gestures. For example, users suggested to quickly move the phone to the right and to the left to go back to the home screen because, as stated by one of the participants, shaking the device feels like starting over.

Despite the amount of kinetic impulse necessary to perform the gesture varying among different works, similar implementations of jerk tilting interactions have been used for different applications. For example, researchers used jerk tilting interactions as an alternative way for inputing text [217].

In the TiltText project, Wigdor and Balakrishnan [217] built a low-cost tilt sensor on the back of a Nokia 5510 to improve the overall user experience when writing messages. As typical for mobile phones during that period, the device presented a T9 QWERTY physical keyboard that required the user to click several times on the same key to select a symbol. To improve the interaction, authors used the sensor built on the phone to allow users to browse among characters by clicking on the desired key and then tilting the device in four different directions (left, forward, right and back). A user study conducted by authors proved that even though TiltText was more prone to errors, it was substantially faster than tapping multiple times on the same key.

Despite the advent of more sophisticated mobile devices that are capable of detecting touch, inserting text on phones still poses many challenges. On small phones, it is easy to tap the wrong key or tap multiple keys at the same time, while on bigger mobile devices or tablets, it is hard to reach symbols with one hand [221]. For these reasons, researchers have suggested motion sensors as a good alternative. Many exploit the use of **continuous tilting gestures** in combination with jerk motions, to browse and select characters to input text [221, 212].

For example, with Rotex [212], users can rotate the device in one dimension (roll axis) to select letters and perform various jerk gestures to insert special characters such as a space or new line. Instead, Yeo et al. [221], extend the shaping writing technique with tilt interactions by proposing SWiM (Shape Writing in Motion). With SWiM, users first tap on a specific portion of the screen to active tilting gestures, then by moving the device they *draw* the shape with a cursor to select characters on the soft keyboard.

In both cases, there is a direct mapping between the orientation of the phone and an item. In Rotex, every rotation angle of the phone corresponds to a specific letter, while in SWiM, the angle will infer the position of the cursor on the keyboard.

In the literature, this type of continuous interaction is often referred to as ***position-based*** solutions (see Figure 2.7). This type of gesture is not the only alternative to of continuous tilting. Researchers have also proposed ***velocity-based*** variants [40, 205, 67]. In this case, instead of mapping a specific orientation to an item or a position of a cursor on the screen, the velocity and angle of the device are used like a pedal of a car. In *velocity-based* solutions, the more the device is tilted, the faster the system will, for example, scroll through a list of items.

Researchers have applied both alternatives to many different applications such as browsing a map [209, 210], selecting items in one or two-dimensional menus [205, 177, 162, 163] or moving 3 or 2D objects in games [36, 159, 75]. Position-based solutions were overall better perceived by participants when selecting items in a menu [162, 205]. As also found by Ruiz et al. [186], a *velocity-based* solution could give less control and

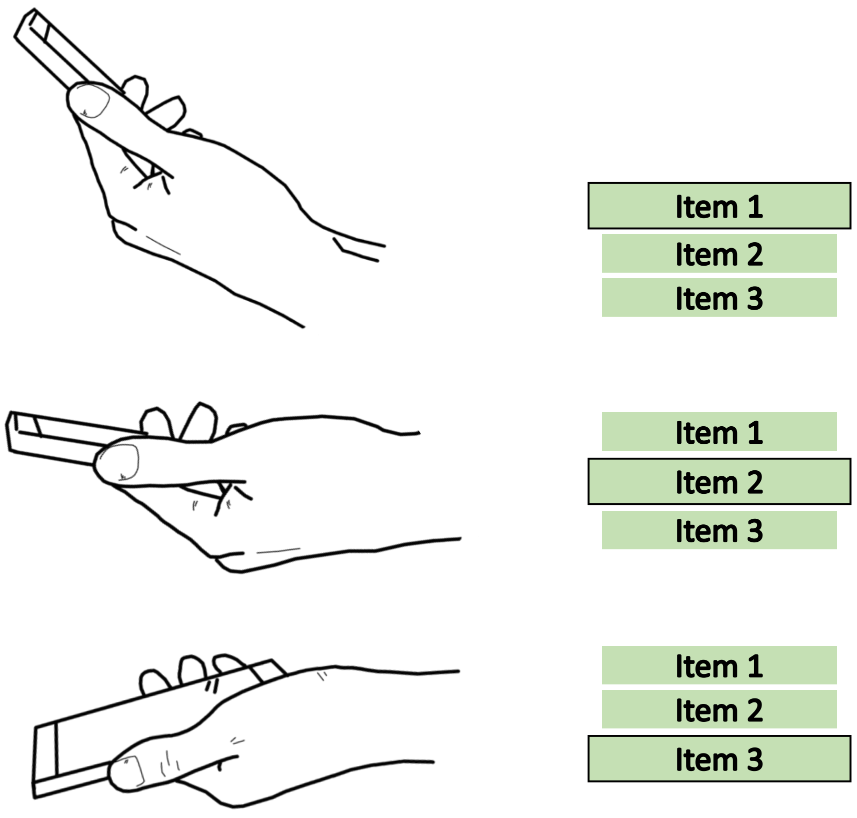


Figure 2.7: Abstract representation of *position* based continuous tilting interactions to select an element in a menu. Every element is mapped to a specific range of orientation angles of the device. In this case, the angle on the yaw axis will infer the item.

therefore, frustrate the user. However, given the nature of *position* based implementations, there is a limit on the number of items that we can associate with orientation angles.

Rahman et al. [177] found that a user can easily control a maximum of 12 levels of pronation/supination (wrist movements on the roll axis) and 8 levels on the flexion/extension (wrist movements on the pitch axis). For these reasons, to browse a broader set of items, position-based implementations could not be sufficient. To overcome this problem, Cho et al. [40], proposed a more sophisticated solution for browsing a gallery of pictures by means of continuous tilting interactions. In their approach, they extended *velocity-based* implementations with the concept of attractors placed in the centre of every image in the gallery. When the cursor guided by the rate of tilt of the device reaches the centre of a picture, its velocity decreases to lower the chances of overshooting. In fact, their approach performed better regarding the number of errors when compared to more classical implementations of continuous tilting interactions. However, the attractor solution was not as efficient as button interactions, where users could browse the gallery by pressing buttons on the phone.

Despite the effort of developers, tilting interactions could still suffer from overshooting problems or from false positives. Users could involuntarily trigger gestures while moving the device when talking or on cars or public transportation. To overcome this problem, researchers have **combined tilting interactions to touch gestures**. For example, Hinckley and Song [92] suggest that users could zoom in and out by performing a hold tap on the element and then tilting the device forward or backwards. Similarly, users can delete pictures by a hold tap on the image and then shaking the phone.

The combination of touch and tilting interactions affects some of the advantages of motion gestures; users are required to interact with the touch screen directly and therefore, occlude parts of the screen real estate. However, the gestures can still be performed in an eyes-free manner since the touch interaction could happen everywhere on the screen. Alternatively, input delimiter can be used to distinguish motion gestures from random movements of the device. For example, Ruiz and Li [185] suggested a double flip gesture (rotation of the phone on its back and then up again) to better recognize the motion interactions that the user will then perform.

Another important role in motion interactions also relies on informing the user that the gesture has been performed. Visual, audio and vibration **feedback**, or combinations of those, can be used to confirm that the system successfully recognised the gesture. As found by researchers [142, 93, 64], these clues are a relevant factor and can improve the overall user experience of tilting interactions.

We also note that researchers have studied tilting gestures to better allow visually impaired users to interact with mobile devices [58, 59, 183]. Given the eyes-free nature of motion interactions, tilting gestures can be highly suited for disabled users when performing everyday actions on phones. In a study conducted by Dim and Ren [58], blind and blind-folded users were more efficient in making phone calls or browsing the list of contacts with motion gestures than with traditional (physical) button interactions.

2.2.2 Mid-air Gestures

As stated by Caramiaux et al. [30], gestures are a complex notion used across different fields. In psychology, Efron studies how speech-accompanying body movements are influenced by the background of the user [62]. For example, in the Italian culture, gestures are an essential extension of the language where hundreds of symbolic gestures are commonly used during conversations [112, 173]. In HCI, Kurtenback and Hulteen defined mid-air gesture as "a movement of the body that contains information" [123]. In this context, mid-air gestures are deliberate body interactions that can be performed independently from speech. The overall goal of these gestures is to execute commands on a system in a natural and unobtrusive way.

Also influenced by the studies conducted by Efron and other similar works [63, 68], McNeil [138] proposes four categories to describe different forms of mid-air gestures for HCI purposes: *beat*, *deictic*, *iconic* and *metaphoric* (see Figure 2.8).

Beat interactions are usually composite gestures (at least two fast repetitive movements). For example as we can see from Figure 2.8, a tap of the finger up and down on a desk is a typical beat gesture. *Deictic*, or pointing interactions, are gestures used to refer to an entity such as an object the user refers to to while talking. Their meaning can vary depending on the value of the pointed region. *Iconic* interactions have a close relationship with what the user is currently saying verbally. As examined by McNeil, the utterance "he tries going up inside the pipe [...]" can be accompanied by an iconic gesture of the hand rising upward. Similarly as we can see from Figure 2.8, users can express rough measurements of objects using their hand. Finally, *metaphoric* classes

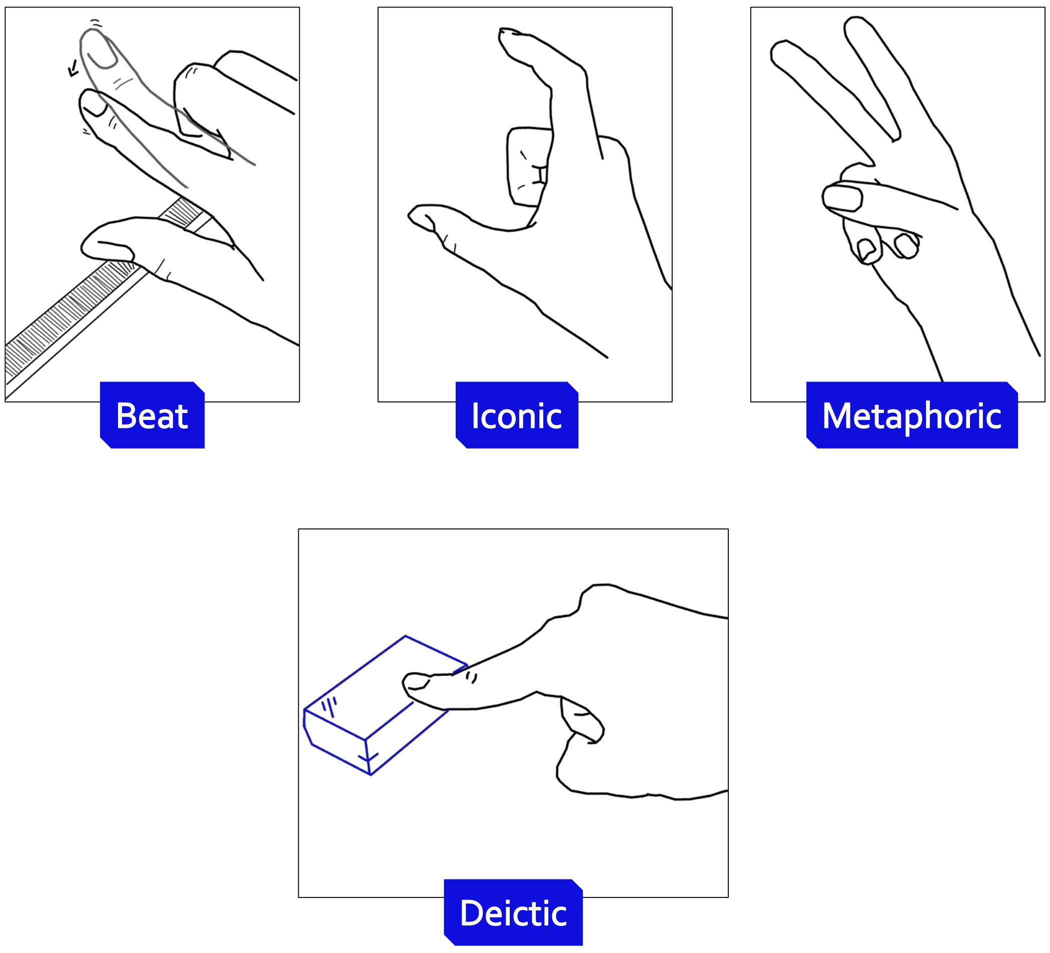


Figure 2.8: Examples of mid-air gestures that follow the taxonomy defined by McNeil [138].

comprehend all those gestures that users perform when describing an abstract concept. This category is similar to *iconic* interactions; however, they do not represent something concrete such as an event or an object. As represented in Figure 2.8, the V sign of the index and middle finger can communicate the "Victory" message.

Over the years, researchers have extended or renamed McNeil classes to categorise mid-air interactions. For example, in the literature we can find the term *pantomimic* to discuss a similar form of *metaphoric* gesture [175, 78]. In this class, users mimic with their body an action to perform a task such as answering a call by moving an imaginary phone near to the ear. Furthermore, authors have often used the term *symbolic* to refer to gestures that represent a symbol such as drawing a question mark in the air [3, 213, 8]. Moreover, gestures can also be static or dynamic [110]. A static gesture is a fixed configuration of the users' hands, fingers or arms. For instance, poses are a classic example of static mid-air interactions [200, 104, 201]. On the other hand, dynamic gestures are continuous movements and they are characterised by three phases: prestrike, stroke and poststroke [140]. Movements of the users' hands along the vertical or horizontal axis or drawing circles in the air, are two types of gestures that fall in this category [147, 116, 35].

Social acceptability is an important point to take into consideration when using and studying mid-air gestures. Performing new and, sometimes, extravagant gestures in public could make the user feel uncomfortable. As found by Rico and Brewster [182], users felt that private settings were better for performing a set of mid-air gestures proposed by the authors. However, they discuss that this *awkwardness* could improve over time. Some participants stated that after multiple trials, they felt more comfortable doing the gestures because they know what to expect. Authors also found that more subtle interactions that look or feel similar to everyday actions are more comfortable to perform. Small movements could also avoid the problem of user **fatigue** [90]. Excessive movements of the hands or arms could make the user easily tired and, therefore, influence the overall experience of systems that exploit mid-air gestures.

Besides the form of the gestures, there are other dimensions with which we can classify these interactions. Karam and Schraefel defined three other factors to categorise mid-air as well as other HCI gestures [110]. First, how the system acts once it recognises the interaction. Authors found that the most popular outputs in literature were audio, visual and CPU commands. Second, the technologies that allow the detection of the gesture. The technology can be non-perceptual or perceptual. With non-perceptual technologies, the user needs to physically interact or wear devices or objects to input the gesture. In the case of mid-air gestures examples of non-perceptual technologies are wearable devices. On the other hand, perceptual technologies do not require the user to interact with physical objects, but they can freely perform the interaction via, for example, remote sensing or visual detection techniques. Finally, the third dimension relies on the application domain within which users can exploit gestures. Examples are desktop applications, mobile and pervasive approaches, games etc.

Given the focus of this section, and this thesis in general, we will mainly discuss related works that study mid-air gestures in mobile application domains while proposing different outputs and detection sensing techniques.

The overall goal of using mid-air gestures on mobile applications is to enlarge the set of possible interactions of phones while avoiding occlusions caused by the fingers touching the screen [104, 120, 149]. In the next two sections, we discuss works that exploit perceptual and non-perceptual technologies for mobile interaction purposes.

Perceptual Technologies

Many researchers have used image recognition systems to detect movements of a users' hands or fingers. Some have used external cameras to detect mid-air gestures. In these works, users do not directly interact with additional hardware such as pens or wearables, but the recognition is performed thanks to special cameras able to detect the users' gestures [10, 200, 87]. With AirPanes, Hasan et al. [87], allow users to perform two-handed interactions to browse a large data set of information. For example, users can select items displayed on the phone by moving and pointing their fingers in a 3D area near to the device. To better detect the interaction performed, authors exploit the use of eight tracking cameras built in the room and markers on the phone and the users' fingers.

To achieve a more compact solution, researchers have built additional cameras directly on the handheld device [158, 120] and some suggested the use of the Leap Motion

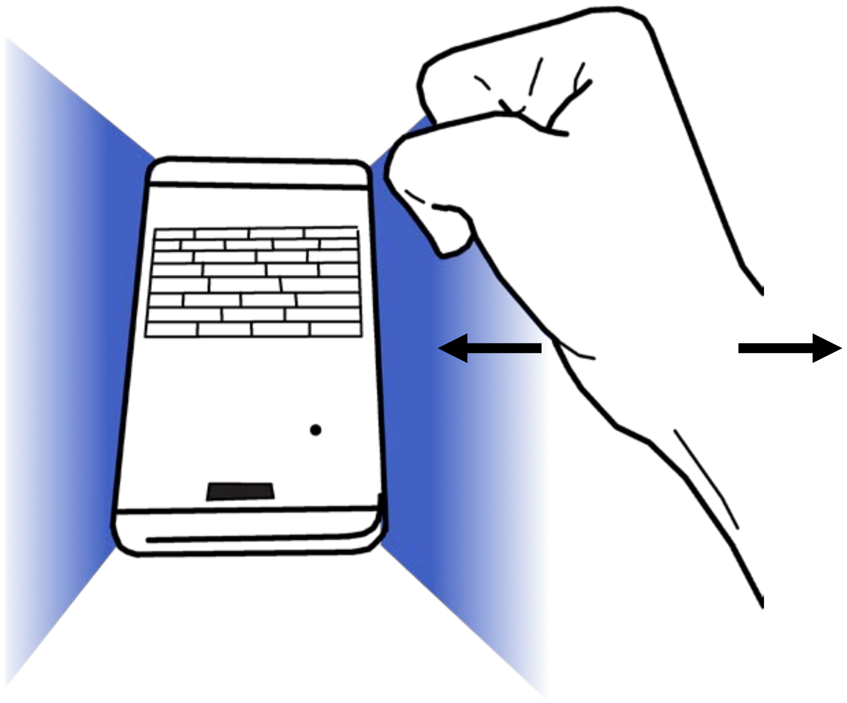


Figure 2.9: Pohl and Rohs envisioned mobile device [174]. The smartphone is equipped with infrared cameras on its later sides to allow around device interactions. In the example, users can play the Arcanoid game with their hand.

controller[47] to detect mid-air gestures and, therefore, allow the user to interact with a mobile device [10, 174, 115]. The Leap Motion is a small rectangular device capable of detecting a users' hands and fingers in an area of 80cm. Two cameras and three infrared LEDs make the recognition possible. To achieve a more compact solution, researchers have built additional cameras directly on the handheld device [158, 120] and some suggested the use of the Leap Motion controller[48] to detect mid-air gestures and, therefore, allow the user to interact with a mobile device [10, 174, 115]. The Leap Motion is a small rectangular device capable of detecting users' hands and fingers in an area of 80cm. Two cameras and three infrared LEDs make the recognition possible.

In 2014, Pohl and Rohs [174] envisioned a future mobile device that will include depth cameras, such as the Leap Motion, on both sides of the device. This futuristic device will allow the recognition of objects or a users' hands around the phone enabling alternative interactions. For example, they suggested games as a possible application where users can control characters by moving their hands near the device. In Figure 2.9, we can see the user controlling the rectangular figure of Arcanoid[49] to shoot bricks. While mobile devices do not yet support these additional sensors, researchers have used the Leap motion for other mobile scenarios such as augmented reality [115] and supporting interactions in cars [10].

Researchers have also exploited the camera already built in the phone to capture in-air gestures [223, 200]. For example, Song et al. [200] proposed a novel machine learning

[47]Leap motion website: https://www.leapmotion.com. Accessed on February 2018.
[48]Leap motion website: https://www.leapmotion.com. Accessed on February 2018.
[49]Web solution for the Arcanoid game: https://goo.gl/X94cqJ. Accessed on April 2018.

algorithm to detect around-device interactions. With their approach, users can perform static pose gestures in sight of the frontal camera of the phone to browse and zoom a document, browse a map or play games. Similarly, Yousefi et al. [223], detect hand gestures via the camera of the phone to interact with 3D objects for augmented reality purposes.

Alternatively to systems that use image recognition, researchers have proposed the use of additional sensors [119] or sonars [149] to detect mid-air interactions. Kratz and Rohs [119] extended a mobile device with six distance sensors to recognise a set of gestures. For example, users can sweep their palms or hand-edge to the right or to the left to browse and select colours in the palette. With FingerIO, Nandakumar et al. [149], used inaudible sound signals played by the mobile device to track the movement of the users' finger. Among all the applications possible with the system, FingerIO allows users to write text on any surface or interact with the phone even when in a pocket.

Non-perceptual Technologies

Jones et al. [107] found that enlarging the mobile-interaction area can be particularly useful for performing tasks. However, using perceptual technologies to achieve this goal can be challenging. This approach requires the user to know the zone where the gesture has to be performed and force the interaction in a specific area that is not clearly delimited by any visual cue. On the other hand, by using external physical devices we can diminish this problem.

Researchers have proposed a number of non-perceptual technologies that allow the users to perform mid-air gestures. For mobile interactions, much attention has been paid to wearable devices such as rings [113, 222, 34] and arm or wristbands [169, 114, 86, 181]. Despite the fact that most of these works do not require the user to physically touch the wearable device, they have to be worn and, therefore, can be categorised as non-perceptual technologies.

In such contexts, an important factor is the ***wearability*** of these smart devices, meaning how users find them in terms of socially acceptability and whether they would wear them in their everyday life [27]. A wearable device has to be fashionable to be accepted and worn by users. For example, as discussed by Martin [135], smartwatch *wearability* can be determined not only by its size and weight but also by its form factor. Overall, a nice ring or a trendy smartwatch will be accepted more easily by people.

On this matter, Chan et al. [34] propose CyclopsRing, a device worn on the finger of the user to recognize hand gestures. The gestures are then used to interact with other devices such as mobile phones. Similarly Kienzle and Hinckley [113] propose LightRing, a smart finger-worn device that detects the 2D position of the fingertip of the user on any surface. Although the authors do not directly mention smartphone applications, this approach would also be feasible to, for example, terminate a phone call or raise the volume of the audio by performing gestures with the finger on a table.

As these works demonstrate, nowadays, powerful technologies can fit into incredibly small devices. However, there is still a compromise between the number of features available on the wearable and its size and, therefore, its acceptance. A small smartwatch that resembles a normal watch could look better and more fashionable than a bigger one, however, it will not include GPS or other more sophisticated sensing techniques. For example, Kim et al. [114] developed Digits, a wrist-worn wearable device capable of detecting 3D hand gestures. With Digits, users can answer calls by mimicking a phone with their hand, or more generally, performing 3D input interactions on a mobile device by moving their fingers. Digits is equipped with a camera facing the palm of the user, motion sensors and LEDs. The sensing capabilities of Digits exceed the ones of more common fit trackers or smartwatches however, its size is far less comfortable.

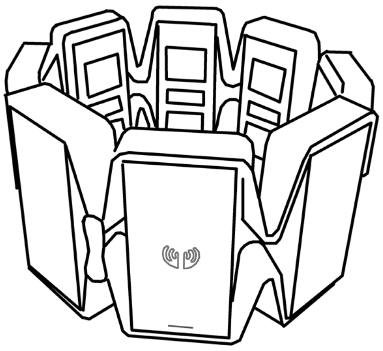


Figure 2.10: The Myo armband.

Alternatively to wristbands, researchers have also studied the use of armbands to perform mid-air interactions [30, 11, 225]. The Myo[50] is an example of such wearable device (see Figure 2.10). It is equipped with accelerometer, gyroscope and eight Electromyography (EMG) sensors that are capable of recognising movements of the muscles. With the Myo, users can interact with other devices, such as computers or smartphones, by performing mid-air gestures of the hand wearing the armband. By default, the APIs are capable of detecting five dynamic gestures: wave-in, wave-out, finger-spread, fist and double tap of the thumb and any other finger (see Figure 2.11). Researchers have exploited the armband to support deaf people in communicating. Paudyal et al. [169] developed Sceptre where the users' signs detected by the Myo will be translated into messages on the mobile device. Other applications of the armband can be found online on the market website of the Myo[51]. Common examples are the possibility to control YouTube, Spotify or Google Earth applications by means of mid-air interactions.

While most of the solutions presented could still be perceived as too invasive or cumbersome, we note that researchers are developing very thin sensors that one day could improve the overall acceptance of wearable devices [215, 28].

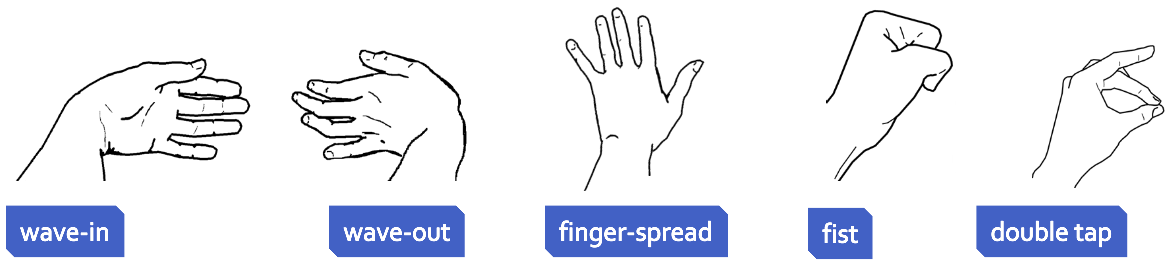


Figure 2.11: The five gestures recognised by the armband: wave-in, wave-out, finger-spread, fist and double tap.

[50]Myo website: https://www.myo.com. Accessed on March 2018.
[51]Myo market website: https://market.myo.com. Accessed on March 2018.

2.3 Cross-device Systems

Nowadays, people not only use a single smart handheld device, but often own and share multiple types of devices, such as tablets, smartwatches and smart TVs. At the same time, public and semi-public screens can now be found in many locations such as streets, airports, offices and train stations. For these reasons, it is clear that we are surrounded by smart devices that can be used for a myriad of different applications and purposes. In these contexts, many researchers have proposed the use of combinations of devices to control screens or share data among other devices by proposing cross-device applications. In Section 2.3.1, we first analyse how many devices people own and how they use them in their everyday life. In Section 2.3.2, we then present common cross-device scenarios and how users can interact with multiple devices by discussing what researchers have proposed in the literature.

2.3.1 Living in a Multi-device Era

As confirmed by a survey conducted by GlobalWebIndex[52], the average digital consumer now owns more than three connected devices[53], and this number is expected to grow in the next decade[54]. While smartphones, tablets and laptops are all capable of performing similar tasks, people tend to switch among them depending on their current activity and physical location. As a Facebook and GfK[55] study on 2000 participants showed, no one device fits all roles[56]. Smartphones are usually preferred for communication purposes, and they are the most commonly used device when people are on the move, such as commuting to and from work. Tablets are often shared with family members and friends, and they are mostly used for entertainment tasks, such as watching YouTube[57] videos or movies. Instead, laptop and desktop machines are preferred for productivity tasks such as managing finances or work. In the same study, they also noticed that users tend to start activities on one device and then move to another device, and they found a correlation between the number of devices owned and this pattern. The more devices people own, the more likely they are to switch among them to complete a task.

Despite this growth of digital devices, people do not always carry their smartphones, tablets or laptops on any occasion [52, 165]. As found by a survey we conducted [52], one out of three participants of a total of 293 participants, did not carry any device when visiting colleagues in nearby offices at work. This number goes up to one out of every two users when we only consider women. This difference might be caused by the lack of pockets of woman cloths. In contrast, as similarly found in the Facebook study, people often carry more than one device when on public transportations, and the majority of people always carry their phones when meeting with friends [52]. During holidays, 89% of our participants stated that they carry their mobile device and a big part of our population also carry their laptop or tablet (47% and 40% respectively).

In addition to mobile devices that people own or share, **public and semi-public**

[52]GlobalWebIndex website: http://blog.globalwebindex.net. Accessed on March 2018.
[53]Report on the number of devices owned (2016): goo.gl/NyJmLB. Accessed on March 2018.
[54]33 billion internet devices by 2020 (2014): https://goo.gl/Vi7RzL. Accessed on March 2018.
[55]Growth from Knowledge (GfK) website: http://www.gfk.com. Accessed on March 2018.
[56]Finding simplicity in a multi-device world (2014): https://goo.gl/JQpLvk. March 2018.
[57]YouTube website: https://www.youtube.com/. Accessed on March 2018.

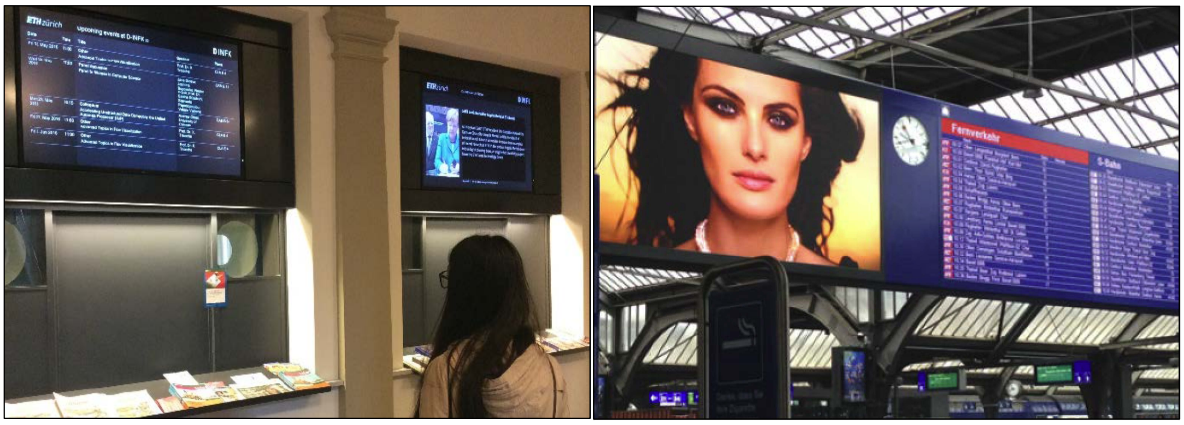

(a) Semi-public screens at ETH (b) Public screen at the Zürich train station

Figure 2.12: Example uses of public or semi-public screens inside buildings (a) and at a train station (b).

screens can now be found in a myriad of different locations. In streets, airports, bus or train stations they often advertise products or services, while within organisations, they are used to raise community awareness as well as providing news or events information [105, 197, 202].

For example, we show the screen in our department in Figure 2.12 (a). These are used to show information about upcoming talks as well as news information. Figure 2.12 (b) shows the main station screen where commercials are displayed next to the list of departures. In our survey [52], we also asked participants how many additional screens were visible from their work desk. 38% of our users stated that they view more than one additional screen from their desk. While it is not clear if these screens were public screens or screens of other colleagues, researchers have suggested the use of these displays to be cyber-foraged [43, 139], meaning that users can temporarily take control of the screen to show some content avoiding examining it on a small device. In an office scenario, when a co-worker is absent, users could use their screen to show pictures or documents from their phone.

2.3.2 Interaction Techniques

As discussed in the previous section and as envisioned by Mark Weiser twenty years ago [216], users can now have access to a multitude of heterogeneous devices. Each type of device is used for different tasks, and they can be found in the home, workplace or in on-the-move environments. It is also clear that users often use multiple devices at a time [52, 188, 48, 106, 189, 203, 45]. For example, watching a TV show while reading related Tweets or checking social media on a smartphone are everyday activities performed in parallel. In our survey [52], we proposed possible cross-device applications and asked participants if they would see themselves using them in the future. The scenarios proposed varied from trip planning to video applications. The feedback was positive overall and the majority of our users seemed to like the idea of combining their devices to perform tasks.

An important factor in such scenario is how users can interact with multiple devices.

Designing cross-device interaction paradigms raises interesting research challenges. For this reason, many researchers have proposed different ways of improving the user experience when interacting in cross-device scenarios.

For instance, researchers have proposed the use of personal mobile phones to interact with big screens such as **public and semi-public displays** [82, 16, 172, 124, 19]. The use of smartphones can potentially mitigate the absence of interactivity of these screens. In fact, the majority of public or semi-public displays do not allow any of form of interaction. They usually offer some form of presentation rather than allowing interactive applications. In some cases, big displays are capable of detecting touch; however, hygiene issues can be particularly problematic in public places such as airports, train or bus stations [47, 187, 73]. Moreover, to allow touch interactions, the screen needs to be located in a place easily reachable by the user raising security issues since a screen could be the victim of vandalism. Touch is not a feasible interaction when screens are distant, meaning that they are not placed within reach of the user. Furthermore, by using touch interactions on a display, users will occlude the screen to other users standing behind [47, 6].

Alternatively, cameras, such as the Kinect, can be installed near the screen to support mid-air gestures as a form of interaction with the display [155, 143, 213, 206, 214, 130, 153]. Given its novelty, this approach could attract more people to come near to the screen [143]; moreover, it solves the hygiene issues raised by direct touch gestures. However, recognising mid-air gestures via image processing algorithms requires the setup of additional hardware such as the camera as well as a computer connected to the screen capable of detecting these interactions.

For these reasons, the use of personal mobile phones to interact with bigger screens can be a good alternative. By using a smartphone, the users do not need to touch on a potentially dirty screen and no additional hardware alongside the display is required.

Some researchers have exploited the use of mobile phone to tap on a bigger monitor [26, 82, 83, 5]. For example, Hardy and Rukzio [82] propose Touch and Interact, an interaction technique in which the user can select elements on a screen by bumping their phone on any position of the bigger display (see Figure 2.13).

This technique was later used for tourist applications, where users could interact with a projected map with their phone to query and browse point of interests [83]. Using this approach, the use of smartphones to directly touch a screen allows the system to identify users and the screen of the handheld device can be used to show additional and more private information [192, 5].

Alternatively, researchers have exploited the use of the touchscreen or physical buttons of handheld devices to interact with a bigger display [180, 136, 23, 148]. For instance, Rekimoto [180] proposes the use of mobile devices capable of detecting touch as a tool palette to write or draw on a whiteboard. Similarly, Matulic et al. [136] exploit smartphones to easily change the thickness and colour of pen strokes on a iteractive whiteboard. Alternatively, Boring et al. [23] allow users to remotely control a pointer displayed on the screen by using buttons on the handheld device.

Touch interactions have also been combined with image recognition algorithms to support **mid-air gestures** [16, 194, 166]. For instance, Bragdon et al. [16] allow users to point with their phones toward the screen and manipulate its elements by touching on the handheld device. Some researchers have also exploited the use of **gaze** recognition algorithms where users first look at a portion of the bigger screen and then touch on the

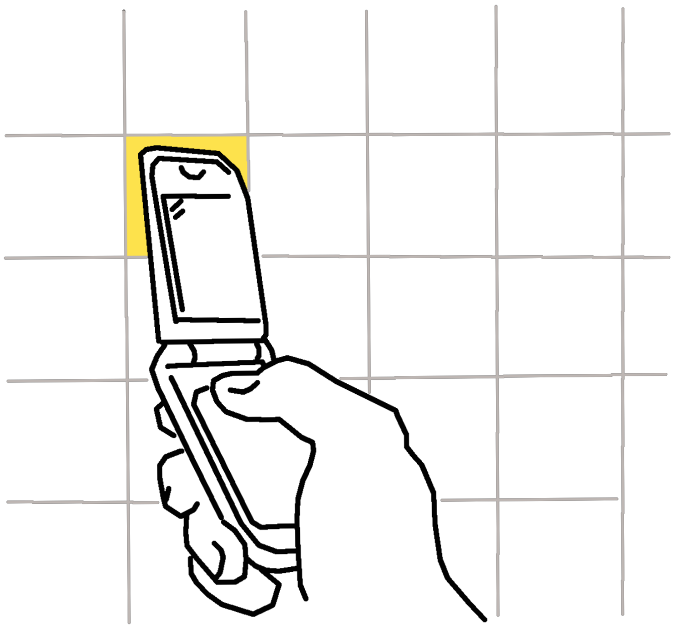


Figure 2.13: Hardy and Rukzio [82] solution for selecting elements on big screens via mobile phones. To perform the interaction, users need to tap the display with their phone on the desired area.

mobile phone to manipulate and select elements [203, 204]. Although these techniques allow users to interact with a bigger display intuitively, they require the use of multiple cameras to recognise pointing or gaze gestures. For this reason, Boring et al. [24] have proposed a system capable of capturing elements displayed on a screen by using the camera of the phone and therefore, without the need of additional hardware (see Figure 2.14).

Another option is to use **tilting interactions** on mobile phones to interact with public and semi-public displays [23, 16, 45, 208, 60]. Motion gestures can be particularly advantageous when interacting with a public screen. In contrast to many touch-based interfaces, tilting interactions do not require the user to look at the small screen, so they can focus their attention on the bigger display without continuously shifting their gaze from the handheld device to the screen.

Boring et al. [23] compare three main forms of interaction to control a cursor remotely on a distant screen: scroll, tilt and move. As mentioned before, the scroll technique allows users to use physical buttons available on the handheld device to move the cursor with a constant speed. With tilt interactions, users move their phone via *velocity-based* tilting interactions: the more the device is tilted, the faster the cursor will move. In contrast, with the move technique, the phone's movements are linearly mapped to the pointer's position similar to a mouse. The move interaction is implemented by using the phone's camera. In a user study, they found that both move and tilt gestures were faster but more prone to errors than scroll.

Researchers have also exploited the use of motion gestures in games [60, 16, 208]. For example, users can control cars or other characters displayed on the bigger display by simply moving their phone. Other applications of tilting interactions to control public screens are the browsing of media galleries or maps. For instance, Dachselt and

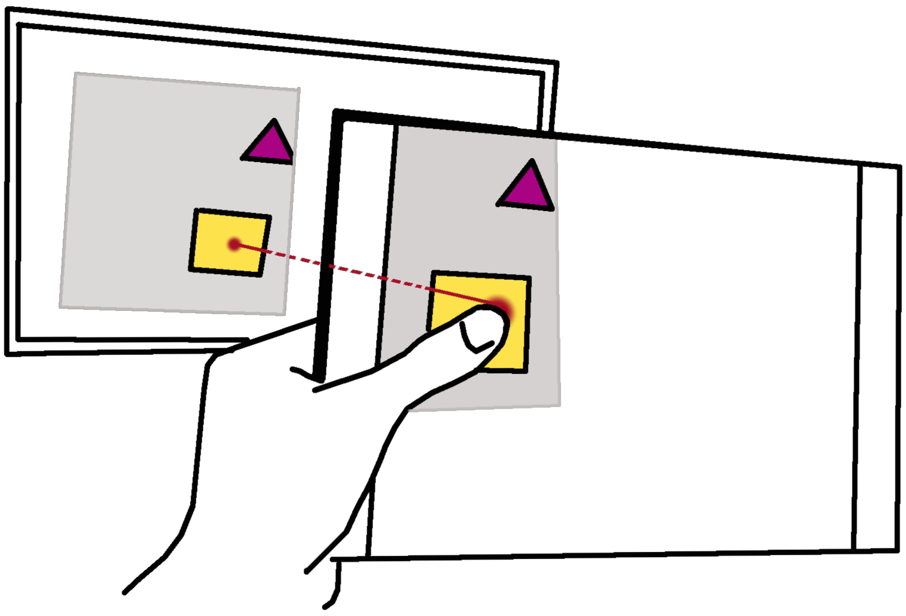


Figure 2.14: Boring et al. [24] solution for manipulating elements displayed on public displays using point and touch interactions on mobile devices.

Buchholz [45] studied the use of jerk tilting gestures to browse a large music collection and more continuous interaction to operate Google Earth[58] remotely on a bigger and distant display.

Tilting gestures have also been used in combination with pointing interactions. With PointerPhone [194], users can manipulate elements displayed on a screen by first pointing at them with their phone and then rotate or move the handheld device. We also note that tilting gestures have been used to navigate in a 3D space displayed on a bigger monitor [124].

Researchers have exploited the use of **wearable devices** such as smartwatches, gloves and armbands to directly interact with distant displays [81, 111, 171, 198, 211]. For instance, Haque et al. [81] used the Myo armband to remotely control a cursor on a large monitor by proposing MyoPoint. With MyoPoint, the movements of the users' arm are mapped to the cursor and gestures, such as fist and finger spread, used to select elements or activate and deactivate the interaction. Taking this work as a source of inspiration, Katsuragawa et al. [111] proposed a variant of MyoPoint, called Watchpoint, that enables users to interact with public screens via a smartwatch. While these works require the user to carry an additional device, in contrast with image processing approaches, they allow the recognition of mid-air gestures without the necessity of additional cameras or other hardware connected to the screen.

In order to interact with a public screen via a mobile phone, devices need first to be paired to initiate the connection. The research community has largely studied pairing mechanisms that can be employed not only on *public screen - smartphone* scenarios but in a multitude of other configurations [94, 95, 164, 178, 91]. For instance, in a meeting, multiple heterogeneous devices, such as laptops, big screens and projectors, tablets and phones can be simultaneously used to interact with each other [199, 195, 98]. Using QR

[58]Google Earth website: https://earth.google.com/web. Accessed on April 2018.

codes to share URLs is the most common and easiest way to create a connection among devices [72, 99, 4]. QR codes can be employed in almost all configurations of devices: the QR code can be shown on one of the available screens, such as a public display, and then one or multiple mobile devices can initiate the connection by scanning it. To scan a QR code, users are required to have the right app on their mobile device and then position the device in the correct way to scan it. For these reasons, this method requires many steps and can feel cumbersome to the user.

For this reason, the research community has proposed more intuitive methods. For instance, users can connect two devices by using a stylus or draw a stroke from one device to the other [94]. Alternatively, users can simultaneously shake [95, 137, 31] or bump devices [178, 91] as well as performing a pinch gesture or joint interactions [164, 38].

Proximity-based approaches have also been proposed as a possible way to naturally establish communication between devices [191, 132, 191]. Marquardt et al. [132] defined different zones of engagement to show information on a screen depending on how close the devices are from each other. In this context, the position of users can also be exploited to understand groups of people holding a device and therefore, create connections among them [133]. We also note that researchers have combined physical and social proximity to pair phones or tablets [102].

Researchers have also exploited the use of the Doppler effect to initiate connections between devices [12, 76, 37]. With DopLink [12], users can pair devices by pointing their mobile phone, or tablet, towards the desired target. DopLink is capable of establishing the connection by playing an inaudible sound from the source device and studying its Doppler effect. With the same technique, SurfaceLink [76] allows users to pair two devices, which are flat on the same surface, by swiping their finger on the table from one to the other. Similarly, with AirLink [37] users can initiate communication between devices by performing in-air swiping gestures from one device to the other.

DopLink, SurfaceLink and AirLink not only allow users to pair devices but also to **share data** among them. In this context, we can categorise two primary forms of interaction: spatially-aware and spatially-agnostic. In **spatially-aware techniques**, the physical position of devices influences the interaction users will perform. Mid-air gestures, as proposed by the above mentioned projects, are particularly suited in this scenario. Swiping gestures from one device to another are intuitive interactions techniques to share data among devices. Alternatively, users can share an element from one device to another by dragging the object toward the edge that is the closest to the desired target [176, 80]. Researchers have also proposed the use of tilting gestures to share data among co-located devices [220, 45, 129, 122]. For example, in the system proposed by Dachselt and Buchholz [45], users can share the currently selected picture from their phone to another screen by performing a tilt gesture in the direction of the target device (see Figure 2.15). Toss-it [220] exploits this form of interaction by allowing users to send data from PDA to another PDA or printer via pointing and motion gestures. Toss-it requires the user to first point with their handheld device to the desired target and then tilt their phone. The location of the source device is recognised by the use of LEDs built on the PDA and a stereo camera installed in the room. Different blinking patterns of the LEDs allow the system to identify each device.

In contrast with spatially-aware techniques, **spatial agnostic interactions** can be performed independently from the position of the source and target devices [176, 80].

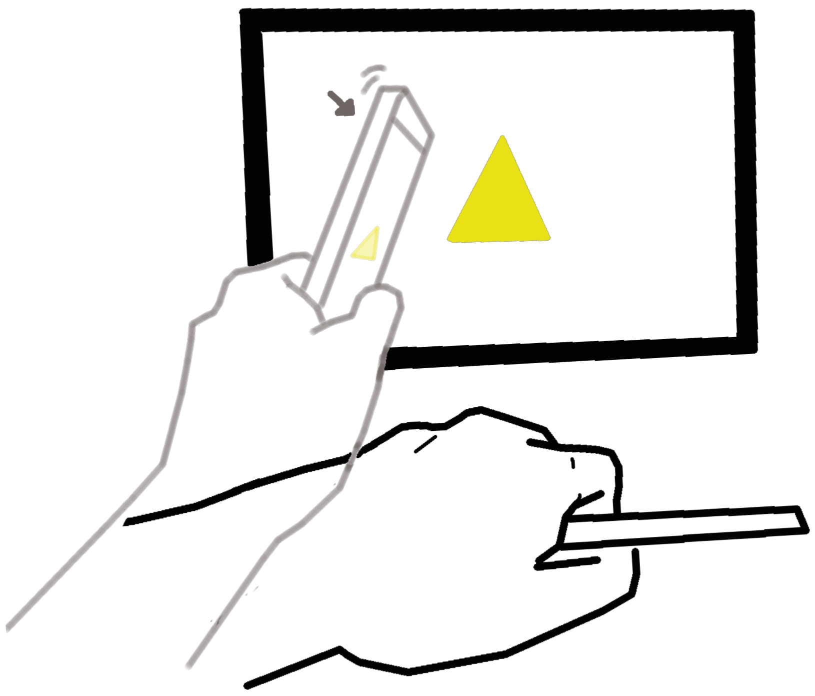


Figure 2.15: Dachselt and Buchholz [45] solution to share a picture from a smartphone to other devices via tilting gestures.

Often, these techniques extend user interfaces to add cross-device features. For example, Hamilton and Widgor [80] proposed Conductor, a system that allows users to select Cues, a particular type of data, and send it to other devices. Once a Cue is selected via a hold tap gesture, a pop-up menu will appear allowing users to send the data to the desired target or set of targets.

Chat systems and cloud services such as iCloud[59], Google Drive[60] or Dropbox[61] also allow users to share data among their devices. However, these systems require the user first to copy the desired content and then open a new window or applications to paste the selected data. This technique requires the user to stop their current task and switch their attention to share a picture or a link. Moreover, by using chat or cloud systems, the data shared is sent to all devices rather than just the desired subset. While broadcasting data can simplify the interaction process, it will cause the chat or cloud system to push annoying notifications on all devices.

Researchers have carried out several studies to understand better what category of interactions, spatially-agnostic or spatially-aware, is preferred by users. Since user-defined gestures have been shown to be more memorable and usually preferred by users when compared to interactions defined by experts [157, 146], researchers have asked users to suggest possible cross-device interaction techniques via **elicitation studies** [176, 121, 150, 195].

[59]iCloud website: https://www.icloud.com/. Accessed on April 2018.

[60]Google Drive website: https://www.google.com/drive/. Accessed on April 2018.

[61]Dropbox website: https://www.dropbox.com/. Accessed on April 2018.

Rädle et al. [176] found that users preferred spatially-aware gestures. Similarly, Kray et al. [121] found that the location of the target device influenced the majority of the suggested gestures proposed by users. However, spatially-aware gestures can raise conflicts when the target device is physically close to other devices. As found by a study conducted by Nebeling et al. [150], users tend to prefer spatially agnostic interactions if devices are close together.

Overall, spatially-aware gestures can be natural and intuitive solutions, however, they are often harder to develop than spatially agnostic techniques. Moreover, while agnostic solutions can also work when devices are not co-located, spatially-aware interactions, by definition, are dependent on the physical proximity of the source and target devices. In this context, researchers assume that devices are co-located as well as turned on in order to start a communication. However, as we found in our survey [52], users do not always have all their devices within sight, or even within reach. While mid-air gestures or tilting interactions have mostly been used when devices are co-located, they could also be exploited for target devices that are not physically close. For example, a jerk tilting gesture or a wave interaction could allow the user to share an article they are reading in the office to their desktop PC at home for future reading.

2.4 Conclusion

As discussed in this Chapter, building web applications as opposed to native solutions offer many advantages for both developers and end-users. Over the years, browsers have been extended to support many features that make them as powerful as native apps. Nowadays, websites adapt to the screen real estate of the client, are fast and can also work offline. In cross-device scenarios, web technologies play a crucial role since they can run over different configurations of devices and platforms [97]. While cross browser compatibility problems can be a big challenge to overcome, frameworks can help developers to build single and cross-device applications that run on heterogeneous devices. From the user perspective, accessing a web application does not require the user to download a special purpose app, that could potentially be used only rarely and easily forgotten.

However, despite the power of current IT web solutions, the primary form of interaction used in mobile web applications has mainly relied on touch gestures. Currently, there is not much support for experimentation with alternative forms of interaction on the web, for both developers and end-users. While a large number of front-end frameworks are now available online, their focus is not on different forms of interaction but instead on the organisation of content [152]. Similarly, EUD systems and CMSs platforms mainly focus on the look and feel of the final website rather than allowing alternative interaction techniques.

Although mobile browsers are now capable of detecting the orientation and acceleration of devices, motion gestures have mainly been exploited on native solutions. As discussed in the literature [15, 96], tilting interactions offer a number of advantages, for example, they are eyes-free gestures and, in contrast to touch gestures, can be performed without occluding the screen. Researchers have studied this form of interactions intensely and proposed many variants which can be potentially influenced by a number of parameters, making the implementation of these gestures even more challenging.

For these reasons, we decided to investigate what forms of frameworks and visual tools could be developed to support both developers and end-users in their experimentation with new forms of interaction such as tilting and mid-air gestures. Further, we wanted to bring the knowledge acquired in single device scenarios to cross-device applications and developed frameworks and tools to exploit alternative forms of interactions when more devices are involved in the communication.

3 Tilt-and-Tap

In this Chapter[1], we present our research addressing how to enlarge the set of possible interactions in single device web applications. We started our investigations by first supporting developers with a set of high-level JavaScript APIs via Tilt-and-Tap (TAT), a framework for the rapid development of tilting interactions in web applications. In Section 3.1, we present the framework, and its implementation details as well as example applications developed with the proposed APIs. Moreover, in Section 3.2, we discuss TAT 2.0, which is an extension of the first framework that supports a broader number of features and offers a larger set of tilting interaction.

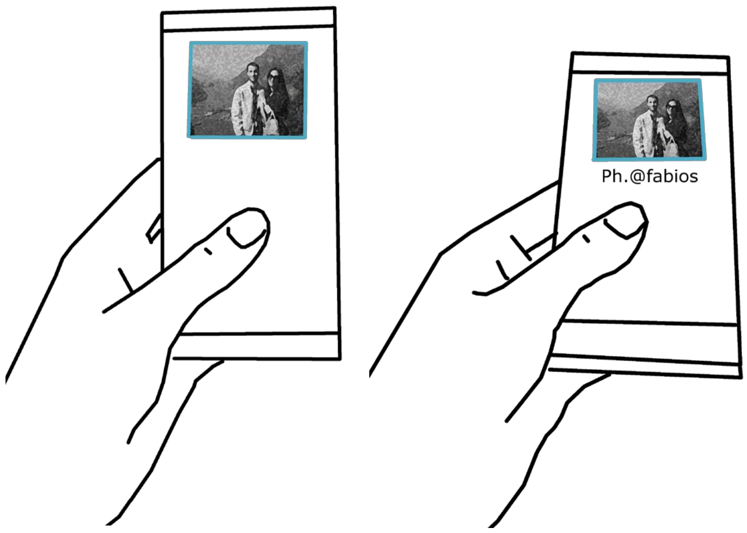
Furthermore, we continued our research by targeting end-users with the development of WP-TAT, a WordPress extension that allows non-technical users to experiment with tilting interactions on their website. As the name suggests, WP-TAT is based on Tilt-and-Tap and offers its main features via a visual interface. We discuss WP-TAT in Section 3.3. A number of user and developer studies have been carried out for each of the developed tools, and they are presented in the corresponding sections.

The experience gathered during the implementation of TAT, TAT 2.0 and WP-TAT, as well as the feedback received from users during studies, gave us the chance to draw a series of design observations on when and how to apply tilting interactions in web applications. We discuss these guidelines in Section 3.4.

3.1 The Framework

Tilt-and-Tap (TAT) is a JavaScript framework, designed to encourage developers to build web applications that exploit tilting interactions. TAT has the goal of allowing developers to experiment with these alternative forms of interaction by offering a high-level set of APIs. Developers are not required to directly manage the raw data returned by sensors or deal with implementation differences between browsers.

[1]Earlier versions of parts of this Chapter were originally published as Di Geronimo et al. [50, 49, 56, 57]



index.js

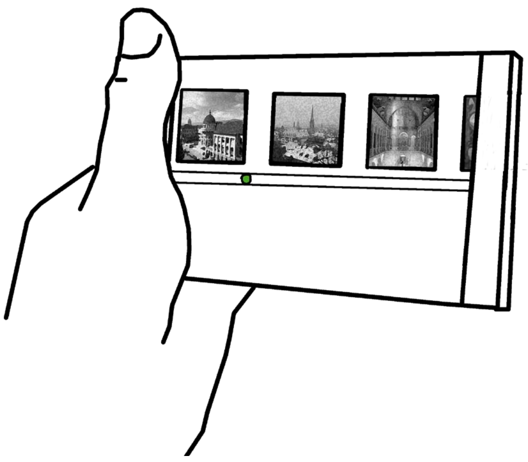
```
var t = new
tiltandtap({
tiltDown: {
callback: openPh,
element: “body”,
touch: “hold”
},

tiltUp: {
callback : closePh,
element : “body”,
touch: “hold”
}});
```

Figure 3.1: Jerk tilting example and its correspondent code using TAT.

Taking previous works on motion gestures [15, 179, 92] as a source of inspiration, TAT offers two main forms of interaction: jerk and continuous tilting gestures. **Jerk interactions** allow users to perform actions by quickly moving their device in some direction. In contrast, **Continuous motion gestures** are sustained tilts of the device in a direction to perform actions such as scrolling through a list of items.

As suggested by Ken Hinckely and Hyunyoung [92], TAT also offers **combinations of touch interactions with tilting gestures**. Single tap, multiple taps and hold tap interactions can be combined with motion gestures. Moreover, visual, vibrotactile and audio **feedback** are also available through the plugin, and they can be triggered when a tilt interaction has been performed.

To give a general overview of the framework, in Figures 3.1 and 3.2, we show an example of how of Tilt-and-Tap can be used. In the jerk tilting example, users can see additional information (photographer name in the example) of a picture by hold tapping anywhere in the page and then tilting the device down. A tilt up will hide



index.js

```
var t = new
tiltandtapc({
target_elem:
“pictures”,
onelementselected :
border,
});
```

Figure 3.2: Continuous tilting example and its correspondend code using TAT.

the box with the information. To reproduce this behaviour, developers have to create a `tiltantap` instance and define callback functions and other options for `tildDown` and `tiltUp` gestures. For `tiltDown`, the function `openPh` will show the name of the topographer. In contrast, for `tiltUp`, the `closePh` function will hide this information. These two functions are created by the developers and they can be placed in the scope of the Tilt-and-Tap instance. For both gestures, the user has to hold tap on the screen while tilting the device up or down, to allow this behaviour developers have to specify that the `touch` gesture required is a `hold` tap and the interaction can be performed inside the `body` element.

In contrast, the continuous motion example in Figure 3.2 allows users to scroll forward and backward through a list of pictures by slowly tilting the device to the left and right respectively. A ball serves as feedback to the user to understand the direction of the interaction. When the ball is below a picture, a border will be added to the image. To reproduce this behaviour only a few lines of code are required. Once the framework is included in the HTML file, the developer has simply to define the desired type of gesture and which callback function has to be called once the green ball is above an element. In Figure 3.2, Tilt-and-Tap is able to recognise the pictures involved in the interaction by their class name `pictures` that is passed to the `target_elem` parameter of the framework. Once the indicator is under a picture, Tilt-and-Tap executes the `border` function associated to the `onelementselected` parameter.

The framework sets an element as selected if the ball is above it. By default, the ball has to be 100% *inside* an element to execute the `onelementselected` callback function. If desired, developers can change this percentage. Moreover, developers can also customise the look and feel of the ball or to completely hide it from the page. In this case, even if not visible, the ball continues to be present in the page and scrolls the elements indicated in the `target_elem` parameter. Given the nature of this element, we often refer to the ball as an indicator or cursor. However, in contrast with a mouse cursor, the selection process does not necessarily require the user to perform an additional interaction once the indicator is over an element. With a mouse, users first move the cursor over an element and then right click on it to select. With continuous motion gestures, the selection behaviour is decided by the developer via the `onelementselected` function.

To improve readability, we often refer to Tilt-and-Tap as one single framework, however, for performance reasons, we offer one framework for each form of interactions: jerk and continuous. As we can see from Figure 3.2, the instance of Tilt-and-Tap for jerk tilting is named `tiltandtap`, while the instance for the continuous example is named `tiltandtapc`.

In Section 3.1.1, we start presenting Tilt-and-Tap by first explaining how it is implemented. A number of applications developed with the framework are then discussed in Section 3.1.2. To better understand the power of the proposed approach, we carried out two studies, a developer and a competition study and these are presented in Section 3.1.3 and Section 3.1.4 respectively.

Please note that Tilt-and-Tap was initially presented as a jQuery[2]-like framework [50] however, in this thesis we will discuss a different version of TAT that does not require jQuery. While some operations were made easier in the development process by using

[2]jQuery website: https://jquery.com. Accessed on April 2018.

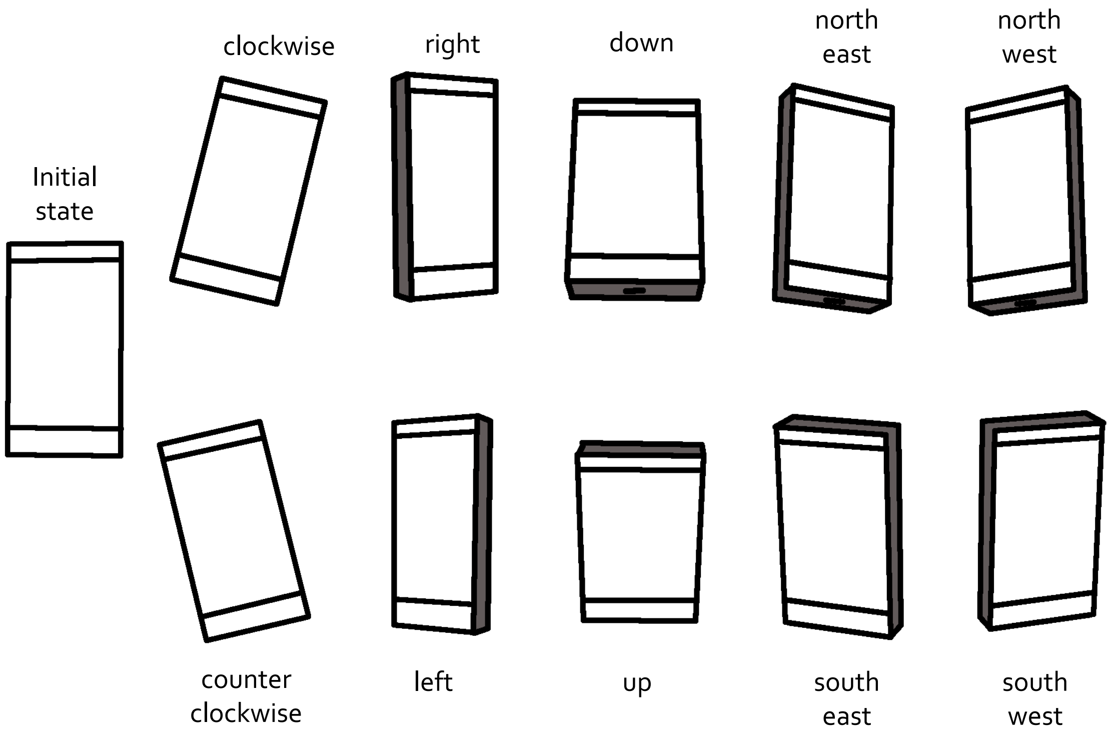


Figure 3.3: The ten supported directions of TAT.

jQuery, the plugin is particularly heavy to download if not already cached (circa 200mb) and adds an additional point of failure if some of its features are changed or compatible with the browser. Moreover, the recognition of jerk tilting gestures was altered to improve its accuracy. We discuss more details about these modifications in the corresponding section. However, the two versions share the same features and similar APIs therefore, they are not different in terms of motivations and goals.

3.1.1 Supporting Tilting Interactions

In this section, we present implementation details of Tilt-and-Tap discussing how jerk and continuous gestures were developed and how we solved incompatibilities issues among browsers.

Jerk Tilting

Tilt-and-Tap offers jerk tilting interactions in ten possible directions: up, down, left, right, south-east, south-west, north-east, north-west, clockwise and counterclockwise (see Figure 3.3). In contrast to the solution proposed by Baglioni et al. [15], we offer clockwise and counterclockwise as additional directions. We speculate that these two interactions were not studied initially by Baglioni et al. because they might be cumbersome to perform with smartphones. However, moving a tablet clockwise or counterclockwise is more natural and we were considering more general forms of mobile devices. Working with web technologies allowed us to quickly experiment these interactions on a broader set of devices also supporting developers to adapt gestures according to the device used.

With JavaScript, developers can access the raw data fired by motion sensors via two events: `DeviceOrientationChange` and `DeviceMotionChange`. The first event returns the orientation angles of the device via three integer values: alpha, beta and gamma. The angles represent the difference between the current position of the device and its rest "state" (flat on a table). `DeviceMotionChange` provides information about the acceleration of the device expressed in Carteesian coordinates and relative to the Earth frame. The event also returns the `rotationRate` object that defines the rate at which the device is rotating around each axes in degrees per second. Tilt-and-Tap uses the data returned by the `rotationRate` function to recognise jerk tilting gestures. Both events are called at regular intervals; however, this interval may change depending on the browsers used. For this example, we will assume that this interval is 50ms.

To detect jerk motion gestures, we implemented an extended version of the solution proposed by Baglioni et al. [15]. As the authors recognised in their work, every time the user performs a rapid movement of the device in some direction, the natural elasticity of the wrist will automatically move the device near to its rest position. In the Boring et al. [15] implementation, a single jerk tilting interaction is composed of two movements: the first in the desired direction of the rapid motion gestures, the second, to the rest position of the device. This implementation assumes that users are in a position that easily allows them to hold the device in its rest position. However, if, for example, the user is lying on a sofa, this assumption may not be true. Holding the device parallel to the ground may be hard when the user is not standing. For this reason, we propose an alternative solution that takes advantage of the recoil factor. When users perform a tilt gesture in some direction, a rapid and involuntarily movement of the device in the opposite direction is also performed. This is implemented as follows:

1. When the framework is initialised, it has an empty buffer of size three.
2. Every 50ms, we store the three values (alpha, beta, gamma) returned by `rotation-Rate` in the first free position in the buffer.
3. When the buffer is full, we check the last object stored:
   (a) If any of the sensor values in the object are bigger than a threshold, we save this information and go to step four.
   (b) If none of the values is larger than the threshold, we clear the buffer and start again from step one.
4. We call the function defined by the developer for the tilting gesture recognised.
5. We clear the buffer and discard the next three sensor readings to cater for recoil.

In Figure 3.4, we show an example execution of our implementation when the user performs a tilt down gesture assuming that the interaction has to be performed in 150ms. When the device is moved down, the beta angle returned by `rotationRate` is influenced. If the absolute value of this angle is larger than the threshold (3 in the example) after 150ms, the framework recognises the jerk interaction and executes the corresponding callback function defined by the developer. In this case, the background colour of the web page is changed to blue. Since a recoil factor could potentially trigger

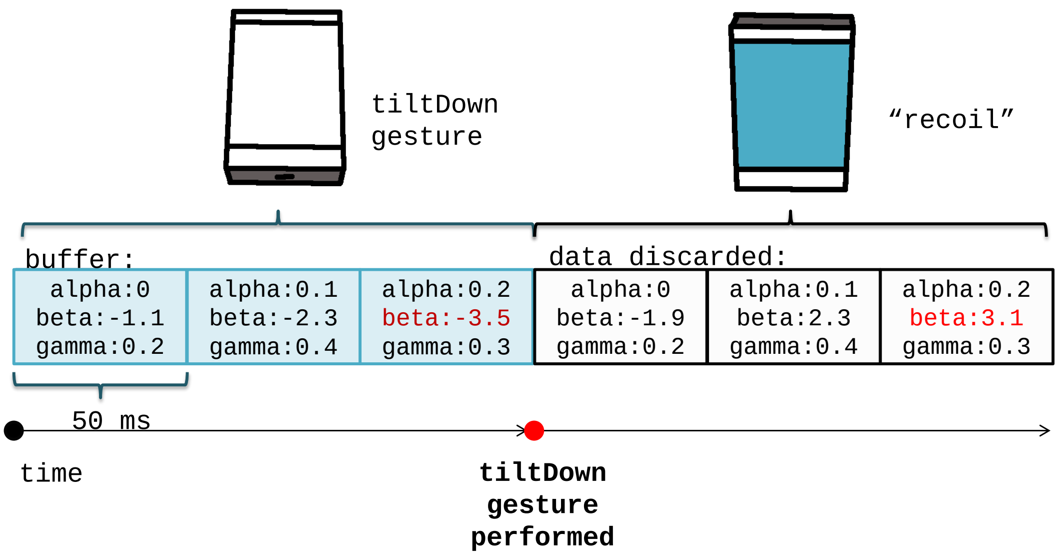


Figure 3.4: Visual representation of TAT recognition algorithm.

a tilt gesture in the opposite direction, we skip the next data fired by the sensor. In a later version of the framework, we improved the recognition of gestures by using a sliding window technique with overlap to avoid missing a jerk interaction due to the fixed size of the buffer. Moreover, we detect a gesture if the standard deviation of the entire buffer is bigger than a threshold. Overall, the general approach of the algorithm is similar; the recoil is still implemented and essential for the detection of a tilting interaction.

Continuous Tilting

In contrast to jerk gestures, Tilt-and-Tap also offers slower movements to perform continuous interactions. Taking previous works in the area [179, 40, 212] as a source of inspiration, we implemented continuous tilting gestures to allow developers to experiment with alternative forms of interaction. Our solution involves the use of an indicator that acts as a cursor and will infer the position of the viewport with respect to the orientation of the device. The position of the indicator is used to scroll the viewport, and it helps the user to understand the direction and the speed of the vertical or horizontal scrolling. Continuous tilting gestures can be performed to scroll elements in one or two dimensions. In one dimension, motion gestures act as traditional scroll bars to browse elements in a vertical or horizontal list. In two dimensional cases, the ball is free to move in both directions. In this section, we use a one-dimensional example to discuss how Tilt-and-Tap works, however, the two implementation use similar concepts and only differ slightly.

In Figure 3.5, we can see a visual representation of our implementation of continuous tilting interactions. Users can horizontally scroll through the gallery of pictures by moving their device to the left and the right. When the ball is under a picture, the image is selected (bold green border).

Continuous tilting gestures are implemented via the the `accelerationIncluding-Gravity` object returned by the `DeviceMotionChange` event. This object is represented as a vector (x,y,z) and describes the acceleration including gravity of the device which is

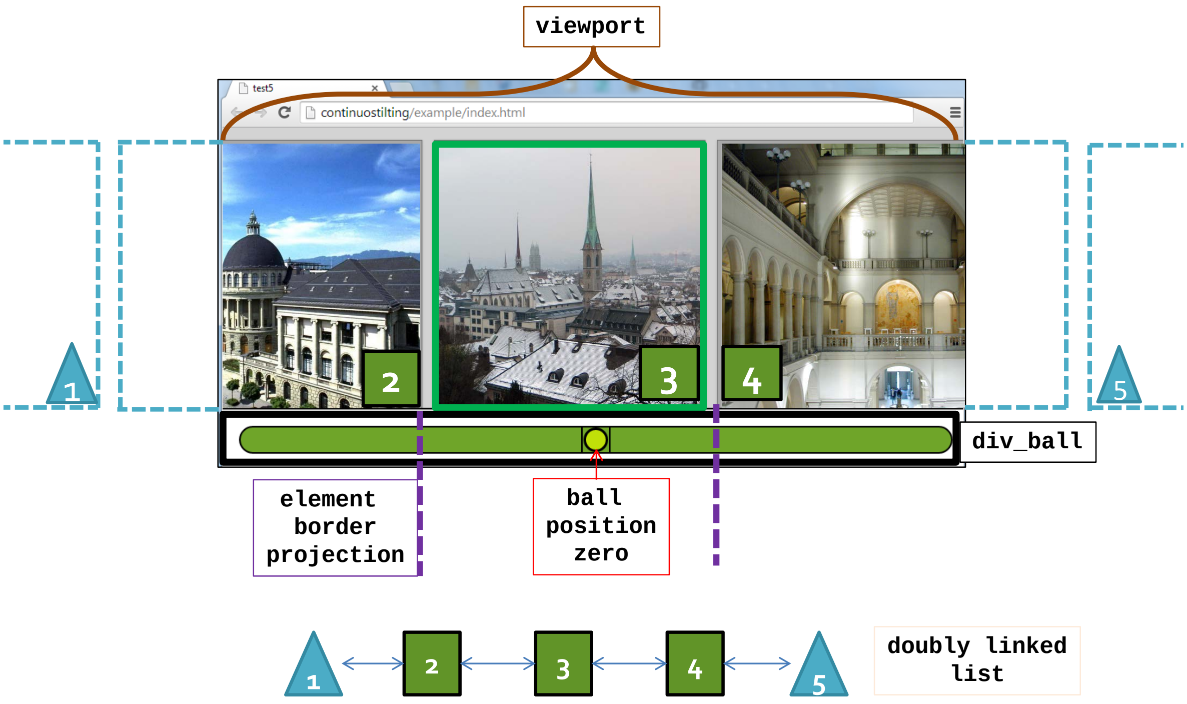


Figure 3.5: Visual representation of continuous tilting gestures. The list of pictures is stored in a doubly linked list. For every movement of the device, we keep track of the elements that are not visible in the viewport (triangles).

returned in meters per second squared (m/s2). As the name suggests, the value returned by the `accelerationIncludingGravity` function is the sum of the acceleration caused by the user and by the gravity.

In Figure 3.6, we can see the pseudocode of our solution which involves five steps:

1. `listElementToSelect` When the document is completely loaded, we create a doubly linked list of all the elements with the class name defined by the developer. Depending on their position, we order the elements in the list. We also remember the first and last element visible in the viewport via pointers.

2. `moveViewPort` In this step, we move the viewport every 50ms depending on the orientation of the device.

3. `updateList` At this stage, we check which elements are in the viewport and update the list and pointers accordingly.

4. `moveBall` Depending on the speed and orientation of the device we update the ball position. We filter minor movements with thresholds. Initially, the ball is located in the centre of its container (`div_ball`). We assume that the ball moves relative to the position of the device when the page is loaded for the first time, thereby accommodating a broad set of initial positions of the user (lying on a sofa, sitting on a chair, standing etc.). This solution contrasts with other works that assume that the rest position is represented by the device flat on a surface. The direction of the ball is decided by a `sign` variable that is set at the beginning of the computation since it depends on the current browser used.

```
onDocumentReady() {
listElementToSelect();
 onmotionchange (Acceleration
data)
    {   updateList (data);
        moveViewport (data);
        moveBall (data);
        elemSelected ();
    }
}
```

```
listElementToSelect()
{
//foreach element in the page with class
//class_elem (specified by the developer)
foreach (element has class class_elem)
      {
     //element contains the ID of the element
     //its dimensions and coordinates
         list_elem [i]= element;
     //double linked list implementation
         list_elem [i] ->next =null;
         list_elem [i] ->prev =list_elem[i-1]
         list_elem[i-1] ->next =list_elm[i]

   //if element is the first one or the last
   //one visible in viewport maintain a
   //pointer to that element
         if(element first in viewport)
         {first_v = element;}
         if(element last in viewport)
         {last_v =element;}
      }
   //order elements in the doubly linked list
   //according on their position in the page
   orderList(list_elem);
}
```

```
moveBall (Acceleration data)
{//if the acceleration is bigger than a threshold
if(Δ acc > th) {
 if (first call)
       {//set the ball in the centre of the div_ball
         setZeroPositionBall()
       }
 new_ball_position.top = ball.top +
((sign)·data.x·speed);
 new_ball_position.left = ball.left +
((sign)·data.y·speed);
//if the new position will put the ball outside the
//displayed
//page it does not move the ball
 if ( checkBallPosition (new_ball_position) ) {
         ball.top = new_ball_position.top;
         ball.left = new_ball_position.left;
       }
}
```

```
elemSelected()
{
  pointer = first_v;
  while (pointer != last_v; )
     {  //check if the ball is inside an element
        //(for a percentage defined by the developer)
        //using its
        // dimension and coordinates
         if(isInside( ball.position, element, percentage))
         {
           cur = element;
            if( elemselected != cur)
            { elemselected = cur;
            //call the function defined by the user
            //when a new element is selected
              onElementSelected(elemselected);
              }
        return;
          }
     }
}
```

Figure 3.6: Pseudocode of continuous tilting gestures. The block of code highlighted in blue represents the first function called.

5. `elemSelected` If the ball is above an element, we trigger the `onElementSelected` function that the developer defined. Developers can also define how much the ball has to be inside an element for it to be selected. By default, the percentage is 100% meaning that the entire ball has to fall inside an element to trigger the defined callback function.

Our doubly-linked list allows our solution to have an O(s) complexity execution every 50ms where s is the number of elements inside the viewport. A possible improvement could involve the use of a heap map with the position of elements as keys. However, since the number of elements in the viewport is small, we decided that the doubly linked list was a good solution for our problem.

Continuous tilting gestures have more options than jerk interactions, however, most of the customisable parameters are optional. While more skilled users can have more control over the framework, less experienced developers can still easily use our framework. Among the various parameters, developers can specify if any touch gestures are necessary to activate the gesture and define the speed as well as the look and feel of the ball. For instance, while the ball is essential for our solution, it can be hidden in the page. In this situation, the ball will still infer the scrolling behaviour of the viewport, but it will not be visible in the page.

Fix Cross-browsers Incompatibilities

While building a web framework allowed us to experiment with a large number of device sizes and platforms, working with web technologies is not free from issues. The main challenge of building Tilt-and-Tap was to develop the framework with two work-in-progress APIs. The `deviceMotionChange` and `deviceOrientationChange` events are not fully implemented by all browsers which currently offer different and, often conflicting, implementations. For instance, the range of angles returned by `DeviceOrientation-Change` and therefore, by the `RotationRate` APIs differs depending on the browser used. To fix these issues, we employ different thresholds for each browser. The thresholds were empirically set and studied to allow similar gestures across different browsers and platforms. However, developers are free to customise them to their needs.

Moreover, the sign that indicates the direction of the movement of the device (left, right, up or down) differs between Android and iOS platforms. After detecting the type of platform of the device, a switch case condition allows the detection of the right direction and offers consistent interaction in both landscape and portrait states.

An additional major difference among devices relies on the granularity of data received from sensors. Via the `interval` value returned by the `DeviceMotionChange` event, we can detect how often the browser will fire motion data. This value differs depending on the browser but, more importantly, it does not truly represent the frequency at which we receive data from sensors. While the parameter could indicate that data are sent every 100ms, in reality, this data can be sent more frequently.

In a previous version, to fix this inconsistency, we just employed a buffer and forced all browsers to collect at least T values (three by default) to detect the gesture. In the updated version of TAT, we add more consistency among devices by forcing the collection of raw data every X-ms. After an empirical test of combinations of most updated browsers and platforms, we found no situations in which motion events are fired slower than 50ms. For this reason, by default, X is set at 50ms. We prevent faster browsers firing more events with JavaScript timeouts. Developers can adjust these parameters to satisfy their needs. However, via a console warning, we advise that the interval should be set between 50 to 100ms to have a more accurate recognition of gestures. Moreover, we also advise that the dimension of the buffer should be adapted according to the newly inserted interval. To give more freedom to the developer, we decided not to automatically adapt the dimension of the buffer to the interval.

Since browsers are frequently updated, it is not known if the events will be modified in the future. A more complex architecture could have been implemented to provide a central database that stores thresholds for each case. The database could be updated by the crowd to cover all combinations of browsers, devices and platforms. Although this solution could fix some of the issues mentioned above, it does not solve significant incompatibilities problems such as the deprecation of the APIs or more complex changes in the events.

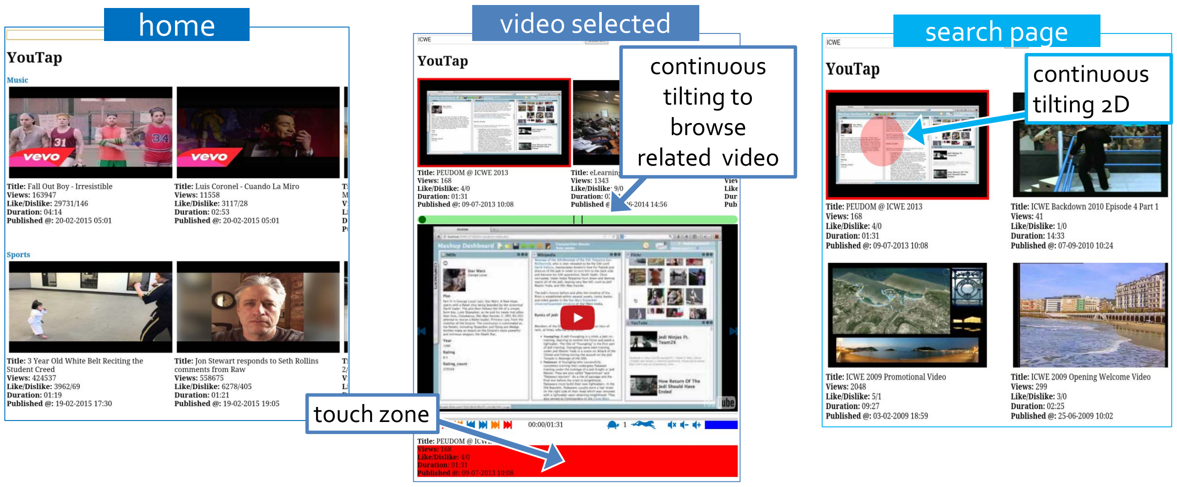


Figure 3.7: Screenshots of the three pages (home, video selected, search page) of YouTap.

3.1.2 Example Applications

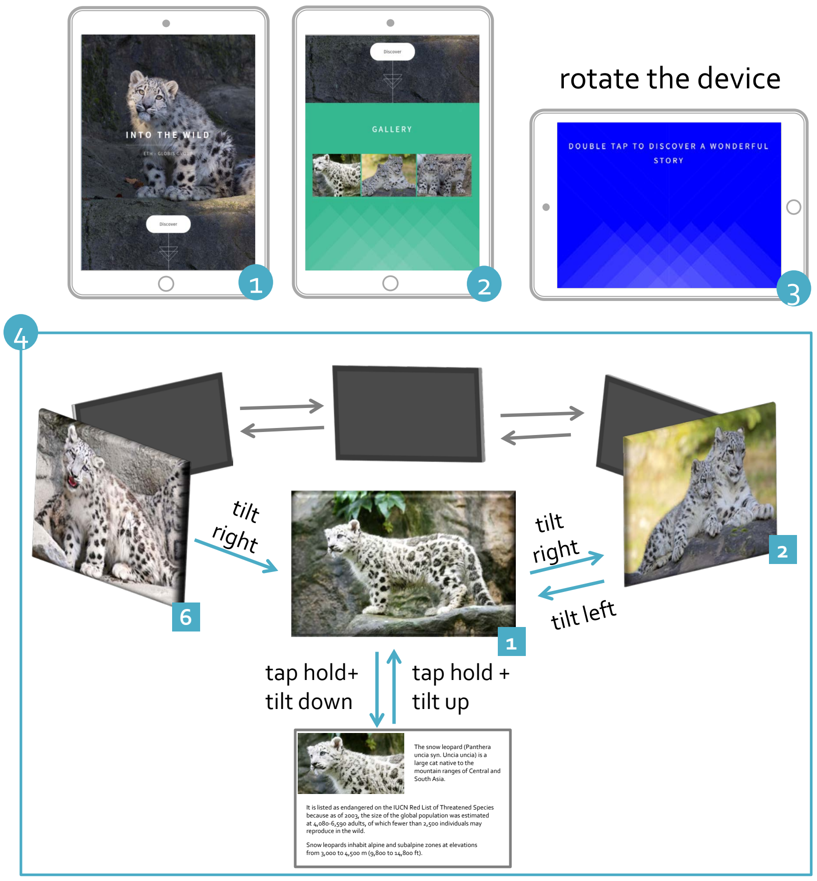
To better understand the power and limitations of our framework, we developed three media-rich web applications that exploit tilting interactions. YouTap, TiltZoo and 3DTiltGallery offered us the chance to experiment with possible adaptations of motion gestures on the web.

As can be seen in Figure 3.7, YouTap is an extended version of the famous YouTube[3] website. The first page of YouTap offers an overview of videos classified by different categories such as music, sport, news etc. Once the user selects one of the videos via tap gestures, the selected media is shown in the middle of the page while related videos are displayed on top. At this point, if no video is played, the user can browse among related media using continuous tilting interactions. A ball shows users the current position in the list. In Figure 3.7, the green ball is below the first item in the gallery of videos. The ball allows the user scroll through the various thumbnails, however, to select a video, users have to tap on the desired item. Alternatively, users can also go to the next or previous videos by holding tap on the red zone and tilting the device rapidly to the right or to the left. All these controls are hidden if the user is watching a video.

While watching, users can perform various gestures to interact with the video directly. For instance, a tilt down or up gesture will turn the volume up or down. Tilt left or right interactions will skip to the next or previous ten seconds of the video. Finally, if the user searches for a video using the search box, YouTap will show the list of results which, this time, can be browsed via two-dimensional continuous tilting gestures. A red ball allows users to hover on the thumbnails and a jerk tilting interaction to the right will cause the video to be played.

We believe that tilting gestures can rich the interactivity of web applications. For this reason, we developed TiltZoo showing that employing motion gestures offered by Tilt-and-Tap can transform a simple gallery website into a creative space (see Figure 3.8). TiltZoo is a one-page web application that can be browsed in two modalities. Users can change modality by moving their device in landscape or portrait mode.

[3]YouTube website: https://www.youtube.com. Accessed on April 2018.



The snow leopard (Panthera uncia syn. Uncia uncia) is a large cat native to the mountain ranges of Central and South Asia.

It is listed as endangered on the IUCN Red List of Threatened Species because as of 2003, the size of the global population was estimated at 4,080-6,590 adults, of which fewer than 2,500 individuals may reproduce in the wild.

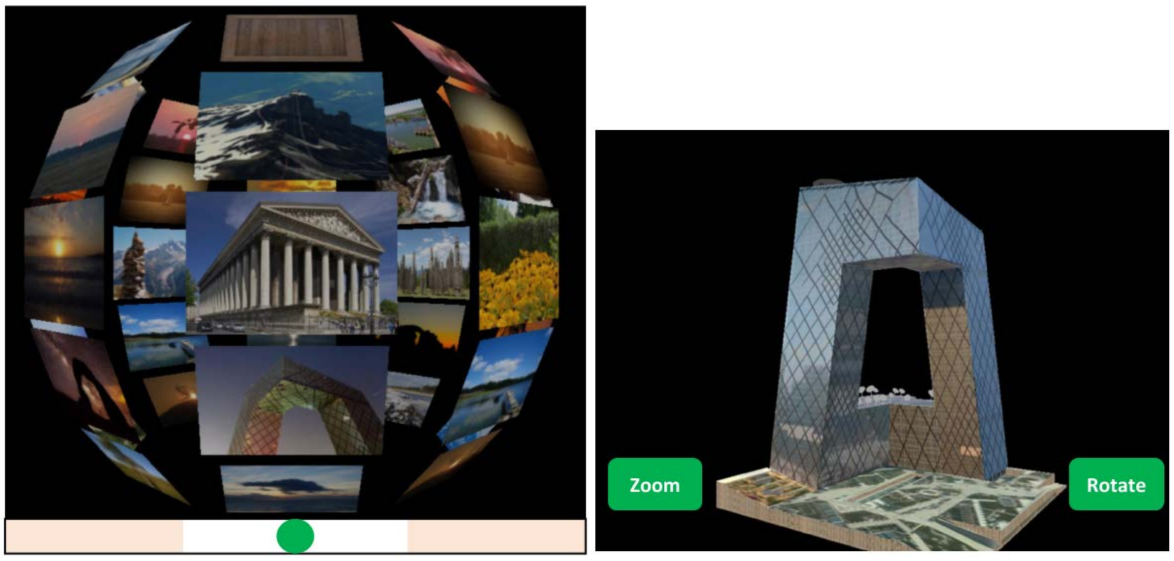
Snow leopards inhabit alpine and subalpine zones at elevations from 3,000 to 4,500 m (9,800 to 14,800 ft).

Figure 3.8: Visual representation of the interaction flow of TiltZoo.

In TiltZoo, when the device is held in portrait mode, we offer a standard browsing experience that displays pictures and content through scrolling and tapping interactions (step one and two in Figure 3.8). If the device is in landscape mode, another modality is triggered: a different UI is shown, and additional gestures are available. In this mode, users can double tap anywhere in the page to enable tilting interactions.

At this point, pictures are shown in full screen and can be browsed through jerk tilting gestures. 3D animations between gestures give the impression that pictures are displayed on a cube. Additional information for each image is available, and they can be triggered by a hold tap gesture on the desired picture and a tilting down interaction. A hold tap and tilt up gesture will return the user to the gallery of pictures. By rotating the device in portrait mode, the system will change modality and allow users to browse the website normally.

Finally, to exploit tilting interactions with 3D objects, we developed 3DTiltGallery, a web application that shows pictures on a sphere. As can be seen in Figure 3.9 (a), images are displayed on the 3D object that can be moved by means of continuous motion gestures. As similarly proposed with YouTap, the ball displayed at the bottom of the page will serve as feedback and make the sphere rotate to the right and the left. When the ball is in the centre of the viewport, the object will not rotate. By tapping on one of the pictures displayed on the sphere, the system will display the 3D model of the selected image in another page. At this point, users can browse the model. We



(a) Home page (b) Model page

Figure 3.9: Screnshots of the 3DTiltGallery.

developed three possible ways of performing rotations and zooming interactions:

1. Horizontal and vertical rotations of the model can be triggered by a combination of hold tap gestures (performed anywhere in the page) and continuous tilting interactions to the right and the left, or up and down. In contrast, to zoom in and out users can hold tap anywhere in the page with two fingers and move the device up and down.
2. Rotate and zoom buttons are displayed in the page (see Figure 3.9 (b)). To perform the desired action, users need to hold tap on the corresponding button and continuously tilt the device in the desired direction.
3. Horizontal and vertical rotations are performed by hold tap gestures and continuous tilting interactions. To change mode and zoom in and out, users need to perform a jerk tilting gesture.

Overall, these applications gave us the chance to exploit tilting gestures on different categories of websites. While TiltZoo and 3DTiltGallery had the goal of making the web a more creative and interactive place, YouTap was developed to extend a famous website with motion gestures. Given the flexibility of TAT, different approaches and combination of interactions could be studied for each of the proposed applications. We believe that Tilt-and-Tap can encourage researchers and, more in general, developers in expanding the set of possible interaction on the web.

3.1.3 Developer Study

We conducted a developer study to evaluate the usability of Tilt-and-Tap and receive feedback on the framework. It is important to note that the study was conducted using the original version of TAT and, therefore, it involved the use of jQuery.

Figure 3.10: Screenshot of the skeleton gallery website.

The study was carried out in 2015 during the Web Engineering class at ETH Zurich. During the course, bachelor and master students learn different web technologies via lectures and group exercises. Students coming from various departments and different levels of education can follow the class. For the exercises, students are awarded a number of points that count toward 25% of the final grade. To evaluate TAT, we asked students to add tilting interactions to a website. The exercise was awarded three points (out of 26 total points given for all exercises) if developed correctly. This exercise was carried out towards the end of the semester when students had already gathered experience with web technologies such as HTML, CSS, JavaScript and jQuery.

Web Engineering students represented the right set of participants for our study. Overall, students have enough knowledge of web technologies to successfully use our framework but, at the same time, they are not professional web developers with years of experience. Good results on the usability of the framework would mean that Tilt-and-Tap should be easy to use for developers that with basic knowledge of web technologies but are not experts.

The TAT exercise involved two major parts. In the first part, students were asked to use the framework to build a predefined website, while the second part, required them to define new gestures using the interactions already proposed by Tilt-and-Tap. As usual for Web Engineering exercises, students had one week to develop their solution before presenting it to the teaching assistants. For this reason, students were free to learn the framework and develop the exercise at their own pace. While this scenario is more realistic than a lab study, we could not observe how students worked on the exercise and which problems they experienced. However, during the presentation of solutions, students filled in a questionnaire.

Study Design and Tasks

We now give details on how the study was carried out. At the end of a lecture, one assistant of Web Engineering instructed the students on the exercise and explained the main concepts of Tilt-and-Tap. After the introduction, the students had one week to develop the exercise in groups. As commonly done for all Web Engineering exercises, students were also allowed to split the work among group members as desired.

Students were provided with a minified version of the framework (the TAT JavaScript file without unnecessary characters), a wiki on the API and several demo examples. On the group website, participants could also download a detailed description of the exercise, all the necessary material and a skeleton of the gallery website (see Figure 3.10) from which they could implement the three required **tasks**:

- **Task A**: In this task, students were required to add jerk tilting interactions to browse through the gallery. A tilt gesture to the right or the left will select the next or previous picture in the gallery (starting from the first one on the left) An image is selected when a red border is displayed on the picture.

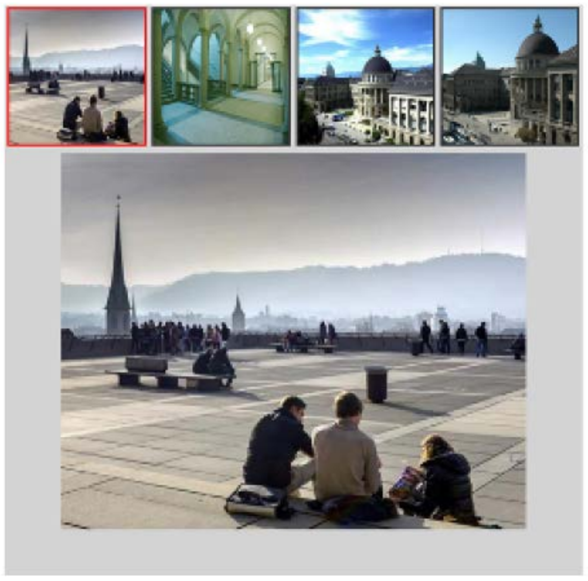
- **Task B**: At this point, students also had to employ combinations of touch and motion interactions. By hold tapping on an area of the page and tilting the device up, the current selected picture should be displayed in the middle of the page in a higher resolution. In this state, if the device is tilted to the right or the left and no touch interactions are performed in the page, the previous or next picture will be selected, and the image in the page will change accordingly. By tilting the device down and performing a hold tap interaction on the page, this view will be closed (see Figure 3.11 (a)).
- **Task C**: While in the first two tasks students had to use the framework directly, in task C participants were required to implement an additional gesture not originally by Tilt-and-Tap. The gesture was called `onTiltUpDown` and, as its name suggests, it requires the user to tilt the device up and down rapidly. Once this gesture is triggered, additional information on the current selected picture should be shown in the page (see Figure 3.11 (b)). To later hide this information, the gesture needs to be performed again. Similar to Task B, if the user changes the currently selected picture with jerk tilting interaction, the displayed information should adapt accordingly.

If teams were able to implement all tasks of the exercise successfully, the assistants would award them three points. If some parts of the solution were incomplete or wrong, points were deducted in accordance with the gravity of the mistake. Moreover, during the assessment, students were required to describe how they solved the exercise and discuss the challenges they faced during the development.

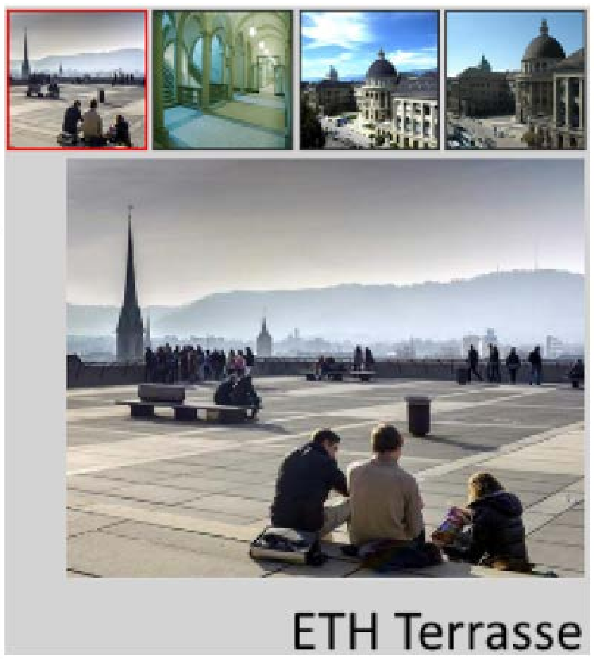
Results

In 2015, 93 students were registered to the exercises of Web Engineering. With a total of 32 groups: 30 groups had three members, one group had two members and one student worked alone for personal reasons.

During the day of the assessment, all 32 groups had at least one member to present the proposed solution. On the day of the assessment 77 students were present and discussed their solutions with one of the three assistants. Their solution were tested on the students' phones or tablets which allowed us to test the framework on a broad set of browser and platform combinations. Overall, all but one team received full points for the exercise. The exception was a group of students who failed to complete task C and said this was due to the insufficient time. While the exercises were evaluated per group, questionnaires were answered by individual students. In total, 48 students filled in the final survey. For anonymity reasons, we did not associate the answers of the questionnaire with the solutions of the participants. 42 of those who answered the questionnaires were male and 6 female. Their ages ranged between 22 and 31 years (avg. 24.93 std. 2.04). In Figure 3.12, we show how students evaluated their knowledge of web technologies (HTML, CSS, JavaScript and jQuery) on a scale from 1 (novice) to 7 (expert). On average, students estimated that they had intermediary knowledge of all the proposed languages. Since most of the students learned these technologies only



(a) Task a



(b) Task b

Figure 3.11: Screnshots of the TAT exercise.

during the Web Engineering class, we expected and aimed for such results. 32 out of 48 participants were computer science students while the remaning 14 participants came from other departments including electrical engineering and management students.

30 participants stated that they worked on all three tasks, nine worked on a single task and the remaining nine students developed some combination of two tasks. On average, all participants needed four hours to develop their parts of the exercise (std. 2.3 hours). Students that worked only on one task needed 3 hours. These results are based on student estimations expressed in the final survey and involve the time they needed to understand the exercise and the plugin as well as the actual development of the task and debug phase.

In the final questionnaire, we asked students which combination of device, platform and browser they used to develop the exercise. We note that participants used more than one device and browser to debug their solution. 38 out of 48 students used Android devices such as Samsung phones (s3, s4 and s5) or Nexus devices (Nexus 7 and 5). iOS was the second most common platform with 9 students working on iPhones or iPads. Only one student used a Windows Phone. Moreover, one participant exploited a convertible laptop to test and develop his application with one device. Chrome was the most popular browser with 38 students using it during the development of the tasks. 9 students used Safari and 2 Firefox.

To better understand what challenges participants had to face, we asked where they spent most of their effort. 24 students (50%) stated that most of their effort was spent in the actual development of the exercise. In contrast, 18 participants (38%) said that learning how to use the plugin was the most time consuming task to perform. Only six students (12%) felt that adjusting thresholds took most of their time.

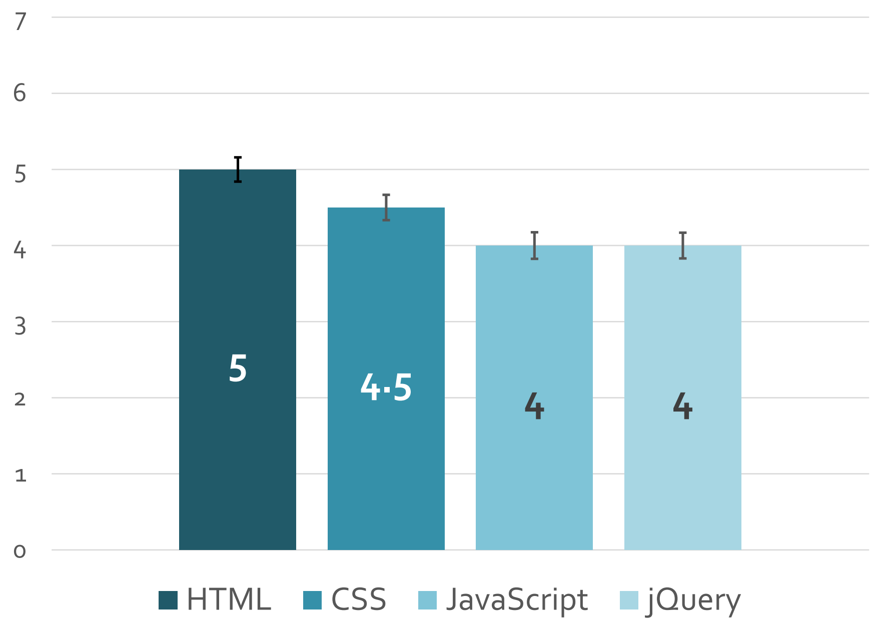


Figure 3.12: Students' estimated knowledge of web technologies. Median values. Error bars represent standard error.

Finally, in Figure 3.13, we can see how participants evaluated the framework in terms of easiness to use and to learn in a scale from one (hard) to seven (easy). On average, students agreed that the framework was usable and they did not have any significant challenges in learning it.

Discussion

The developer study gave us the chance to gather feedback on the framework and test our solution on a broader set of combination of devices and browsers. Moreover, some of the feedback received during the study was used to infer further improvements of Tilt-and-Tap and its wiki.

Although most students had only intermediate knowledge of jQuery, the majority of our participants successfully completed the exercise and were able to discuss their solution with the assistants. However, students also reported some issues they experienced during the development of the tasks. The main concern that students expressed in the questionnaire and during the assessment were related to thresholds. While they believe that the framework was well structured and easy to use, in some cases, it was necessary to adjust parameters to better detect gestures. Although TAT was developed to cater for a broad spectrum of devices and platforms, some combinations were not yet thoroughly tested before the developer study. Moreover, in 2015, some devices did not support motion events on any browsers (e.g. Windows Phones and Nexus 7 tablets). In a later version of the framework, we decided to improve the recognition of jerk tilting interactions and offer a more consistent implementation among browsers and platforms (see Section 3.1.1).

Regarding the design of the study, we believe that a group exercise without the presence of an external observer is a good setup to test a framework. This type of study gave us the chance to have more participants than a lab study while simulating

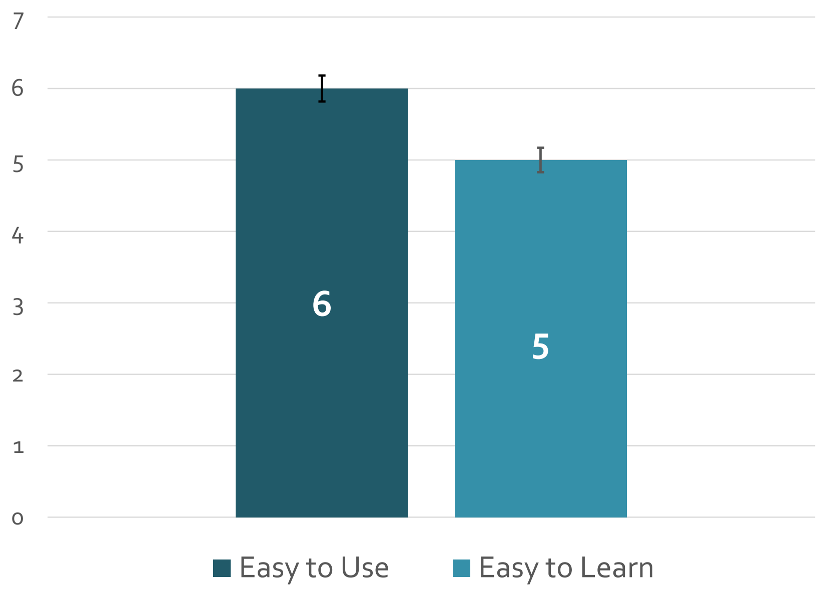


Figure 3.13: Participants ratings of Tilt-and-Tap. Median values. Error bars represent standard error.

a real case scenario. Web developers often work in small teams and use wikis and examples to learn a new technology [160]. However, we could not study the learning curve of the framework, and it was not possible to see how much a single member of a group contributed to the exercise. On the other hand, in a lab study, the length and complexity of tasks must be limited for time reasons. Moreover, less experienced developers could feel intimidated by the researcher while programming.

3.1.4 Competition Study

After participants presented the exercise discussed in Section 3.1.3, Web Engineering students had the opportunity to participate in a competition study. Students were required to design and develop an application using our framework. The author of the best solution was awarded with an iPad. The goal of the study was to evaluate the power of Tilt-and-Tap in improving creativity in web development.

After the assessment of the TAT exercise, one assistant described the goal of the competition to Web Engineering students and published the rules and a description of the competition online. Students had one month to submit their solution to the assistant. After the deadline, participants were required to show their web application and justify design choices to a jury composed by assistant and professors. Later, the participants presented their solutions to the students of the course. Participation in the competition was not mandatory, and it did not count toward the final grade of the class. Similarly to the developer study, Web Engineering students were the perfect candidates for the study since they were already acquainted with the framework and with web technologies.

We received a total of three submissions that we named as follows: Tilt5, TiltEarth and MenuTilt. Tilt5 and TiltEarth applied continuous and jerk motion gestures to games and simulation web applications. With Tilt5 users, could control a character

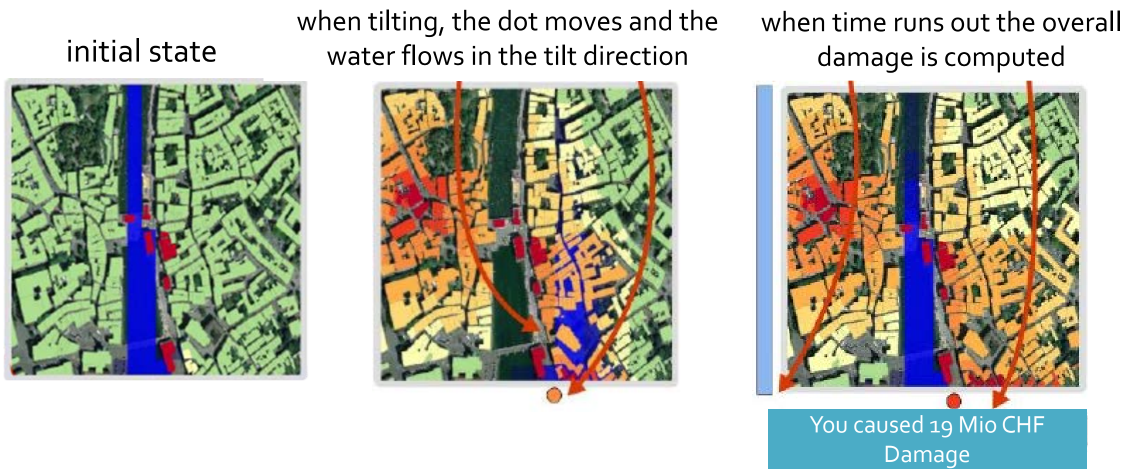

initial state
when tilting, the dot moves and the water flows in the tilt direction
when time runs out the overall damage is computed
You caused 19 Mio CHF Damage

Figure 3.14: Screnshots of TiltEarth.

in a racing game by moving their device continuously to the left and the right. Users could jump using a rapid movement of the device upwards.

In TiltEarth, users could simulate an earthquake by shaking their device (see Figure 3.14). Continuous tilting gestures would infer the power and movement of the seism and cause a flood of the lake displayed in the page. In TiltEarth and Tilt5, all motion interactions were implemented via TAT while the logic of the application was developed on the client via HTML, CSS and JavaScript languages.

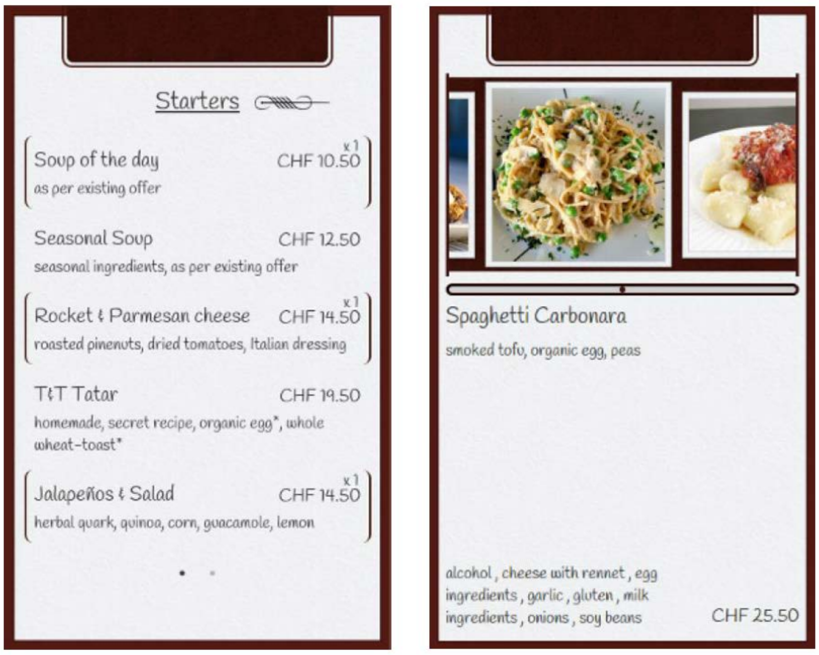
Finally, MenuTilt represents a menu of a restaurant that can be browsed via tilting interactions (see Figure 3.15). The goal of MenuTilt was to offer an experience similar to a real physical menu while also presenting some innovative interactions to make the experience more memorable. Jerk motion gestures and appropriate visual animations simulate the browsing experience of a physical book. Dishes can be ordered by performing a hold tap gesture on the corresponding item and tilting the device up and down. When the user performs a counterclockwise jerk interaction, pictures of the dishes will be displayed. These images can be browsed via continuous tilting gestures. MenuTilt employed Tilt-and-Tap for the recognition of motion interactions, and AngularJS[4] for front-end development.

The jury, unanimously, awarded MenuTilt as the winner since not only was the most complex application, but it was also created *around* motion gestures to create an interactive and memorable experience.

3.2 Tilt-and-Tap 2.0

With TAT we were able to initiate our research in enlarging the set of possible interactions on the web by supporting tilting gestures. Although TAT offered the two most common motion interactions (jerk and continuous), other variants of tilting gestures were not taken into consideration. For instance, TAT does not differentiate between *position* and *velocity-based* implementations. With *velocity-based* continuous interac-

[4]AngularJS website: https://angularjs.org. Accessed on April 2018.


Starters

Soup of the day x1 CHF 10.50
as per existing offer

Seasonal Soup CHF 12.50
seasonal ingredients, as per existing offer

Rocket & Parmesan cheese x1 CHF 14.50
roasted pinenuts, dried tomatoes, Italian dressing

T&T Tatar CHF 19.50
homemade, secret recipe, organic egg*, whole wheat-toast*

Jalapeños & Salad x1 CHF 14.50
herbal quark, quinoa, corn, guacamole, lemon

Spaghetti Carbonara
smoked tofu, organic egg, peas

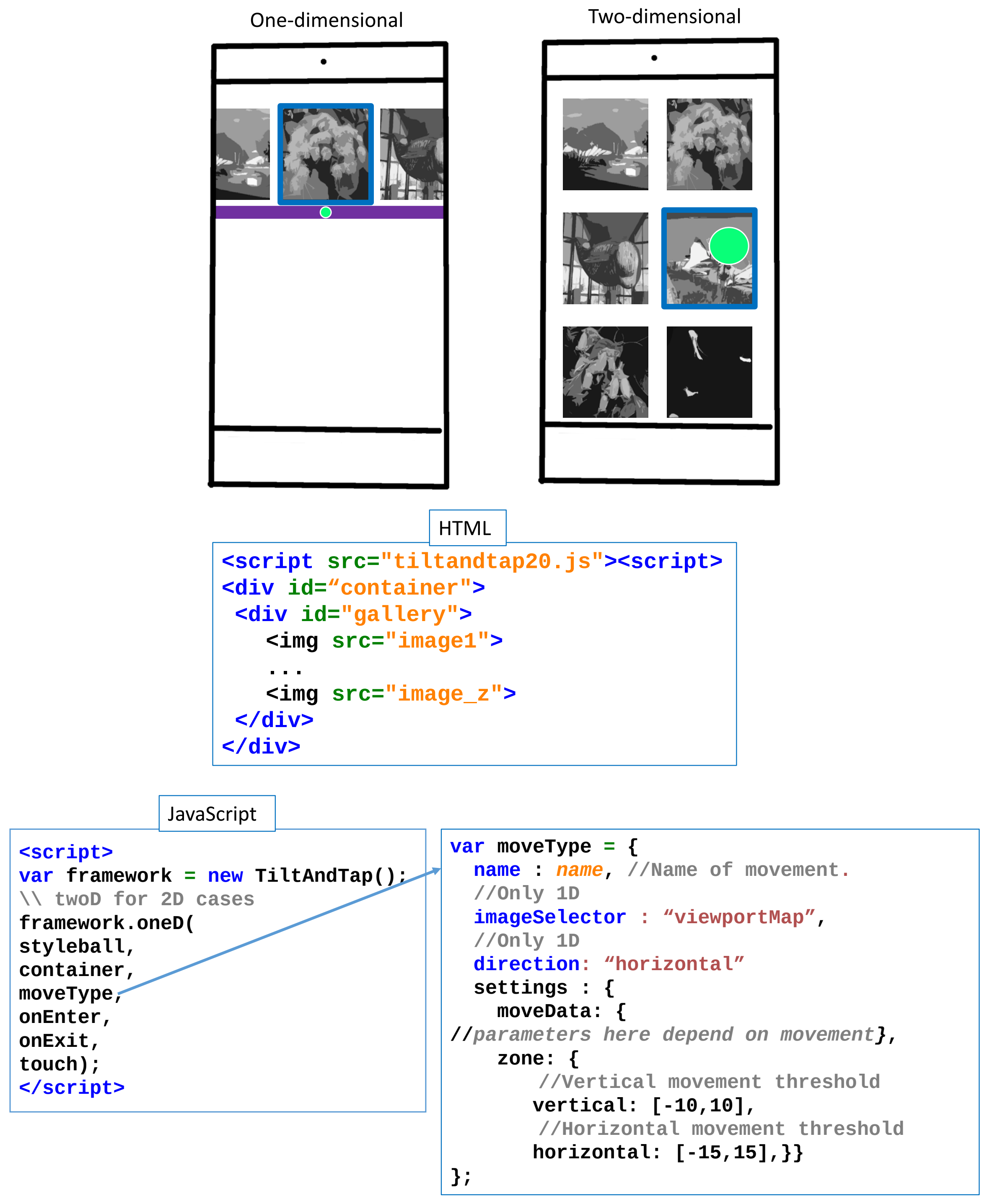
alcohol, cheese with rennet, egg ingredients, garlic, gluten, milk ingredients, onions, soy beans CHF 25.50

Figure 3.15: Screnshots of MenuTilt.

tions, the more the device is tilted, the faster the system will scroll a list of elements. In contrast, in *position-based* solutions there is a direct mapping between the orientation of the device and the position of a cursor on the screen. As stated by Teather and MacKenzie [205], both approaches take into consideration the orientation of the device but yield two different user experiences. Moreover, both techniques can be influenced by a number of parameters such as the speed of the movement and possible triggers to activate the gesture. For these reasons, we extended TAT to support a broader set of possible continuous motion gestures to understand which tilting interaction is most suited to the web and in which context. This second version of our framework is called Tilt-and-Tap 2.0. In Section 3.2.1, we start by discussing the main concepts behind TAT 2.0 and present implementation details of each variant we proposed. In Section 3.2.2 we present the architecture of TAT 2.0 Finally, in Section 3.2.3 we conclude with some final remarks.

3.2.1 Concepts and Example Usage

Similar to TAT, the extended version supports one-dimensional and two-dimension- al continuous motion gestures. In one-dimensional cases, users move the device vertically or horizontally and scroll the viewport only in the corresponding axes. In contrast, in two-dimensional cases, an indicator will move in the entire viewport allowing the user to browse the web page on the vertical and horizontal axis. In Figure 3.16, we can see an example use of TAT 2.0 for 1D and 2D cases. In both examples, continuous tilting gestures are used to browse and select pictures. In the one-dimensional case, pictures are displayed in a horizontal one-dimensional list. Instead, in the two-dimensional case, images are in a grid across the entire web page. In both scenarios, when the cursor is inside an element, a blue border will be displayed.



HTML

```
<script src="tiltandtap20.js"><script>
<div id=“container">
 <div id="gallery">
    <img src="image1">
    ...
    <img src="image_z">
 </div>
</div>
```

JavaScript

```
<script>
var framework = new TiltAndTap();
\\ twoD for 2D cases
framework.oneD(
styleball,
container,
moveType,
onEnter,
onExit,
touch);
</script>
```

```
var moveType = {
  name : name, //Name of movement.
  //Only 1D
  imageSelector : “viewportMap”,
  //Only 1D
  direction: “horizontal”
  settings : {
    moveData: {
//parameters here depend on movement},
    zone: {
        //Vertical movement threshold
       vertical: [-10,10],
        //Horizontal movement threshold
       horizontal: [-15,15],}}
};
```

Figure 3.16: Example use of TAT 2.0 for one dimensional and two dimensional scenarios. Developers can specify the desired case by using the right keyword (`oneD` or `twoD`).

Movement Type	Dimension
Constant Move	1 and 2D
Static Acceleration	1 and 2D
Dynamic Acceleration	1 and 2D
Mapped Container	1 and 2D
Balance Board	1 and 2D
Two Zones	1D
Map Slider	1D

Table 3.1: List of Movement Types and their supported dimensional cases.

To reproduce this behaviour, TAT 2.0 has to be included in the web application, and developers need to create an instance of the framework in a JS script as shown in Figure 3.16. At this point, a number of parameters have to be passed to the TAT 2.0 instance such as the callback function to be called once the ball selects an de-select an element (`onenter` and `onexit` variables), the style of the indicator (`styleball`), eventual combination of touch interactions (passed as Strings via the touch `parameter`) and the `moveType` variable. The `moveType` variable represents the type of movement developers want for their applications. TAT 2.0 supports a number of different variants of continuous tilting gestures for both one-dimensional and two-dimensional cases. We call these solutions *movement types* which differ in terms of offered user experience. In Table 3.1, we list the supported movement types. Some movements are shared between 1D and 2D scenarios while others are available only for one-dimensional cases. We discuss in detail each of these movements and the corresponding `moveType` parameter of TAT 2.0 in the following sections.

Independent of the type of movement and by whether 1D or 2D, the `moveType` variable requires the specification of the following parameters: the `name` of the movement type in a String format and the specification of the ***dead zone***. We define the *dead zone*, as the orientation of the device in which no movements will be triggered. In the example of Figure 3.16, movements of the device that happen between -10 and 10 degrees on the vertical axis will not be recognised as a movement by the framework in which case the indicator will remain in its position.

TAT 2.0 allows developers to decide which element the indicator will select depending on its position. Figure 3.17 shows an abstract representation of the three approaches supported by the framework. In 1D cases, developers can decide to place the indicator at the centre of the device and always select the image in the middle of the viewport (see Figure 3.17 (a)). This can be defined by setting the parameter `imageSelector` to `centerSelector`. By default, this parameter is set to `viewportMap`. In this condition, the ball is free to move and its position infers the which element is selected among only the ones currently visible in the viewport (see Figure 3.17 (b)). We also support the `imageMap` solution where we partition the slider where the ball into pieces. Each of these pieces corresponds to every element in the gallery. Depending on which partition the ball is in we determine the selected image (see Figure 3.17 (c)). It is important to note, that if developers want to use the `imageMap` solution, only a subset of movements are available for this option. We discuss details about this context in the following sections. The axis of the movement can be chosen by the `direction` parameter. This variable can be set to **horizontal** or **vertical** depending on developers needs.

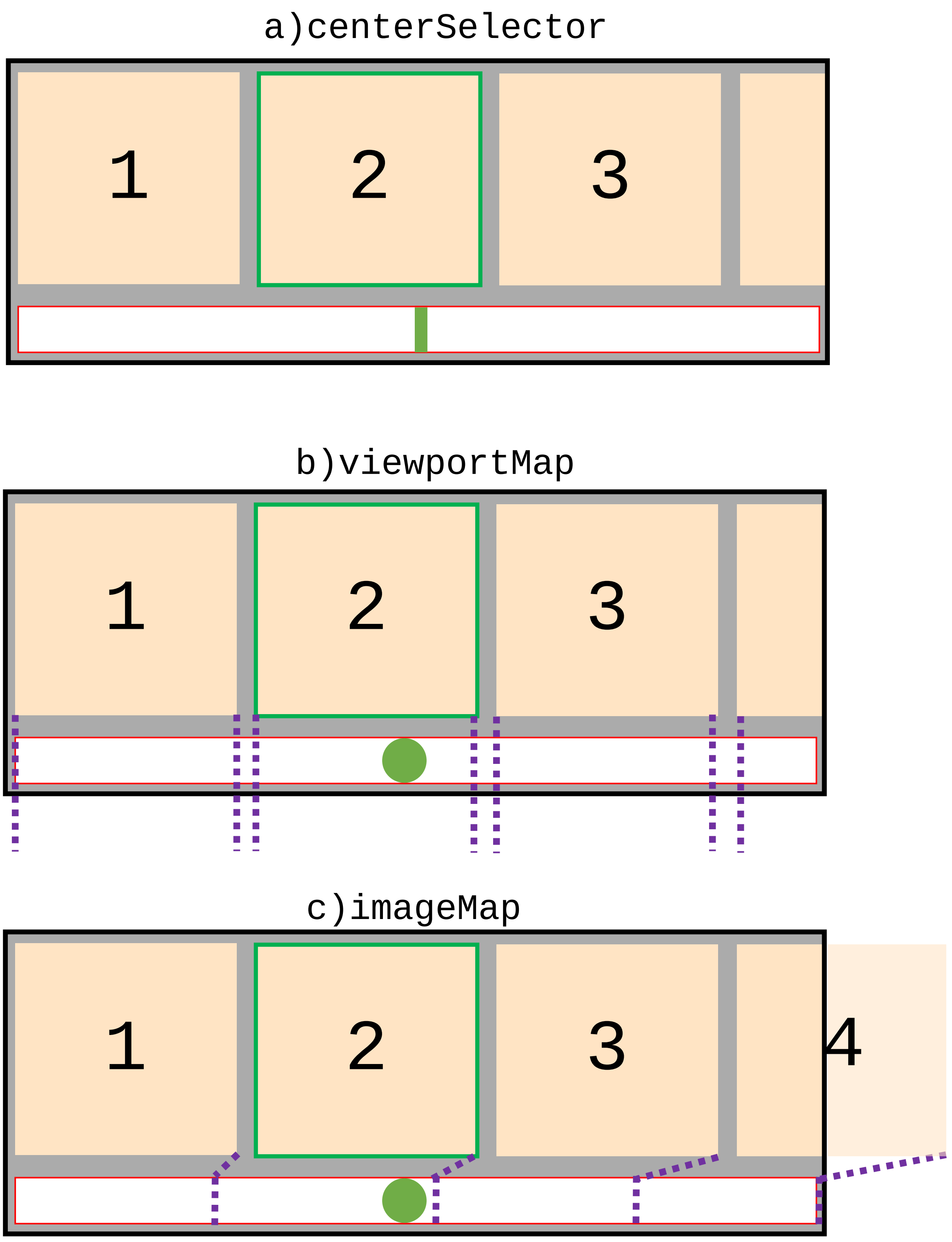
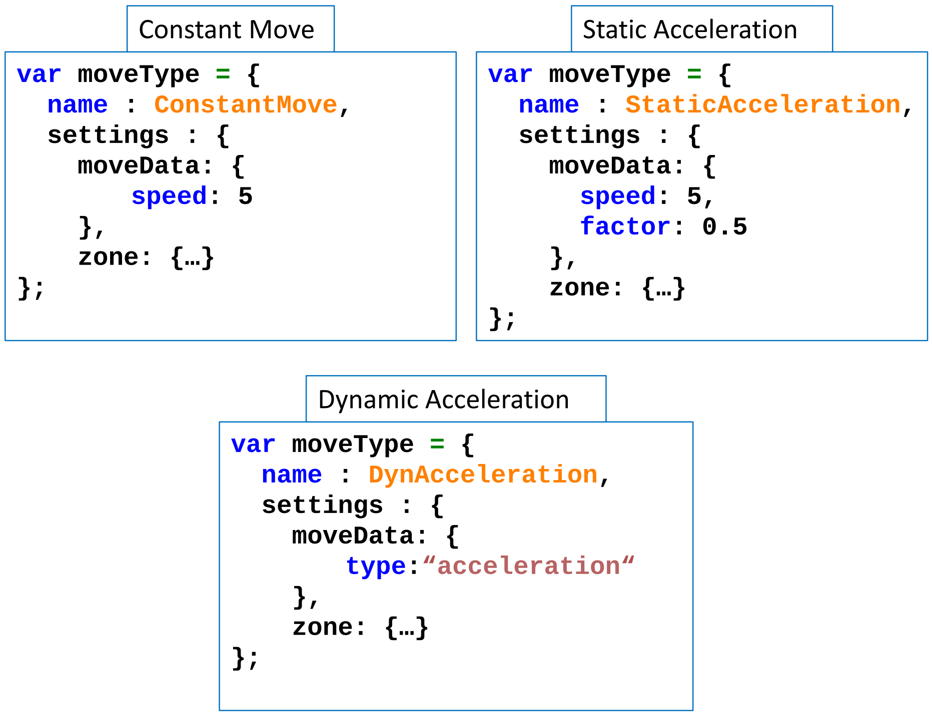


Figure 3.17: Abstract representation of the three selection solutions supported by TAT 2.0.



Constant Move

```
var moveType = {
  name : ConstantMove,
  settings : {
    moveData: {
        speed: 5
    },
    zone: {…}
};
```

Static Acceleration

```
var moveType = {
  name : StaticAcceleration,
  settings : {
    moveData: {
      speed: 5,
      factor: 0.5
    },
    zone: {…}
};
```

Dynamic Acceleration

```
var moveType = {
  name : DynAcceleration,
  settings : {
    moveData: {
        type:“acceleration“
    },
    zone: {…}
};
```

Figure 3.18: Code snippet of the `moveType` variable.

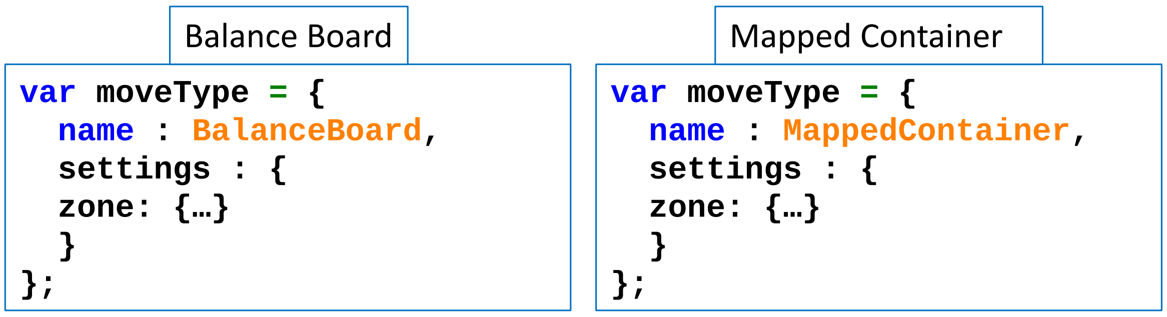
Constant Move, Static Acceleration and Dynamic Acceleration

The Constant Move, Static Acceleration and Dynamic Acceleration movement types can be used in 1 and 2D cases. In Figure 3.18, we can see how developers can use these three solutions. As the name suggests, in Constant Move we use a constant speed when outside a dead zone. In this solution, the orientation of the device only infers where the movement of the viewport and the ball (if any indicators are involved) will go and not their speed which is fixed and predefined by the developers. The speed can be set via its corresponding parameter `speed`. This variable must be an integer number from 1 to 100. The bigger the `speed` parameter is, the faster the movement will be.

Conceptually, we applied this type of motion gestures to the action of pressing physical arrow-keys on a keyboard. Tilting the device to the right corresponds to pressing the *right key* down instead, moving the device in its initial space inside the *dead zone*, corresponds to a key up event. For this reason, in 2D dimensional scenarios, these three movements will move the ball only in eight directions: up, down, right, left, north-east and west, south-east and west.

With the Static Acceleration movement type, when outside the dead zone, we apply an acceleration factor to the speed defined by the developer (see Figure 3.18). Also in this case, the orientation of the device only decides the direction of the movement and not its speed. In contrast with Constant Move, with the Static Acceleration movement type, over time, we speed up the movement of the ball by a factor defined by the developer. Also in this case, the ball can only move in above mentioned eight directions.

Finally, with the Dynamic Acceleration movement type, we directly map the data received from the sensors to the movement of the viewport and the ball. In this case, developers can decide to use the `acceleration` data or the `accelerationIncluding-`



```
var moveType = {
  name : BalanceBoard,
  settings : {
  zone: {…}
  }
};
```

```
var moveType = {
  name : MappedContainer,
  settings : {
  zone: {…}
  }
};
```

Figure 3.19: Code snippet of the `moveType` variable.

`Gravity` returned by the `DeviceMotionChange` event. In contrast with Static Acceleration, the Dynamic Acceleration speeds the ball depending on how fast the user tilts the device. Also with this movement type, the ball is free to move only in eight directions.

Balance Board and Mapped Container

The Balance Board movement is inspired by the Marble Maze game[5]. In this game, a marble is positioned in a box with holes and obstacles. Users can move the ball in the box by tilting the container with a knob. The goal of the game is to move the ball in the box to reach an end point. In our context, the device is the box that the user can tilt by moving it in the desired direction. As in the Marble game, the ball will move faster if the device is tilted in a rapid manner. Balance Board allows the indicator to move in any direction depending on the orientation of the device. Our Balance Board solution was inspired by the velocity-based tilting interaction proposed by Teather and MacKenzie [205]. In contrast to their solution, we do not require the user to hold the device parallel to the ground. In our approach, the user is free to interact with the device in any holding position they desire.

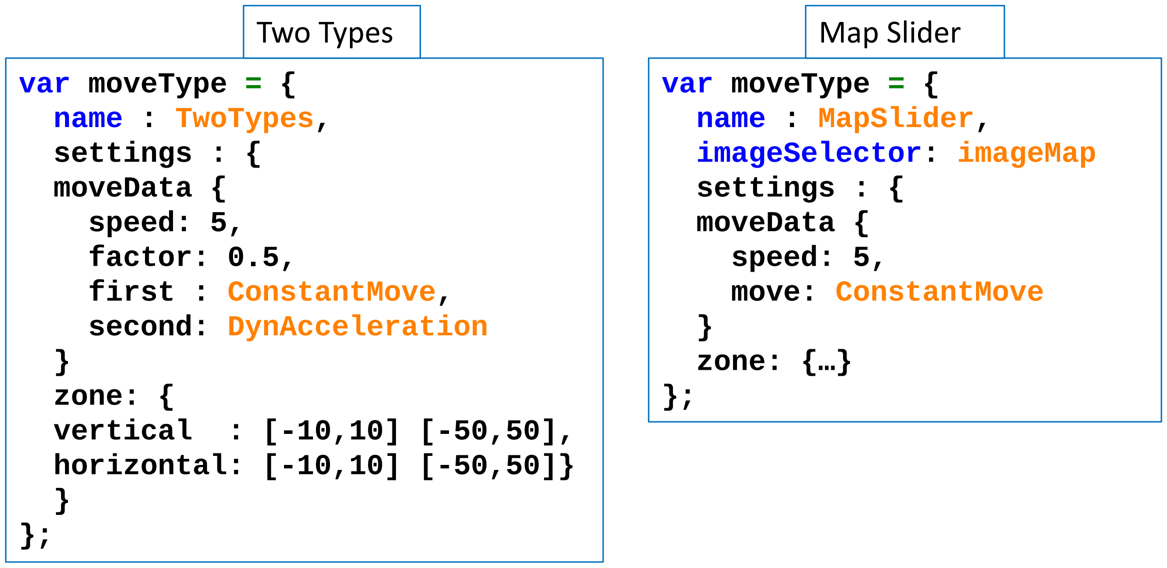
We support position-based solutions for continuous tilting interactions with the Mapped Container. Where the orientation of the device to a specific position of the indicator in the box. Also with the Mapped Container movement type, the ball is free to move in any direction similar to Balance Board. However, the two movements offer different user experiences. While with the Balance Board movement the ball continuously moves from one position to another, with the Mapped Container the ball *jumps* from one position to another depending on the orientation of the device. In 1D scenarios, developers can use the Mapped Cointainer movement only if they set the parameter `imageSelector` to `imageMap`. In Figure 3.19 we can see the code that developers needs to specify to use the two movements.

Two Zones and Map Slider

The Two Types movement type allows developers to combine two different movements. The viewport and the ball (if present) will move using the first type of movement defined by the variable, until passing a threshold (see Figure 3.20). At this point, the movement behaviour will change following the second desired movement type. In one-dimensional

[5]Marble Maze game video: https://goo.gl/UDM8Tb. Accessed on May 2018.

cases, if the developer has set the parameter `imageSelector` to `imageMap`, the Mapped Slider movement becomes available and can be employed as shown in Figure 3.20. Every position of the ball is mapped to a specific element in the list and the movement of the ball is defined by the movement type passed in the `moveName` parameter. In the example shown in Figure 3.20, the desired solution is Constant Move.


Two Types
Map Slider

```
var moveType = {
  name : TwoTypes,
  settings : {
  moveData {
    speed: 5,
    factor: 0.5,
    first : ConstantMove,
    second: DynAcceleration
  }
  zone: {
  vertical  : [-10,10] [-50,50],
  horizontal: [-10,10] [-50,50]}
  }
};
```

```
var moveType = {
  name : MapSlider,
  imageSelector: imageMap
  settings : {
  moveData {
    speed: 5,
    move: ConstantMove
  }
  zone: {…}
};
```

Figure 3.20: Code snippet of the `moveType` variable.

3.2.2 Architecture

One goal of TAT 2.0 was to support a broad set of interactions and allow developers to further extend these continuous motion gestures in the future. For these reasons, the framework was developed to have high maintainability by employing a strategy design pattern. We identified five different modules with different responsibilities (see Figure 3.21).

The `Controller` module performs preprocessing tasks such as filtering data from the sensors, creating the necessary data structure to allow the interactions and keeping track of elements influenced by orientation events. Moreover, the `Controller` checks if an indicator is used, checks if it moves within its boundaries and executes the `onEnter` and `onExit` functions when an element is selected.

The `MotionEvents` component manages the motion and touch event listeners and handles the calibration by setting as the initial state of the device its first orientation. `MotionEvents` also handles the initialization of the various movements by using the function `setMoveType` which processes the `moveType` data defined by the developer. The class `MoveType` is extended by the two modules `Movements1D` and `Movements2D` that defines the behaviour of the various movement for 1 and 2D scenarios.

This architecture gave us the chance to add new movement types to the framework easily. In this solution, movement types are decoupled by other responsibilities and independent from the application from the application in use.

Also Tilt-and-Tap 2.0 does not need jQuery or any other plugins and it is implemented by using only plain JavaScript.

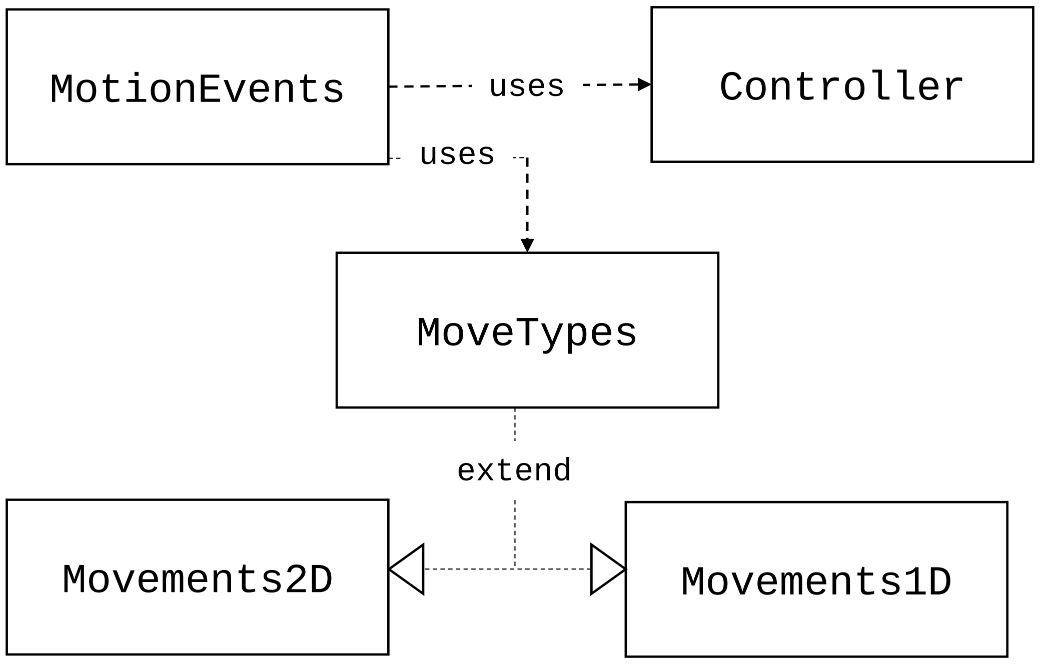


Figure 3.21: Architecture of TAT 2.0.

3.2.3 Discussion

In contrast to TAT, Tilt-and-Tap 2.0 offers a broader set of continuous tilting gestures. While we did not carry out an evaluation study in single device scenarios, in Section 4.3 we present a user study where we evaluated the gestures supported by TAT 2.0 in cross-device environments. This in contrast with previous works that mainly studied these interactions only to control a single smartphone.

We recognise that TAT 2.0 is more complex to use than its predecessor. However, the extended framework offers a higher number of interactions that can be influenced by a multitude of parameters. Although the number of combinations can feel overwhelming, the majority of the available options can be omitted. If developers are keen to experiment with alternatives they are free to adjust the parameters as desired, however, these options are not mandatory. The framework supports default behaviours for each movement type in both dimensional cases. For this reason, we believe that TAT offers a good trade-off between the number of available features and its usability.

As discussed in this section, movement types were inspired by previous works in the field of continuous tilting gestures [205, 23, 179]. These solutions were empirically tested on the web and then adapted to improve their user experience. Moreover, initially, the framework supported a larger number of interactions that, after empirical tests, were later discarded. For instance, the Two Zones movement type was also applied in two-dimensional cases. We discarded this approach since we found it particularly cumbersome to perform in 2D scenarios and, therefore, it was not included in the final set of available interactions.

3.3 WP-TAT

TAT and TAT 2.0 represented the first steps in enlarging the set of possible motion-based interactions on the web. The main goal of these projects was to encourage developers to add tilting interactions on their web applications by offering a high-level set of easy to use APIs that manage sensor data and deal with browser incompatibility issues. As a next step, we also wanted to support end-users by providing a visual tool for the rapid customisation of motion interactions on their web applications. To reach this goal, we developed WP-TAT, a WordPress extension that enables users with no programming skills to integrate combinations of touch and tilting gestures in their WordPress website.

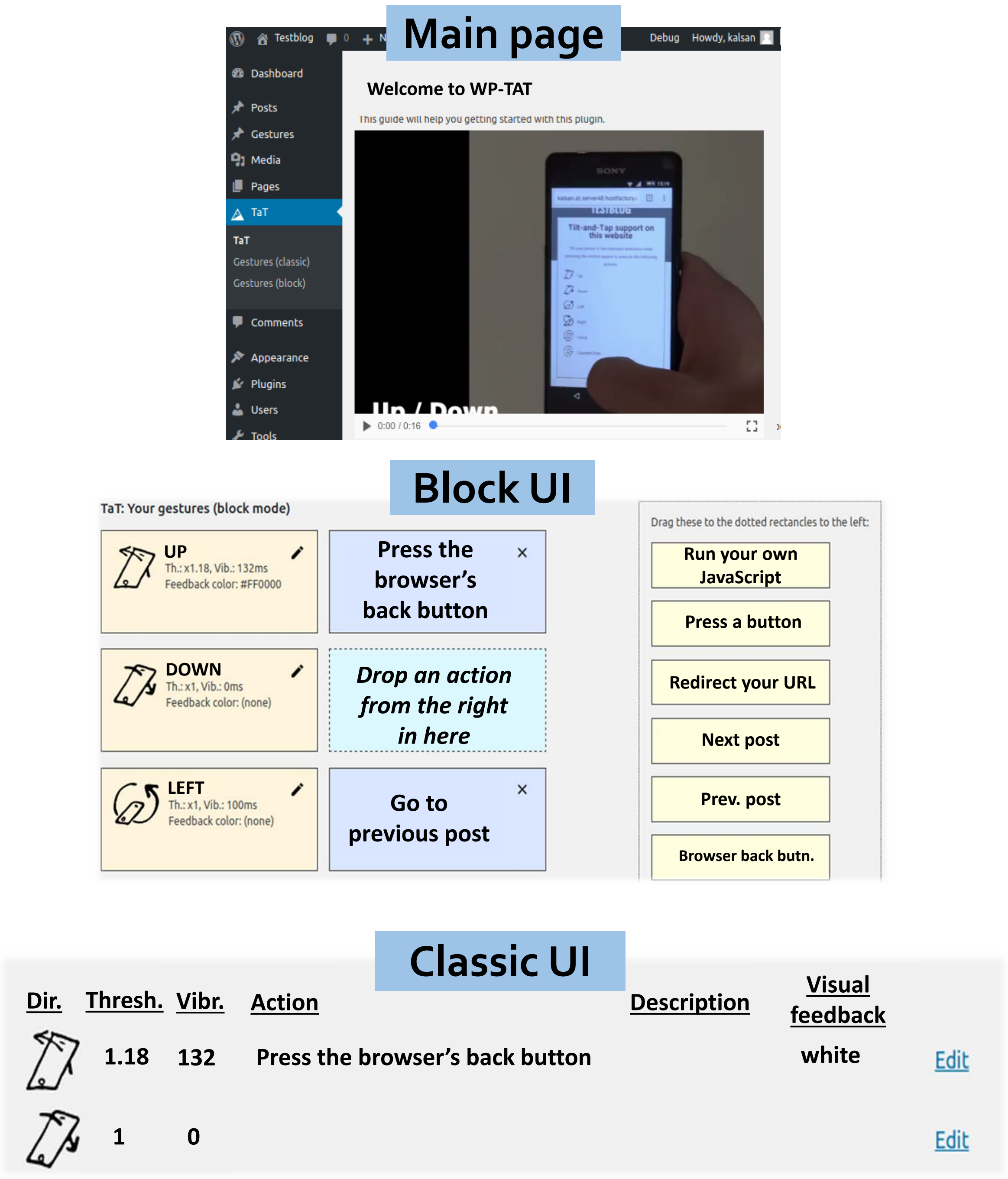
In Section 3.3.1, we start by discussing the main concepts of WP-TAT. Details about the architecture and implementation of the WordPress extension will be presented in Section 3.3.2. In Section 3.3.3, we discuss a number of example applications developed using WP-TAT. To better understand the usability and power of the tool, we conducted a preliminary user study that we present in Section 3.3.4. Finally, we provide some concluding remarks in Section 3.3.5.

3.3.1 The Tool

Although WordPress offers users the opportunity to set up a simple website quickly, the CMS does not offer the opportunity to add any alternative forms of interaction. To bridge this gap, we developed WP-TAT, a WordPress extension that allows end-users to experiment with tilting gestures in their website. With WP-TAT, users can associate *global actions* to motion gestures via a visual interface. We define global actions as actions that globally affect the entire website. For example, a tilt of the device to the right or to the left can redirect the user to the next or previous post of the website. Currently, WP-TAT supports the following global actions:

1. Run custom JavaScript
2. Press a button
3. Redirect to URL
4. Go to next or previous page or post
5. Press the browser back button
6. Search Google for user-selected text
7. Search Google Maps for the user-selected text

Users can associate each global action to one or more jerk motion gestures. WP-TAT supports all ten jerk tilting interactions available in TAT and all their customizable options such as combinations of touch gestures, threshold levels, visual, vibration and sound feedback. Once installed, WP-TAT can accessed via the WordPress dashboard. In the main page of WP-TAT, a summary of the features offered by the tool is shown to the user. Moreover, a video presenting jerk tilting gestures is also available in this page (see Figure 3.22).



Welcome to WP-TAT

This guide will help you getting started with this plugin.

TaT: Your gestures (block mode)

UP
Th.: x1.18, Vib.: 132ms
Feedback color: #FF0000

Press the browser's back button

DOWN
Th.: x1, Vib.: 0ms
Feedback color: (none)

Drop an action from the right in here

LEFT
Th.: x1, Vib.: 100ms
Feedback color: (none)

Go to previous post

Drag these to the dotted rectancles to the left:

Run your own JavaScript

Press a button

Redirect your URL

Next post

Prev. post

Browser back butn.

Dir.	Thresh.	Vibr.	Action	Description	Visual feedback	
	1.18	132	Press the browser's back button		white	Edit
	1	0				Edit

Figure 3.22: Partial screenshots of the WP-TAT tool.

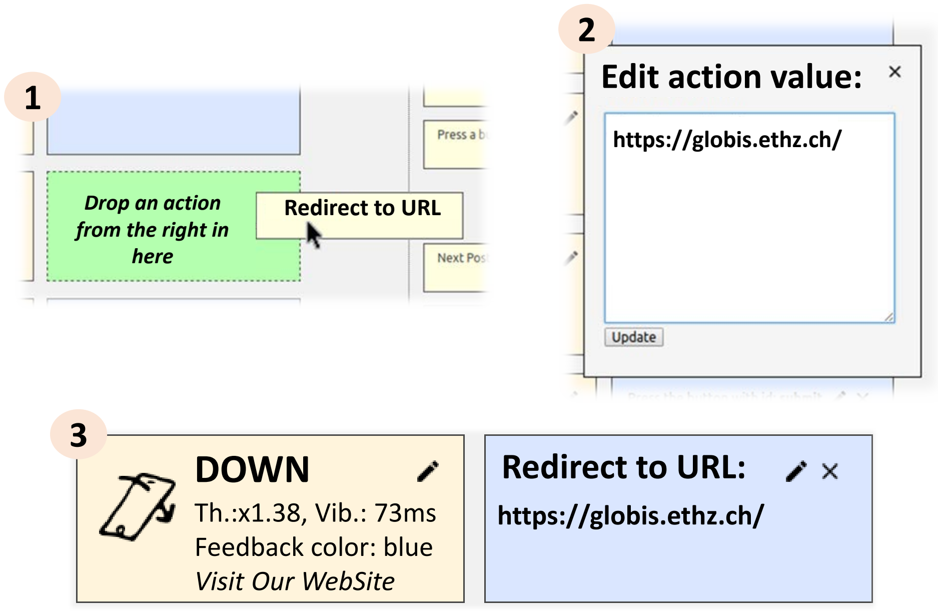
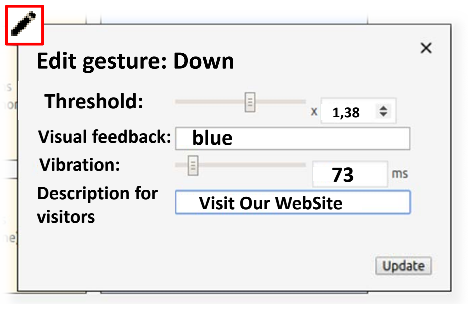


Figure 3.23: Screnshots of the drag and drop interaction.

At this point, users can start adding tilting gestures by accessing the **Block** UI from the main page. The Block UI allows users to associate global actions with tilting gestures by drag and drop interactions. To evaluate the tool, we also developed a different visual interface. We named this alternative UI, **Classic**. The Block and Classic visual interfaces offer the same features, however, while the Block interface is more visual, the Classic approach offers a more traditional solution.

As we can notice from Figure 3.22, in the Classic UI approach, gestures and actions are represented in a textual format and can be customised by clicking on the edit links. In contrast, in the Block UI solution, global actions are represented as rectangles that can be dragged and dropped into the desired jerk tilting gesture. For instance, the user can redirect the website to another URL when the device is tilted down, by dragging the *Redirect to URL* global action to the tilt down box. In Figure 3.23, we can see a visual representation of this example. Once the user drops the global action into the gesture, a pop-up window will show up asking the user to type the desired URL. At this point, the rectangle near to the jerk interaction will show the dropped global action, in this case, the *Redirect to URL* option with the URL inserted by the owner of the website. Users can change the URL by clicking on the pen icon on the action box. Similarly, users can customise the gesture by clicking on the icon in the interaction box (see Figure 3.24 (a)).

Users can combine tilting interactions with touch gestures via the main page of WP-TAT. From this page, users can also activate or de-activate an *info box*. The info box is a DIV shown in the bottom of the page that displays a summary of the available interactions (see Figure 3.24 (b)). The owner of the website can define the description of the gesture by editing the *Description for visitors* option in the setting of the gesture. If the user leaves this variable empty, the description of the gesture will be set as the name of the global action associated to the gesture. The box will be visible once the user taps on a button on the bottom corner of the webpage. The user can also customise the colour of the button or its entire CSS class via the global settings of WP-TAT.



Edit gesture: Down

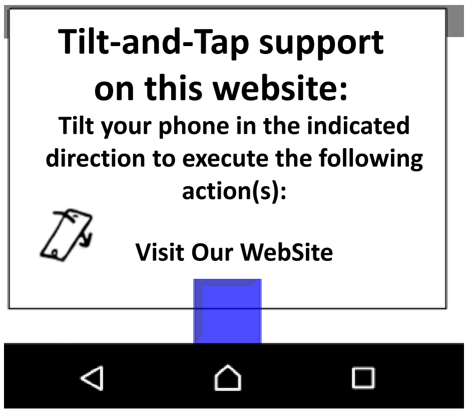
Threshold: x 1,38

Visual feedback: blue

Vibration: 73 ms

Description for visitors: Visit Our WebSite

(a) Gesture customisable option



Tilt-and-Tap support on this website:
Tilt your phone in the indicated direction to execute the following action(s):

Visit Our WebSite

(b) Info box

Figure 3.24: Screenshots of the customisable options and info box.

3.3.2 Architecture and Implementation

In Figure 3.25, we show the architecture of our tool. The main script (`tat.php`) connects WP-TAT to the hook system offered by WordPress. `tat.php` uses some helper scripts that offer functionalities that are shared between the Block and Classic UI. The two approaches are implemented in the `administrative interfaces` set of scripts. The client allows tilting interactions via JavaScript using generated instances of TAT. The association of gestures and global actions are stored in the `gesture` custom post type[6]. In WordPress, content is generally stored in posts which can be customised via custom post types. While posts have a specific set of fields (title and content) with custom post types developers can add more attributes. In WP-TAT, the `gesture` custom post type contains the direction of the gesture (i.e. up, down, left, right, south-east and west, north-east and west, clockwise and counterclockwise), the name of the associated global action as well as all the properties of the gestures such as thresholds and feedback.

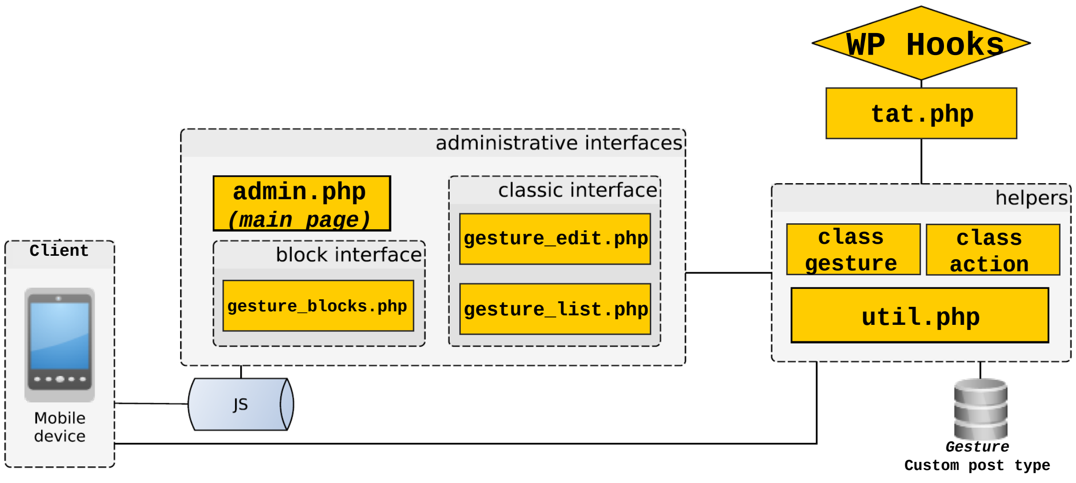


Figure 3.25: Graphical representation of the architecture of WP-TAT.

[6]Custom post type: https://codex.wordpress.org/Post_Types. Accessed on May 2018.

WP-TAT implements a publish-subscribe pattern to associate tilting interactions with global actions (see Figure 3.26). Once the web page is loaded, we first check if the client is a touch-enabled device and if it supports the `DeviceMotionChange` event. If the device passes this first step, we are then free to extend the set of possible interactions on the application by adding an instance of Tilt-and-Tap in the header and then attaching to the body of the page. The proprieties of the instance are overridden by the attributes defined by the users that are stored in the `gesture` custom post type. In the header, we then subscribe to the global actions. Global actions are developed as JavaScript snippets and added to the desired gestures. At this point, global actions are given as anonymous callback functions to the publisher-subscriber object.

When the device triggers one of the gestures, TAT will execute the corresponding callback function. At this step, the callback function will publish an event with the name of the corresponding jerk interactions (i.e. tilt down). The functions of all global actions that subscribed to that event will be triggered and executed.

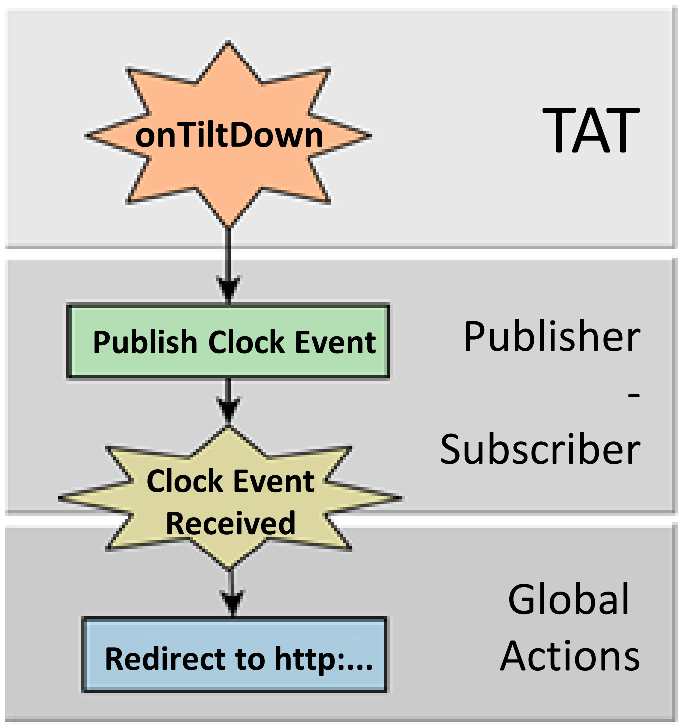


Figure 3.26: Visual representation of the event chain for a tilt down gesture.

3.3.3 Example Applications

To evaluate the flexibility of WP-TAT, we developed three diverse web applications that emulate common types of websites that can be found online. The first application is AmaTilt, an e-commerce website that reproduces the features of Amazon while offering motion gestures to browse articles. With AmaTilt, users can search for additional reviews of the product they are visiting by selecting its name, hold tap on the blue button in the bottom of the page and then tilting the device to the right (see Figure 3.27). By performing this interaction, users will be redirected to the Google results of

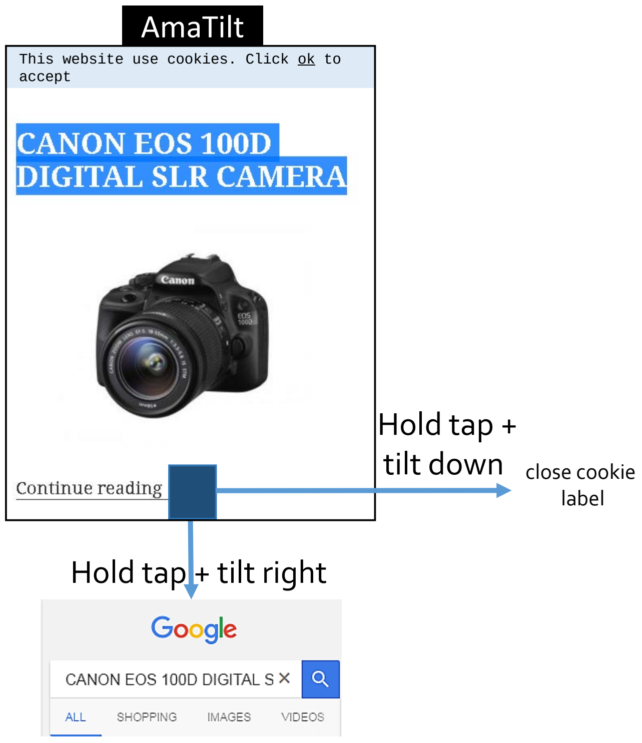


Figure 3.27: Screenshots of AmaTilt and its interaction flow to search selected text on Google.

the selected text. Without tilting gestures, this operation would require the user to select the text and search for the "web search" action in the pop-up menu. We discuss on the advantages of using tilting interactions in this and similar contexts in more detail in Section 3.4.

In AmaTilt, tilting gestures are also used to close and, therefore accept, the privacy cookies information. Users can perform this action with a hold tap gesture performed on the blue button and then a tilt down interaction. To apply this behaviour, the owner of the website should be aware of the ID of the close button of the DIV containing the privacy alert and then associate this global action to the tilt down gesture. We support users in this operation by linking a small tutorial on how to get the ID of an element once they select the *Press a button* global action.

As a second web application, we developed Tiltify, a website that emulates Spotify[7] the popular music gallery application. With Tiltify, users can browse through songs by tilting the device to the right or to the left. A tilt down gesture will open the Spotify page or the official website of the current artist. As similarly discussed for AmaTilt, tilting gestures could improve the user experience when touch gestures would involve tedious copy-paste interactions. Moreover, tilting gestures could make a music application more fun to use and to explore.

As a third and last application, we developed ResTilt, a typical restaurant website. In ResTilt, users can access the reviews of the restaurant in Trip Advisor[8] by a hold tap gesture on the button and then a tilt down jerk tilting interaction. This action

[7]Spotify website: https://www.spotify.com. Accessed on May 2018.
[8]Trip Advisor website: https://www.tripadvisor.com. Accessed on May 2018.

will redirect the user to the Trip Advisor page of the restaurant without requiring any additional steps. Similarly, users can easily access the Google Maps location of the restaurant by selecting the address displayed in the contact section of the website and then tilting the device to the right.

3.3.4 Preliminary User Study

A preliminary qualitative user study was conducted to gather general feedback from users. We recruited four WordPress users, two males and two females (avg. age 28, std. 15.5) to perform a series of tasks with the Block and Classic visual interfaces of WP-TAT. After the completion of the study, we asked participants to answers some questions in the form of semi-structured oral interviews. All participants of the study had some knowledge on how to use WordPress. However, none of our users had ever developed a plugin for the CMS. Among all participants, only one user had background skills in computer science.

Study Design and Tasks

Participants completed the tasks using a laptop which had already installed WP-TAT. Users were sited near a desk with the laptop, a mouse and a camera that recorded the study (all participants agreed to be recorded). In Figure 3.28, we can see the setup of the evaluation. At the beginning of the study, the researcher briefly explained the tool, its goals and concepts. However, the participants were not instructed on the interfaces provided by WP-TAT. As mentioned in the previous section, after users completed the tasks, a semi-structured oral interview was performed. In this phase, the participant was asked to discuss the tool in terms of its ease of use and learn, and their overall opinion of the goals and features of WP-TAT.

Figure 3.28: Set-up of the study.

We designed two sets of tasks: A and B. Both tasks were composed of a series of sub-tasks that were printed on paper and given to participants at the beginning of each series. All sub-tasks were designed to allow users to experiment with all the main features supported by the tool. For this reason, tasks required participants to associate jerk gestures to global actions, customise parameters of the interaction (i.e. thresholds, visual or vibration feedback, description etc.) and check the final website. The interactions were added to a test website that contained mocked data. The order of the tasks and the interface used (Block or Classic) was shuffled among participants.

Results

All users were able to complete the tasks with both interfaces and learned how to use the tool within minutes. During the semi-structured oral interview, the researcher asked participants which interface they preferred. Three users stated that Block was their favourite UI. These users found this approach more intuitive and fun to use than the Classic approach. They also stated that the drag and drop interaction seemed the appropriate interaction to associate gestures and actions. In this context, one participant also said that the visual representation of gestures and global actions made it easier to have a quick overview of the association between the two.

In contrast with these opinions, one participant argued that the "Block UI felt too much as gamification [...]" and he preferred the soberness of the Classic visual interface. Overall, all participants appreciated that in the Classic approach all options were displayed in the same page while, in the Block UI, they had to click on the "edit" button to see all the parameters of each gesture.

We asked participants to give some general comments on WP-TAT, and therefore, gathered some feedback on our implemented interfaces. Overall, users suggested only some minor improvements to our UI such as the necessity of a colour picker to modify the colour of the visual feedback or the possibility to drag a global action from one gesture to another.

3.3.5 Discussion

With WP-TAT, we allow end-users to extend their web application with tilting interactions. To use WP-TAT, users are not required to have any particular programming skills and gestures can be applied by simply using a visual interface. We designed a number of possible global actions and created three different web applications to show the potential benefits of motion gestures as well as to demonstrate the flexibility of our tool. A preliminary user study was carried out to gather qualitative feedback on the tool and the two proposed interfaces, Block and Classic UI. All participants were able to learn the tool within minutes, and three out of four users preferred the intuitiveness of drag and drop interactions to associate gestures and global actions.

One issue of WP-TAT relies on the possible inconstency of tilting interactions among different websites. In our approach, every user can decide which motion gesture is best suited to their use cases. This freedom could potentially cause different web applications to use the same gesture for two different purposes. For instance, some users could decide that a tilt down interaction is a good way to close and accept the information cookie label displayed in their website. In contrast, another user could believe that a tilt up is a better suited gesture for the same action. Such lack of consistency can confuse users browsing various websites. Although we recognise this issue, we believe that by providing the user with tools to easily enlarge the set of possible interactions on the web, the interest in motion gestures might spread and lead browsers to support these interactions more consistently in the future.

Finally, the experiences gathered with TAT, TAT 2.0 and WP-TAT gave us the chance to better understand the potential of motion gestures. We discuss our findings in Section 3.4.

3.4 Design Observations

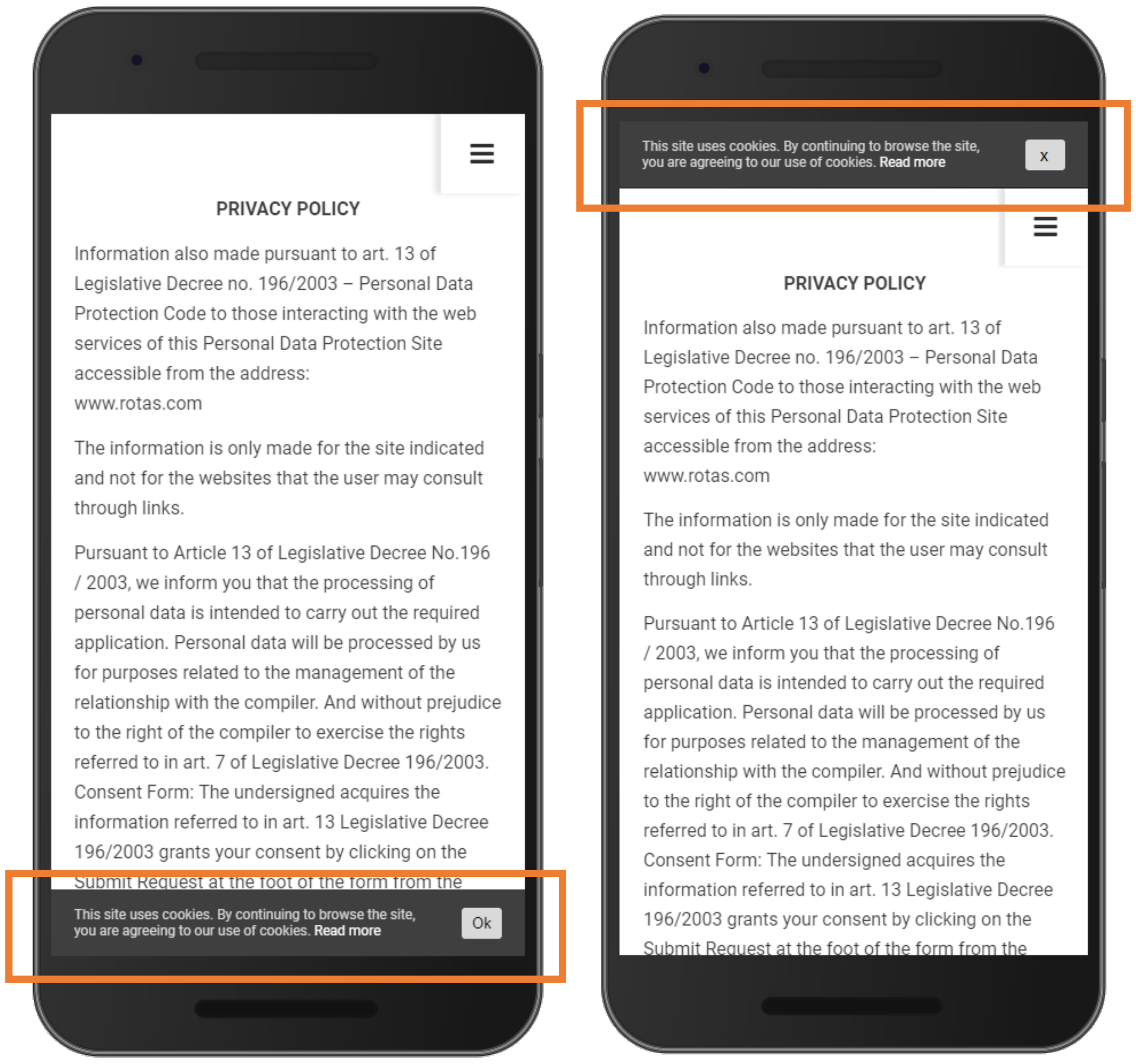
Thanks to our tools and frameworks, we were able to quickly experiment with the use of tilting interactions on many diverse web applications. This knowledge, together with the feedback received from participants during developer and user studies, gave us the chance to better understand where tilting gestures could improve the overall web browsing experience. We empirically drew a series of design observations that could help future developers and researchers when exploiting motion gestures on mobile devices.

We believe that tilting interactions could make web interactions more rich. However, they should not replace touch gestures entirely. Motion interactions can be a good addition to the traditional set of gestures particularly when solo-touch interactions could potentially fail. For example, nowadays web applications are required to inform users that cookies are used by the website[9]. Commonly, this information is shown in the form of an alert box on the top or bottom of the web page (see Figure 3.29). At this point, users can hide this information box by clicking on an “ok” or “close” button. This information box needs to be visible, however, it should not hide too much content.Buttons could be hard to reach and tap if too small and placed too high (or too low) in a page. These issues can be particularly problematic when using big phones or tablets. In this contexts, users cannot always reach all sections of a page with the hand holding the device. In such cases, tilting interactions could be a valid alternative to touch gestures. A rapid tilt of the device in some direction could offer a better user experience than trying to reach and tap a small button. Further, motion gestures have the advantage that they do not require the user to change their hand position when holding the device. Overall, tilting interactions can be suited when, for specific design choices, UI elements are too small or placed in positions that are hard to reach with the thumb.

Moreover, motion gestures could be exploited to perform actions that would require several steps with touch interactions. As developed in AmaTilt, Tiltify and ResTilt (see Section 3.3.3), tilting gestures can be used as short-cuts to perform more operations at once. For instance, motion interactions can be used to search the currently selected text on Google, or find a physical location on Maps. Nowadays, browsers show a pop-up menu when selecting text. Users can browse through this menu to copy, share or do a Google search of the currently selected text (see Figure 3.30). However, the number of available options is limited since the menu should not hide the rest of the content displayed on the page. Adding new options could overcrowd the page with too many entries and negatively influence the user experience. In contrast, tilting interactions could offer more options without the need to represent them visually in the page.

Although tilting gestures can offer many advantages in this and other contexts, they are not free from issues. Motion interactions could be triggered involuntarily by the user and, for this reason, they should not be employed for sensitive operations. Deleting emails or buying products online are not good scenarios for these type of gestures. On the other hand, closing the cookie info box alert or browsing a gallery of pictures could be more suitable use cases. Moreover, in the Google search and Google maps scenarios, the number of false positives could be reduced since motion gestures would be triggered only if text is selected first.

[9]Protection of personal data: https://goo.gl/r8XJ9C. Accessed on May 2018.


PRIVACY POLICY

Information also made pursuant to art. 13 of Legislative Decree no. 196/2003 – Personal Data Protection Code to those interacting with the web services of this Personal Data Protection Site accessible from the address: www.rotas.com

The information is only made for the site indicated and not for the websites that the user may consult through links.

Pursuant to Article 13 of Legislative Decree No.196 / 2003, we inform you that the processing of personal data is intended to carry out the required application. Personal data will be processed by us for purposes related to the management of the relationship with the compiler. And without prejudice to the right of the compiler to exercise the rights referred to in art. 7 of Legislative Decree 196/2003. Consent Form: The undersigned acquires the information referred to in art. 13 Legislative Decree 196/2003 grants your consent by clicking on the Submit Request at the foot of the form from the

This site uses cookies. By continuing to browse the site, you are agreeing to our use of cookies. **Read more**

This site uses cookies. By continuing to browse the site, you are agreeing to our use of cookies. **Read more**

PRIVACY POLICY

Information also made pursuant to art. 13 of Legislative Decree no. 196/2003 – Personal Data Protection Code to those interacting with the web services of this Personal Data Protection Site accessible from the address: www.rotas.com

The information is only made for the site indicated and not for the websites that the user may consult through links.

Pursuant to Article 13 of Legislative Decree No.196 / 2003, we inform you that the processing of personal data is intended to carry out the required application. Personal data will be processed by us for purposes related to the management of the relationship with the compiler. And without prejudice to the right of the compiler to exercise the rights referred to in art. 7 of Legislative Decree 196/2003. Consent Form: The undersigned acquires the information referred to in art. 13 Legislative Decree 196/2003 grants your consent by clicking on the Submit Request at the foot of the form from the

(a) Label at the bottom with "Ok" button (b) Label at the top with "X" button

Figure 3.29: Two examples of cookie informative labels (highlighted in orange).

Combining tilting gestures with touch interactions could be used to improve the accuracy of motion interactions [92]. To provide such combinations, a portion of the screen needs to be dedicated to detect tap and hold tap gestures. While this could hide some of the content in the website, we can diminish this factor by making the DIV almost transparent when not touched. On the other hand, this button could capture the attention of the user, and it could provide an effective way of instructing visitors on tilting interactions. In fact, another challenge of motion gestures is the necessity of informing the user that these gestures are available. Since motion interactions are not yet widespread in browsers, users would not be aware of this possibility. The button could attract users to interact with it and, once clicked, an informative label can summarise the available gestures and inform the user about the interactions. However, we believe that if browsers support motion gestures in the future, the intuitiveness of these interactions could allow users to learn them easily.

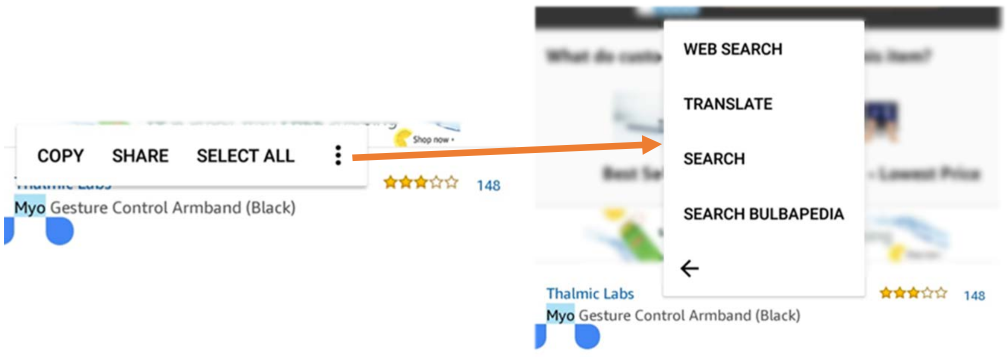


Figure 3.30: Screenshots of the pop-up menu when text is selected in Chrome.

We continue this discussion on how to best apply tilting gestures on the web in Section 4, where we also present design observations obtained when exploiting motion interactions in cross-device applications.

3.5 Discussion

In this chapter, we presented tools to encourage end-users and developers to enlarge the set of possible motion-based gestures in web applications by exploiting tilting gestures. Although motion interactions have a number of potential benefits [15, 163, 156], they have rarely been applied in web applications.

Overall, tilting gestures can be particularly challenging to implement. Given their nature, motion interactions rely on sensor data that can differ between devices, they can be developed in many alternatives, and various parameters can influence the final user experience of the interaction. Moreover, different browsers deal with sensor data in different ways causing incompatibilities issues between platforms and devices.

With our tools, we wanted to improve the development process of these gestures and, therefore, allow developers and end-users to experiment with these interactions on the web. After an analysis of related work on tilting gestures, we categorised these interactions into two major classes: jerk and continuous motion gestures and developed Tilt-and-Tap. Each class of interaction can be combined with touch gestures and customised by a number of parameters. These set of interactions was later extended with Tilt-and-Tap 2.0 that offers all main variants of continuous tilting gestures discussed in literature [205, 23, 179]. Furthermore, we used TAT to encourage end-users in exploiting tilting interactions in their web application by offering WP-TAT, an easy to use visual interface built on WordPress.

To show the capabilities of our frameworks and tools, we developed a number of diverse real-case applications and carried out developers and user studies. Furthermore, we draw a series of design observations on how to use motion gestures on the web. Finally, the Tilt-and-Tap project gave us the chance to easily experiment with tilting interactions also in cross-device scenarios and evaluate different motion gestures when more than one device is involved in the communication. We discuss these findings in the next chapter.

4 Cross-Tilt-and-Tap

In this Chapter[1], we move our research forward and explore the use of tilting interactions in cross-device web applications. Motion gestures could offer a good alternative to touch interactions when more than one device is involved. For example, waving a smartphone or a tablet in the direction of another device to share data could provide an intuitive form of interaction between heterogeneous devices [23]. However, building such applications raises many challenges. A preliminary user study was carried out to understand which issues developers have to face when developing cross-device applications that use tilting gestures. The experience gathered during the study, and the feedback received from the users gave us the chance to list a series of requirements to improve the development process of such applications as well as propose a better user experience. Informed by these findings, we built Cross-Tilt-and-Tap (CTAT)[2], a framework for the rapid development of cross-device applications that exploit motion gestures.

In this Chapter, we start by presenting the requirement analysis phase and its results in Section 4.1. In Section 4.2, we present the final framework in terms of its concepts and example of its use. With CTAT we were also able to evaluate continuous tilting gestures developed with TAT 2.0 in cross-device scenarios. We report the design and the findings of the user study in Section 4.3. Finally, we give concluding remarks in Section 4.4.

4.1 Requirements Analysis

As previously studied in research [23, 16, 45], the use of tilting interactions in cross-device applications can offer many advantages especially when a mobile device is used to control a public or semi-public screen. Using motion gestures does not require the user to touch a potentially dirty display and they offer an intuitive way of remotely

[1]Earlier versions of parts of this Chapter were originally published as Di Geronimo et al. [56, 53]
[2]Video showing CTAT main features: https://goo.gl/RRmNPv. Accessed on May 2018.

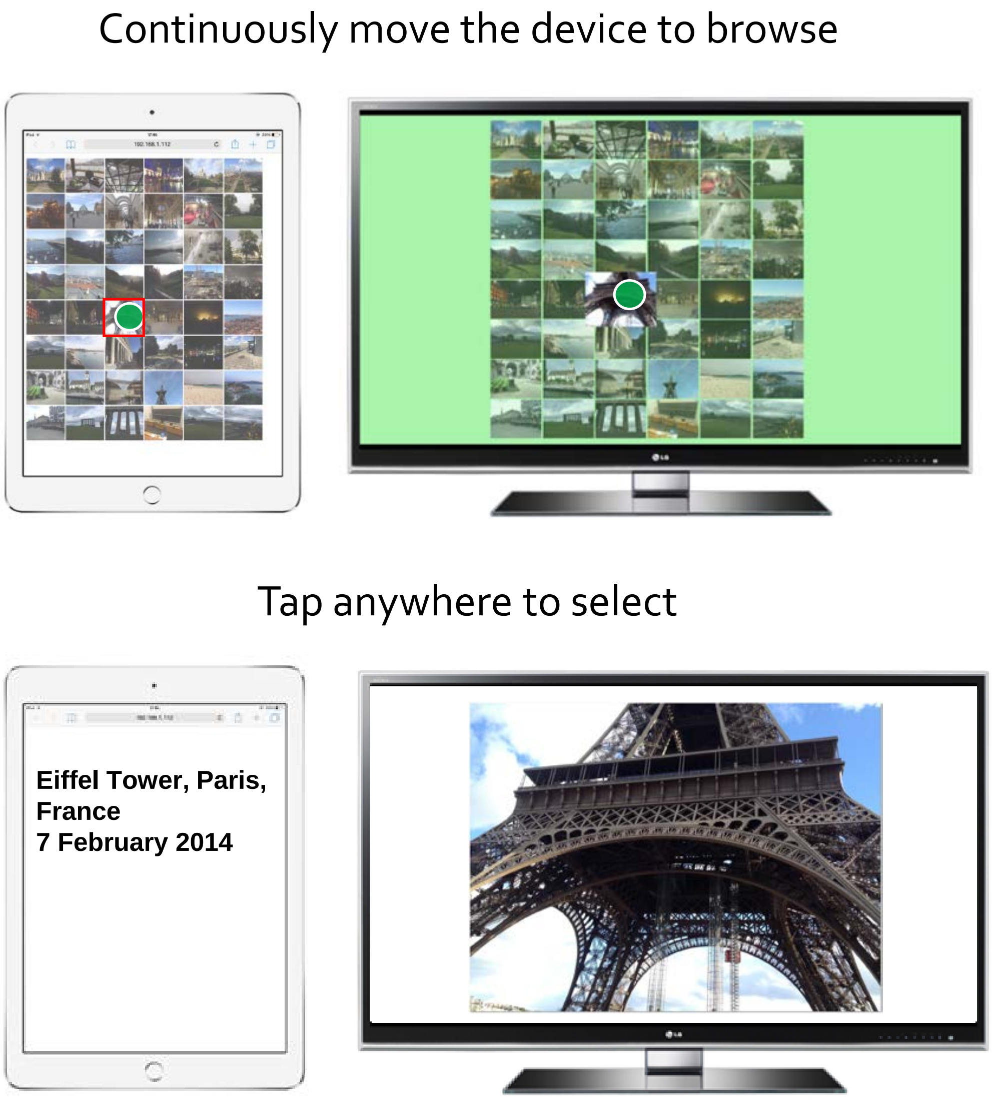

Continuously move the device to browse
Tap anywhere to select
Eiffel Tower, Paris, France
7 February 2014

Figure 4.1: Screenshots of Tilt-Gallery and its interactions flow.

moving a cursor on a big monitor [23]. However, tilting interactions can be particularly challenging to implement in cross-device scenarios since gestures have to be recognised, sent and received on devices of different sizes and platforms.

To better analyse these issues, we developed Tilt-Gallery, a cross-device web application that allows users to browse pictures on a public screen by performing motion gestures on their a mobile device.

Figure 4.1 provides an overview of Tilt-Gallery. In the example, an iPad is used to remotely control a cursor displayed on a large display. The cursor is moved according to the orientation and speed of the device. Images are displayed in a grid and, every time the ball is over one of the pictures, a red border will be displayed on the small device while the image is slightly enlarged on the large screen. To select a picture that is currently below the cursor, users can tap anywhere in the page on the mobile device. Once selected, the photo will be shown in full size on the large screen while additional information about the image will be displayed on the mobile device. In the example shown in Figure 4.1, an image of the Eiffel Tower is selected, therefore, the date and title of the picture are shown on the tablet, while on the large monitor it is displayed in full size.

Mobile devices can be paired to the large screen by typing the URL of Tilt-Gallery in a browser. If a mobile device and a screen are already connected to the application, any other device that connects will not be able to control the large display. On these clients, we show a mirrored image of what is visible on the large display. However, if the first mobile device disconnects by closing the application, future connections from other clients will be allowed.

Tilt-Gallery was developed using the first version of TAT (see Section 3.1) for the recognition of continuous motion interactions. Node.js and Socket.IO allow the communication between devices. We developed Tilt-Gallery with three goals in mind. First, we wanted to study what challenges arise when building applications that run across devices and use tilting interactions. Second, via a preliminary user study, we wanted to gather feedback from the user on the scenario envisioned. Finally, the experience gathered during the development of Tilt-Gallery and the results obtained from the study would allow us to list a series of requirements for a framework with the goal of improving the user experience and the development process of such applications. In the next sections, we present the study conducted on the application, and report on the requirements drawn.

4.1.1 Study Design and Tasks

To gather feedback from users and evaluate the advantages of using tilting interactions in cross-device web applications, we carried out a preliminary user study with 12 users (9 males and 3 females).

During the evaluation, we asked participants to use a mobile device to select pictures on a larger screen using the Tilt-Gallery application. In the study, users were required to find and select a series of pictures in the gallery. The task had to be performed in four different setups: smartphone tilt, smartphone touch, tablet tilt and tablet touch. In the tilt cases, users could browse the gallery only via continuous motion gestures, while in the touch version, tilting interactions were disabled and users could only select images via touch gestures. The size of the images and indicator were scaled accordingly to the dimension of the mobile device. In the study, we used an iPhone 6 in the smartphone cases and an iPad Air for the tablet cases. At the end of the four tasks, participants were asked to rate their experience using the application by filling in a questionnaire.

The structure of the study was organised as follows:

1. The researcher introduced the participant to the study and explain how the application works.
2. The participant is required to try the application on a test page using tilt and touch interactions on the iPhone.
3. The researcher explained the tasks.
4. The participant performs the tasks with smartphone touch.
5. The participant performs the tasks with smartphone tilt.

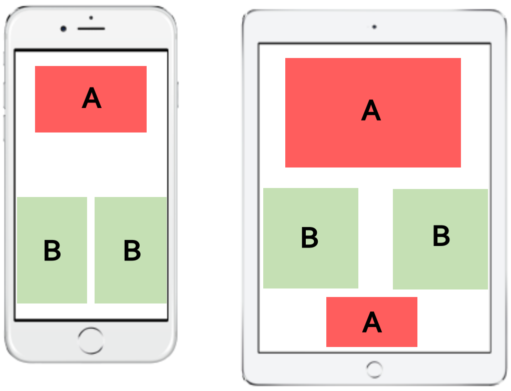


Figure 4.2: Classification of areas for the phone and tablet.

6. The participant is required to try the application on a test page using tilt and touch interactions on the iPad.
7. The participant performs the tasks with tablet touch.
8. The participant performs the tasks with tablet tilt.
9. The participant fills in the questionnaire.

We shuffled the order of tilt and touch tasks, as well as the smartphone and tablet cases for each participant. The pictures that participants had to select were classified into two categories, depending on where they were located in the page and, therefore, on how hard are they are to tap assuming that the device is held in portrait mode. As shown in Figure 4.2, we defined areas that are hard as class A, in contrast, the areas labelled with the B class are more comfortable to reach [219]. For the mobile tilt and touch, users were required to search and select nine pictures (three for each zone). In the case of the tablet, participants had to select eight images (two for each zone). The set of pictures were different for each task.

Figure 4.3 shows the setup of the study. A 24 inch TV was placed on a desk, and was used as the larger display of the evaluation. Near to the screen, a laptop was placed to

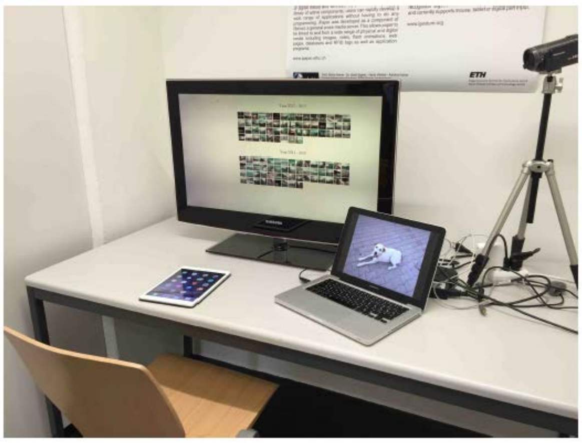


Figure 4.3: Set-up of the study.

show the picture users had to search for and select in the gallery. Every time the correct image was selected, the researcher show the next picture to be found on the laptop. Users were seated circa 1.50 meter away from the screen during the study. In a real case scenario, users would most probably be standing since semi and public-displays are usually hung on a wall. However, since participants had to use the application for an extended period, we decided to let them be seated during the evaluation to lower possible discomfort. Participants were recorded during the study and times were logged.

4.1.2 Results and Discussion

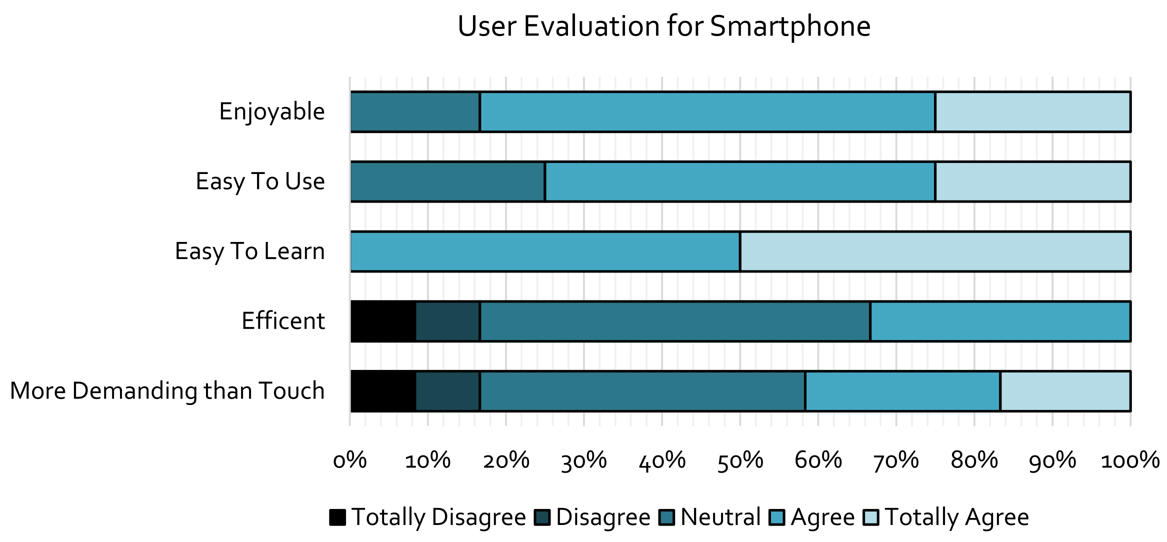
At the end of the study, we asked participants to rate their experience of using tilting interactions on tablets and phones. Figure 4.4, shows the users' feedback on five different factors: enjoyability, easiness to use and learn, efficiency and, finally, on how demanding motion interactions were when compared to solo-touch gestures.

Overall, participants found tilting interactions enjoyable to use. However, we note a difference between tablet and smartphone tilt tasks. 80% of our users agreed or strongly agreed that tilting interactions were enjoyable to use on the smartphones, instead, on tablets, 70% of our participants were of the same opinion. We can find this difference also in the easiness to use rating. 70% of our users found tilting interactions easy or very easy to use on phones, in contrast, only 50% of them found them easy to use on tablets. This trend is also found on the efficiency of the gesture. 40% did not find tilting gestures efficient on tablets, while on smartphones, only 20% of our participants had the same opinion. Similarly, 50% of users felt that tilting gestures were less or as demanding as touch when using the iPhone, while in tablet tasks, 59% felt the same.

We speculate that the differences between tablet and smartphone ratings were influenced by the velocity of the cursor selecting pictures. In both tasks, the speed of the ball was kept the same. The indicator moved only according to the orientation, and the rotation speed of the device and no additional acceleration were applied. For this reason, to move the ball to the desired location, users had to tilt the device more in tablet tasks than in smartphone tasks. We asked participants to comment on this factor, and some users felt that the ball was slower during the tablet tasks than in smartphone tasks. However, other participants discussed that they preferred using motion gestures on tablets since its form factor gave them more control over the indicator.

In terms of timing, participants needed around 7 seconds to find and select one image (59.45 avg. total time per task). This time was similar and not statistically different among all four version. Moreover, we did not find any time differences between selecting pictures located in different areas. Concerning error rates, in mobile tap tasks, a total of three errors were performed by two participants due to selecting the wrong image. In mobile tilt, we counted six errors from four users. In all these cases, participants correctly located the picture but selected the wrong one due to overshoots. Similarly, in tablet tilt, four different participants overshot the target a total of five times, while in tablet touch, no errors were performed.

It is important to note that the main goal of this study was to gather feedback on our approach and not to compare tilting interactions with touch gestures. As discussed in Section 3.4, we believe that motion interactions should not entirely replace touch gestures but instead be used in combination to make web applications more intuitive and fun to use. Moreover, in cross-device scenarios, such as the one studied in this



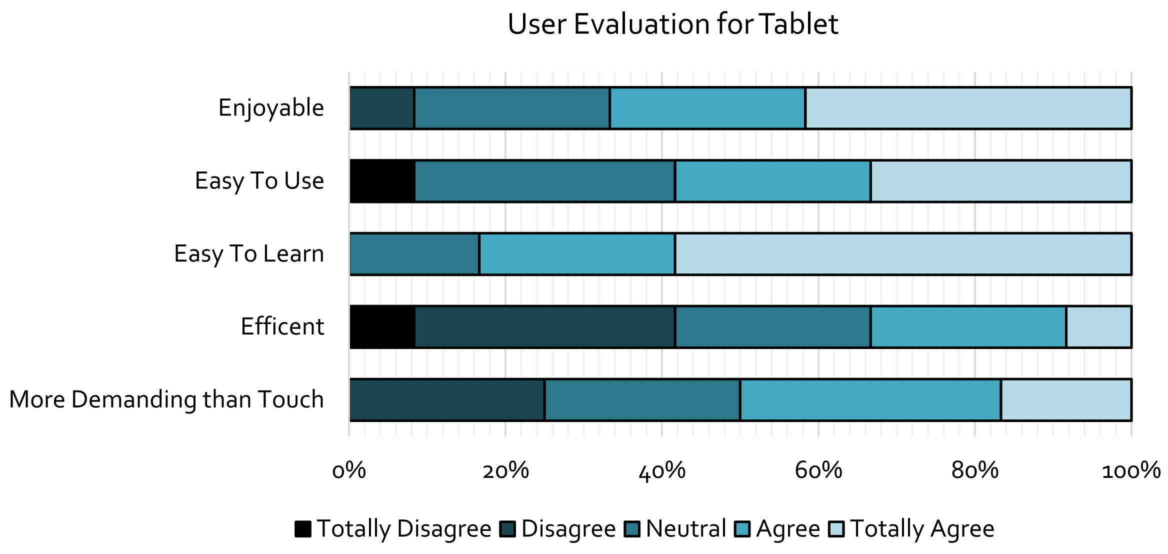


Figure 4.4: Participants ratings of tilting interactions on smartphone and tablet tasks.

evaluation where a mobile device is used to control a bigger screen, motion gestures could offer features that are not achievable with solo-touch interactions. For instance, with tilting gestures, users do not need to focus on their mobile device to interact with the bigger display. As found in our study, the attention of the user was almost exclusively on the large screen and not on the iPad or iPhone during tilt tasks. In contrast, by using touch gestures to select pictures, participants needed to look at the mobile device more often. One participant stated *" [...] during the tapping I could not use the TV at all since I would have to search for the image twice."*.

Only one user of our study started by looking at the bigger display in the tablet touch case, however, after few seconds, he realised that it was not a comfortable way to select pictures and said *"Why look at the TV, I need to select it from the iPad anyways."*

These results motivated us to continue our work on the use of tilting interactions in cross-device scenarios. The experience we gathered during the creation of Tilt-Gallery together with the feedback we received from users, gave us the chance to understand what could improve the development process of cross-device applications. We discuss these possible improvements in the next section.

4.1.3 From TAT to CTAT

The experience gathered during the development of Tilt-Gallery as well as the feedback received from the participants of our user studies, gave us the opportunity to draw a list of requirements of a framework with the goal of improving the development process and the user experience of such applications.

In terms of development, building applications like Tilt-Gallery requires many steps as well as the necessity to write repetitive code that could easily lead to errors. To better understand this issue, we give an overview of how Tilt-Gallery was developed.

As discussed in Section 4.1.1, in our application, we recognise movement of the device by using our TAT framework and then send messages to the screen via Socket.IO. Figure 4.5 shows an example instance of Tilt-Gallery when a mobile device (M1) and a large screen (S1) are connected to the Tilt-Gallery URL. SE represents the Node.js server that allows the communication between devices, while M2 is a second mobile device that connects to the URL only after M1. In this case, the application works as follows:

1. S1 and M1 connect to the Tilt-Gallery URL. We identify which connected client is a mobile device and which is a screen by looking at which device supports motion gestures via TAT.

2. At this point, every time M1 returns a movement change, we send the new position of the cursor to SE, which receives the data and sends it back to M1.

3. When an element is hovered by the cursor from S1, we send a message to SE that dispatches it to M1 and executes the callback function.

4. When an element is selected from S1, we send another message to SE that dispatches it to M1 and executes the callback function.

5. M2 connects. Tilt-Gallery recognise M2 as a mobile device, however, since M1 is still connected to the application, M2 cannot control the cursor of Tilt-Gallery.

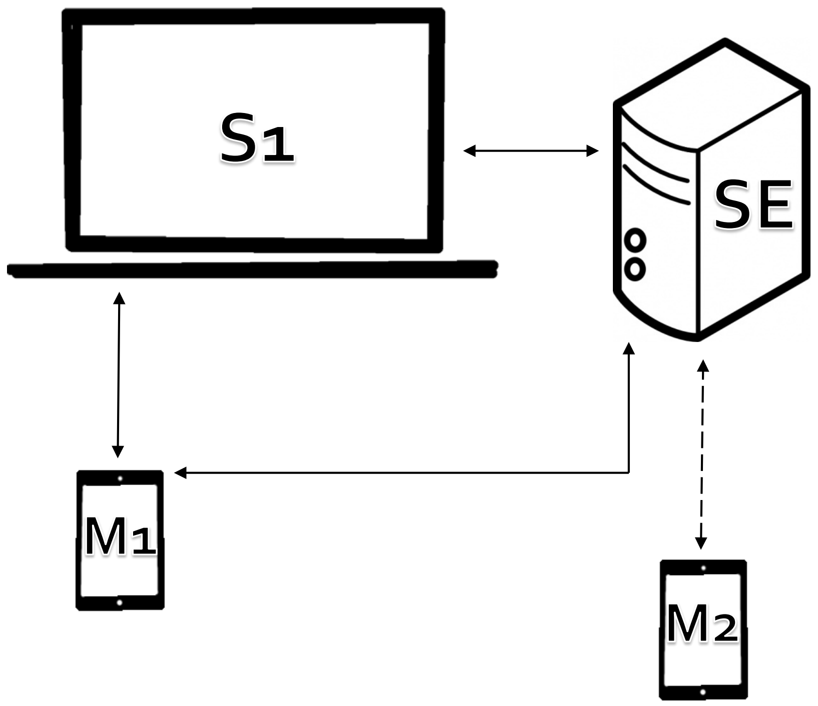


Figure 4.5: Example execution scenario of Tilt-Gallery.

M2 is presented with a mirrored image of what is displayed on S1. If M1 disconnects, Tilt-Gallery checks if there are other mobile devices connected, find M2 and gives it the control of the ball.

On the client, there is no way to easily distinguish the sender and the receiver. The developer is responsible for recognising the devices and executing the right functions depending on their roles. More issues arise if other features are added to the application. For instance, developers might want to show the indicator only on the receiver and not on the sender. Moreover, they might want to send the movements of the ball only to one specific receiver and not to all devices that are connected to the same URL. In these cases, sending messages back and forth from the client to the server and back can easily confuse the developer. Any additional feature makes the communication more prone to errors and challenging to develop.

Below, we list four main of features that a framework should offer to improve the development process of such applications:

1. Automatic recognition of senders and receivers among the connected devices.

2. Automatic execution of callback functions on clients once events are triggered (e.g. image selected).

3. Hide implementation details on new connections and disconnections of clients.

4. Hide implementation details on the recognition of tilting gestures.

In terms of user experience, during the preliminary study, many participants felt that tilting interactions were more efficient on phones rather than on tablets and some stated that the indicator was too slow on the larger mobile device. TAT was built with the goal of supporting interactions that would be consistent among different devices form factors, however, this feature might not be beneficial for tablets in cross-device scenarios. In this context, since users are focusing on a large screen while moving the device they could easily feel that the indicator is moving too slowly despite the fact that they are rotating the tablet quickly.

Taking these points into consideration, we developed CTAT, a framework that automatically recognises the correct senders and receivers of the messages, manages motion events fired by mobile devices, and avoid the ping-pong exchange of messages between the client and server. Moreover, as inferred by users' feedback, CTAT adapts the speed of the ball accordingly to the size of the screen of the mobile device. In the next sections, we discuss all the features offered by CTAT and its implementation and architecture details.

4.2 The Framework

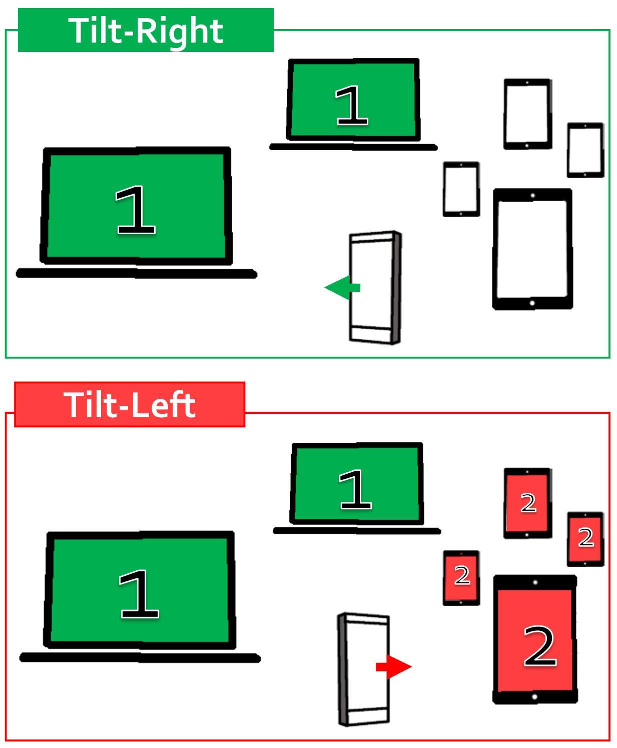
Taking into consideration the requirements listed after the development and study of Tilt-Gallery challenges, we developed CTAT, a framework for the rapid development of tilting interactions in cross-device scenarios. In the next sections, We give an overview of the framework, its architecture and implementation details and present a demonstrator application to show its capabilities.

4.2.1 Concepts and Example Usage

As supported by TAT, in CTAT, jerk and continuous tilting gestures can be customised as desired and used in combination with touch interactions. For performance reasons, we decided to split CTAT into two variants CTATJ (CTAT-Jerk) and CTATC (CTAT-Continuous). Diminishing the size and the complexity of the framework improves the performance of the system. However, if developers desire to use both motion gestures, they are free to use CTATJ and CTATC together.

To give a general overview of CTATJ and CTATC, we present three examples usage of the frameworks to show its features. In Figure 4.6, jerk tilting gestures are used to interact with other devices. In the example, a tilt of the smartphone to the right will modify the background colour of the laptops, while a tilt to the left will change the background colour of the tablets. To reproduced this behaviour, after including the framework in their application, developers are required to indicate some parameters for each tilting interaction. In our case, two variables are required, one for tilt left (`stiltleft`) and one for tilt right (`stiltright`). In the example. these two variables indicate: the sender of the interaction (`‘‘smartphone’’`), the receivers (`‘‘laptops’’` and `‘‘tablets’’`); the desired tilting gesture (`‘‘tiltleft’’` and `‘‘tiltright’’`) and, finally, the callback function to be called once the interaction has been performed (`‘‘changecolorgreen’’` and `‘‘changecolorred’’`). Once these two variables are defined, they can be passed to the CTATJ instance.

As seen in this example, to target only one device, developers can use the singular form of a specific category of device (e.g. smartphone), while to target more devices of the same type they can use the plural form (e.g. laptops, tablets, etc.). CTATJ



index.js

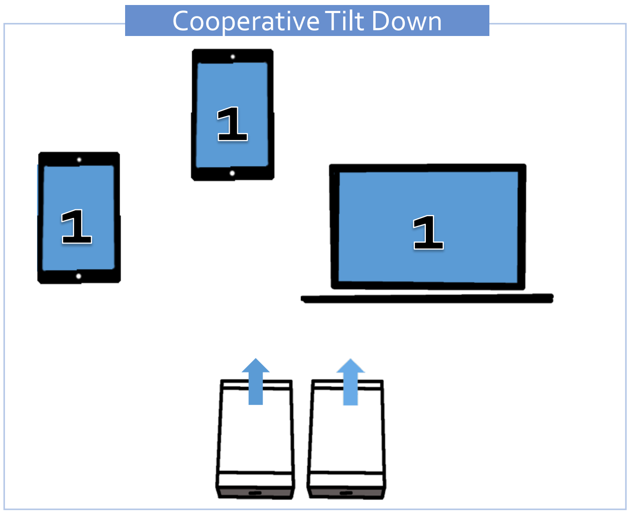
```
var stiltleft= {
sender: “smartphone”,
receiver: “laptops”,
tilting: “tiltleft”,
callback: changecolorgreen
}

var stiltright = {
sender: “smartphone”,
receiver: “tablets”,
tilting: “tiltright”,
callback: changecolorred
}

var settings = new Array();
settings[0] = stiltleft;
settings[1] = stilrright;
var ctatj = new
ctatj(settings);
```

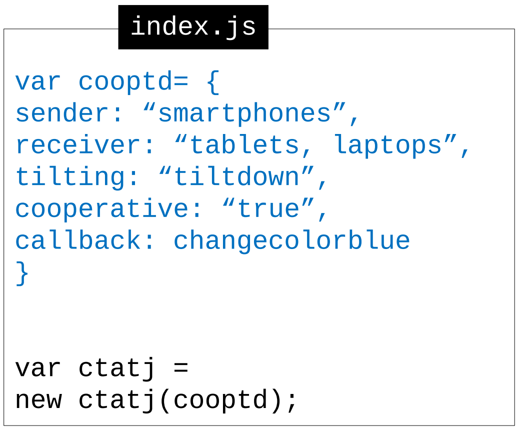
Figure 4.6: Example usage of CTATJ.

supports many-to-many interactions, meaning, that more devices can trigger jerk tilting gestures to one or more receivers. When developers indicate the singular form of a category, CTAT will target the first device of that category that connected to the URL. If one device involved in the communication disconnects, the next appropriate client will be assigned to the interaction. The system detects the category of each client, and determines the corresponding behaviour. In CTATJ and CTATC, we support four different types of devices: laptops (and big screens), tablets, smartphones and smartwatches. We distinguish and categorise clients into these categories based on their screen resolution. We give more details on how this is detected in Section 4.2.3.

With CTATJ, developers can also exploit cooperative gestures. A cooperative gesture is an interaction performed by more than one device at the same time. In Figure 4.7, we can see an example of such gesture where the background colour of laptops and tablets is changed to blue when two smartphones are tilted down at the same time. As similarly done for the previous example, to reproduce this behaviour, developers have to define specific settings for each tilting interaction. In our case, only one variable (`‘‘cooptd’’`) is necessary. This variable indicates that the senders of the interaction are `‘‘smartphones’’`, the receiver `‘‘tablets’’` and `‘‘laptops’’`, the gesture to be performed is a `‘‘tiltdown’’` interaction and `‘‘changecolorblue’’` is the callback function. We can distinguish cooperative interaction to not-cooperative by the `‘‘cooperative’’` setting that, in our case, is set to true.

As seen in this example, developers can target more than one type of device by using their labels (in singular or plural form) separated by a comma. If developers want to target all devices as senders or receiver, they can specify the keyword `‘‘all’’` in the sender and\or receiver setting.





index.js

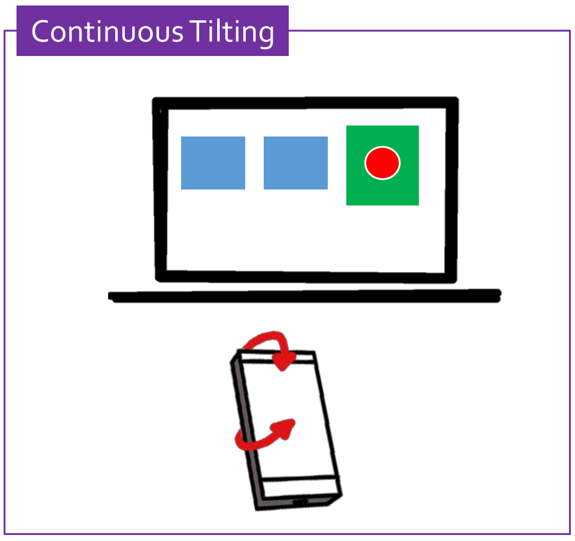
```
var cooptd= {
sender: “smartphones”,
receiver: “tablets, laptops”,
tilting: “tiltdown”,
cooperative: “true”,
callback: changecolorblue
}


var ctatj =
new ctatj(cooptd);
```

Figure 4.7: Example usage of CTATJ for cooperative gestures.

Finally, in Figure 4.8, we show an example use of CTATC with or without the use of TAT 2.0 (index.js and index.js* accordingly). As similarly implemented for Tilt-Gallery, users can browse a gallery of pictures shown on the bigger display by moving their mobile phones. The indicator is shown in the form of a ball, and it is displayed on both devices (laptop and smartphone). Once the indicator is inside an element, the picture is enlarged on the bigger screen. To reproduce this behaviour without the use of TAT 2.0, developers are required first to include CTATC in their application and then define the setting variable to pass to an instance of the framework. As similarly implemented for CTATJ, the setting variable (`settings`) must indicate the sender of the interaction (`‘‘smartphone’’`), the receiver (`‘‘laptop’’`) and the elements that will be selected (`‘‘elem’’`). If not specified otherwise, the ball will be displayed on both screens. Alternatively, if developers want to show the ball on only one of the devices involved in the connection, they can set the variable ball to `‘‘smartphone’’` or `‘‘laptop’’`. In contrast with CTATJ, CTATC offers one-to-many interactions, meaning, that only one device can trigger continuous motion gestures while one or many can be the receivers of the messages.

With these settings, the ball will move using the balance board behaviour discussed in Section 3.2. To exploit other forms of continuous interactions, developers can simply use TAT 2.0 before the creation of the CTATC instance (see Figure 4.8 index.js*). In this case, the callback functions and the element variable can be directly indicated by using TAT 2.0.

Overall, CTATJ and CTATC are able to automatically detect and execute callback functions on the correct senders and receivers indicated by developers. The framework hides implementation details of motion gestures and automatically manages connections and disconnections of clients. Moreover, CTATC adapts the speed of the cursor depending on the size of the devices to improve the user experience. For these reasons, we fulfil the requirements inferred by the preliminary user study discussed in Section 4.1.



index.js

```
var settings = {
sender: “smartphone”,
receiver: “laptop”,
ball: “laptop”,
elements: “elem”
callback: enlarge
}

var ctatc =
new ctatc(settings);
```

index.js* (with TAT 2.0)

```
var tat20 = new TiltAndTap();

…
tat20.twoD (styleball,
styleball,
container,
moveType,
onEnter,
onExit,
touch);


var settings = {
sender: “smartphone”,
receiver: “laptop”,
}

var ctatc =
new ctatc(settings);
```

Figure 4.8: Example usage of CTATC with and without TAT 2.0.

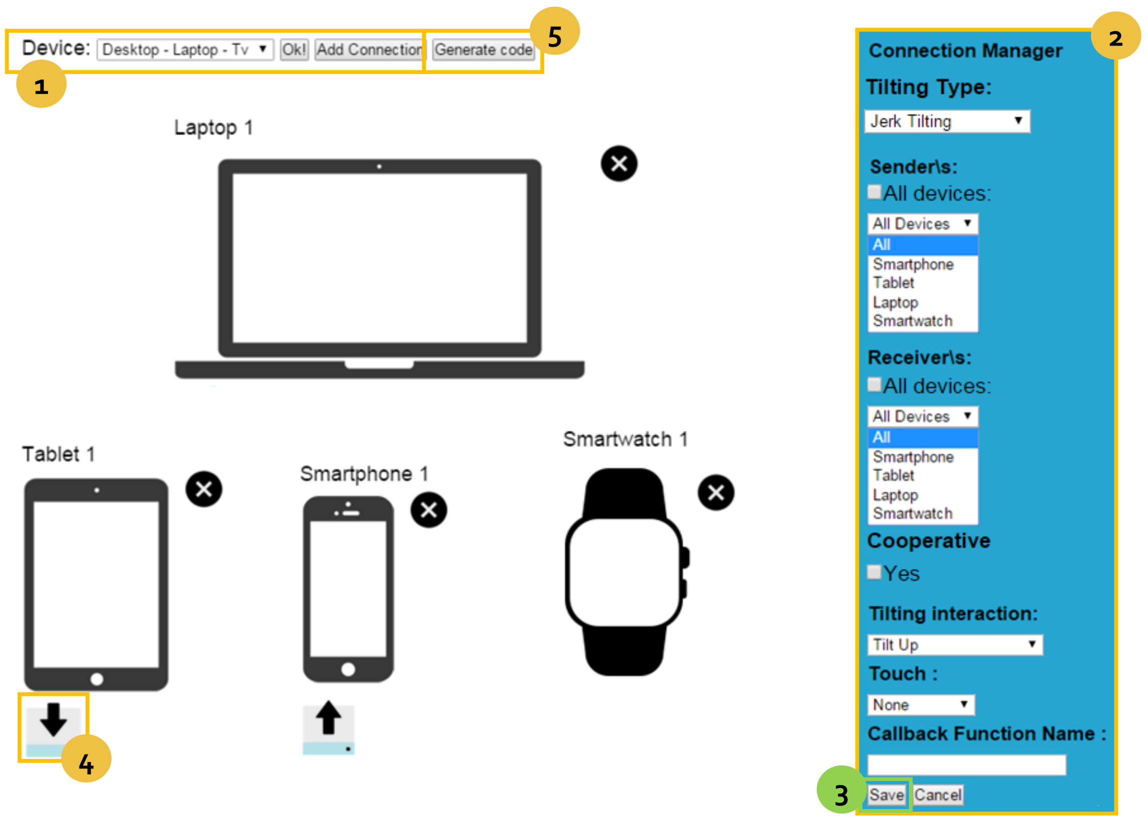


Figure 4.9: Screenshot of VCTAT.

4.2.2 The VCTAT Tool

With CTAJ and CTATC, developers can easily specify tilting behaviour for their cross-device applications. As discussed in the previous section, developers are only required to indicate senders and receivers of a specific motion gesture to extend the interactions available in their applications. Although this approach allows developers to easily manage interactions, the number of variables increases with the number of desired gestures. Since for each interaction, the developer is required to define a variable with four or more parameters, the code can be repetitive and this could lead to possible errors. For these reasons, we decided to help developers even further by proposing VCTAT, a visual tool that automatically generates CTAT objects (see Figure 4.9).

With VCTAT, developers can create CTAT instances by using a user interface. Devices and connections can be added by using the UI elements on top of the page (step 1). Four types of clients can be added: big screens (Desktop - Laptop - TV), tablets, smartphones and smartwatches. Every time a new device is added, it will be show in the center of the tool and it can be dragged and dropped anywhere in the page. The device is labelled with the name of its category and a number. This number is a counter that is increased every time another device of the same category is added. By clicking on the “Add Connection” button (step 2), developers can specify all the necessary parameters for the desired connection (sender, receiver, cooperative flag, tilting interaction, touch gesture and callback function). Once the connection is saved (step 3), an icon will be displayed to all the devices that are involved in that connection. If a device is the sender, an up-arrow symbol will be associated to that device. In contrast, if a device is the receiver, a down-arrow will be displayed. By clicking on the icon (step 4), the connection can be modified and saved. This operation will update all icons. If developers are satisfied with the connections created, they can click on the Generate Code button (step 5). At this point, the system will show the CTAT object created. The object can be copied in the clipboard and used in the application. VCTAT is a client-side application, it does not need external servers and it is implemented by using JavaScript and HTML.

4.2.3 Architecture and Implementation

As discussed in Section 4.1.3, after the development of Tilt-Gallery and a preliminary user study, we recognised a series of requirements to improve the development of such applications. Informed by these requirements we developed CTAT, a framework for the rapid development of tilting interactions for cross-device applications.

To manage connections among clients and servers, CTAT uses XD-MVC [100], a cross-device framework developed in our group to manage the exchange of messages between clients. XD-MVC requires a Node.js server, and it exploits Socket.IO and Peer.js to create a connection among different devices. When the browsers support peer-to-peer communication, XD-MVC uses this type of connection to reduce latency. This factor is particularly crucial for CTAT since a slow response of the clients to tilting gestures can easily decrease the user experience of these interactions. If peer-to-peer communications are not supported, XD-MVC will automatically recognise that and establish client-server communication. While there exist many frameworks similar to XD-MVC, we picked this technology for its ability to adapt type of communication (peer to peer or client-server) depending on the platform of the devices.

To use CTAT developers simply need to include it in their dependencies. Other than this installation, and the use of a Node.js server, no additional frameworks need to be added on the server or on the client. XD-MVC is directly included in the dependencies of CTAT and, therefore, the developers can be unaware of its presence.

As mentioned in Section 4.2.1, we distinguish devices into four categories: laptops (and big screens), tablets, smartphones and smartwatches. We categorise clients by using thresholds on their screen width and by looking at the browser used. For instance, all devices with a large screen width and a desktop browser will be categorised in the laptop class.

To describe the architecture of CTAT and discuss its implementation details we now present the execution flow of the framework when continuous tilting gestures, performed on a smartphone, are used to control an indicator on a large display (see Figure 4.10). In this example, we assume that the large screen is already connected to our application, while the smartphone is currently connecting to the URL.

When a new client is connected, CTAT recognises its type. If the device is the sender or the receiver of any tilting interactions, we keep this information and save the resolution of the client in an object that is stored on the server. This object is shared and synchronised with all clients involved in the communication. This factor is particularly important when a device disconnects since CTAT is then responsible for finding a possible new sender or receiver for the communication. In our example, if the mobile device disconnects from the application, CTAT finds, among the connected devices, another smartphone and gives it control of the cursors.

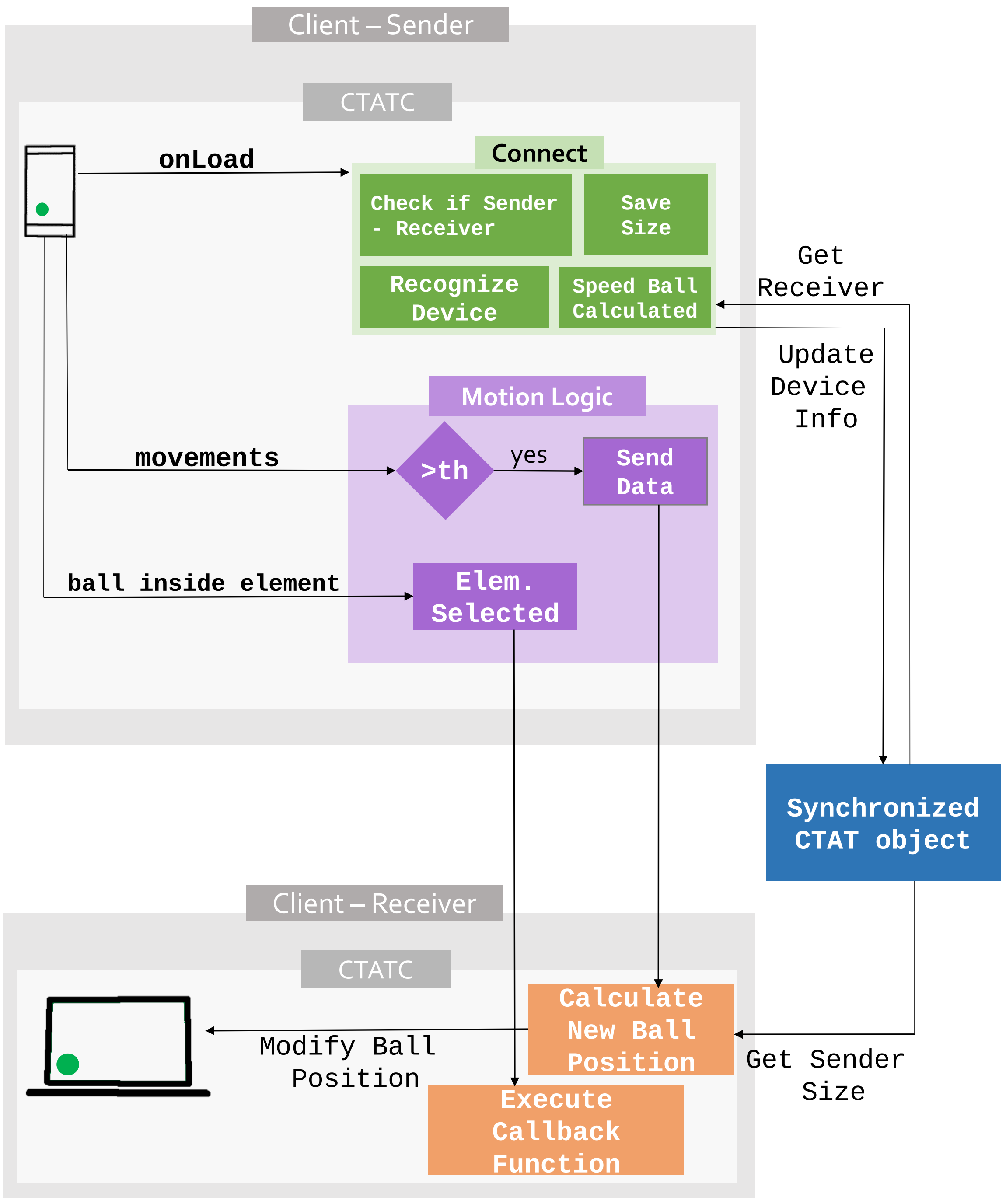


Figure 4.10: Example of CTATC execution flow with a smartphone as sender and a laptop as receiver.

By default, the speed of the cursor is increased according to the dimensions of the sender. The developer can change this factor via a specific parameter (`ball_factor`). This behaviour fixes the user experience issue we experienced during the preliminary user study. It is important to note that with the mapped container movement type, this factor is not taken into consideration since the ball does not move from one position to another but it jumps depending on the angle of the device (more detail on this movement type were discussed in Section 3.2.1). At this point, CTAT manages the motion events fired from the sender and sends the data to the receiver. For performance reasons, we send only movements that are bigger than a threshold. Small movements are filtered to avoid the congestion of the connection. If movements are detected, the new coordinates of the ball are sent to the receiver. On the laptop, the position of the ball is calculated by multiplying the ball position received from the sender with the quotient of the width of the laptop and smartphone. Although senders and receivers can offer different resolutions, this proportion allows a good match between the two devices.

When the indicator selects one of the elements, it triggers the associated callback function. We wrap this event to send a message to the receiver to notify it of the specific element selected.

4.2.4 Demonstrator Application

As shown in our preliminary user study (see Section 4.1), participants found tilting gestures fun to use. We believe that this factor can potentially improve user cooperation when interacting with public and semi-public screens. Motion gestures could mitigate the problem of poor user engagement of pervasive displays while allowing users to focus on the bigger screen rather than their phones. Furthermore, tilting interactions do not require the user to physically touch the display, allowing the screens to be placed almost everywhere in a room. For these reasons, we decided to exploit the advantages of motion gestures as well as show the capabilities of our framework by developing aCrossETH, an advanced cross-device gallery application.

The main goal of applications such as aCrossETH would be to engage users to interact with semi-public screens that might be available in offices or universities, as well as promote social interactions among colleagues and friends. The application allows users to upload, like, dislike and add to favourites pictures on large displays by using their mobile phones or tablets.

In aCrossETH, different devices have different roles and they show different content to the user. Figure 4.11 shows the categories of large screens and roles considered by our application. The sideshow display shows the six most popular pictures. The images are shown in full size one after the other with a fade animation. The popularity of an image uploaded by users is calculated by taking into account its freshness, number of likes and favourites. As shown in Figure 4.11, the slideshow role is taken by a projector display in social area. Voting screens are smaller displays that shows the most recent uploaded images in a grid layout. A QR code is shown on both types of large screens and they can be scanned via mobile phones to allow interactions.

The first mobile device that connects to the aCrossETH application will be assigned the role of the *controller*. All the other devices will be assigned the role of *viewers*. Figure 4.12 shows the interface users will see on *controller* and *viewer* devices. The

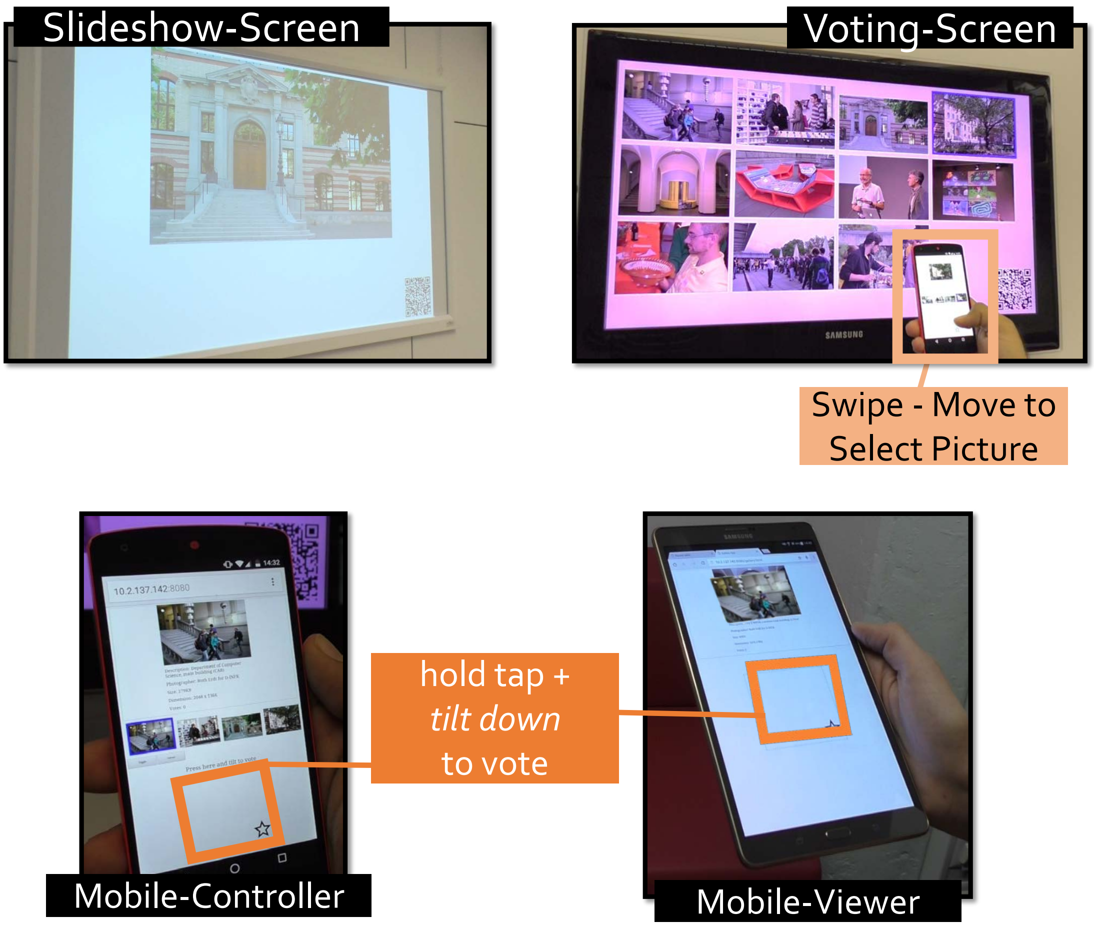


Figure 4.11: Different screens and roles of aCrossETH.

controller can browse images that are shown on the voting-screen via continuous tilting gestures. The selected image will be enlarged on the *voting-screen* and a blue border will be displayed on the *controller*. Both the *viewer* and the *controller* will show additional information (description of the picture, the name of the photographer, and the number of likes) of the currently selected image. On the viewer, only the currently selected picture will be displayed. In contrast, the controller has an overview of all images displayed on the *voting-screen*.

To like the selected picture, users can hold tap on a specific area of the screen and tilt the device down. To allow cooperation between viewers and, therefore, to improve user engagement, if more than one viewer likes the same picture at the same time, the number of likes is doubled. The more users cooperate, the higher are the chances that the image will be displayed on the *slideshow-screen*.

To upload a picture, users can tap on the upload button displayed on the mobile device. Users can select pictures they have in the gallery of their tablet or phone. Once the image is selected, aCrossETH will show a preview of the image. At this point, users can decide to share the picture with other viewers by performing a tilt left gesture (see Figure 4.13). This interaction will send the image to all devices that are now able to like or dislike the picture. This information is sent back to the owner of the picture that is now free to consider these likes and decide to upload the image or not. A tilt

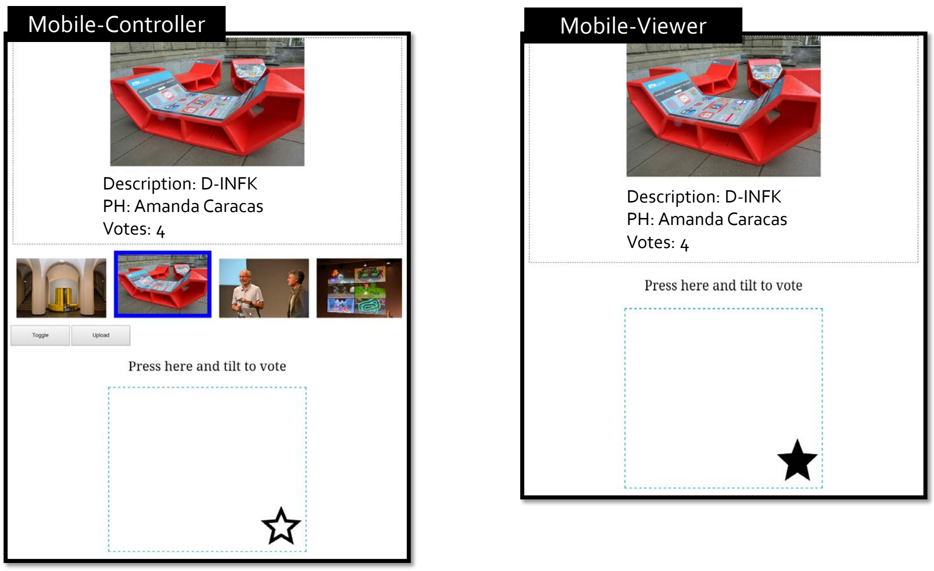


Figure 4.12: Screenshots of aCrossETH when viewed by a mobile-viewer and by a mobile-controller.

down gesture will send the picture to the *voting-screen*.

aCrossETH gave us the chance to test the features offered by CTAT. In contrast with our first attempt made in Tilt-Gallery, experimenting with different interactions was made easier thanks to the capabilities of our framework.

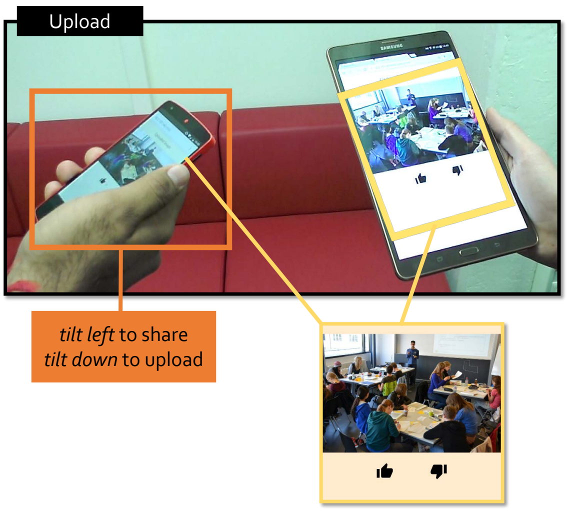


Figure 4.13: Upload interaction flow of aCrossETH.

4.3 Cross-device Continuous Movements Evaluation

By using the cross-device features offered by CTAT and the different continuous tilting interactions variants supported by TAT 2.0, we were able to further evaluate motion interactions. While the gestures supported by TAT 2.0 were inspired by previous research, these interactions were mainly studied in single devices cases. For instance, researchers have compared *velocity* and *position-based* continuous tilting gestures to browse a 1D menu [162], or to select elements on a mobile devices [205], however, these interactions have not yet being evaluated in cross-device scenarios. While position-based solutions have been proved as better variants of continuous tilting gestures, we wanted to test if these findings coincide also in cross-device applications.

To reach this goal, we decided to compare three variants of continuous tilting gestures to control an indicator on a large screen remotely. We evaluated constant move, balance board and mapped container movements type. The study was performed by 14 participants, five males and nine females (average age: 30.2, std.: 8.8).

4.3.1 Study Design and Tasks

As similarly studied by Boring et al. [23], in our study participants were required to select several elements shown on a large screen by moving a mobile device. During the study, the display was placed on a desk and the participant was seated two metres from the screen. As similarly discussed for the preliminary user study of CTAT (see Section 4.1), we recognise that in a real-case scenario, users would be standing near to a semi-public or public screen. However, the length of the study could strain the user and, therefore, influence negatively the result of the study.

The mobile device used by participants was a Huawei smartphone (model: GRA-L09), and a 30” HP LCD screen (model: LP3065) with a density of 3.96 pixels per millimetre. We decided to evaluate the three movement types with the device held in portrait mode. While it would have been interesting to compare the movements in different states (portrait and landscape), this would have lengthed the duration of the study considerably. Since users usually held smartphones in portrait mode, we decided to use this state to study the movements. For this reason, the area where the ball can be moved is a portrait box with a width of 1100px and a height of 870px. The box and the ball are displayed on both screens (the mobile phone and the display).

Before the participants started the study, they were explained the goal of the evaluation and described the application users were going to use. After a period of training on the application and interaction techniques, participants could start with the study and perform the various tasks.

At the beginning of each task, the ball was located at the centre of the box inside an orange rectangle. The ball started moving only when users performed a hold tap gesture on the mobile device. Once the ball moved, a blue rectangle would appear, and participants were required to select it by moving the device. If the indicator touched the target element, the colour of the element would change. After the selection was performed, users had to move the ball back to the orange rectangle to start a new trial. It is important to note that we did not consider the time to return to the orange box in the total time to complete the trial.

Participants had to select seven different targets ten times each. As we can notice from Figure 4.14, we used different target dimensions (50, 75 and 100 pixels) and distances from the centre. Moreover, we also studied two selection delay variants: zero (0-s) and one second (1-s). In 0-s tasks, elements were selected as soon as the ball collided with the target. In this case, the colour of the element will change from blue (not selected) to green (selected) directly. In contrast, in 1-s tasks, users had to maintain the position of the ball for one second to select the target. In this case,

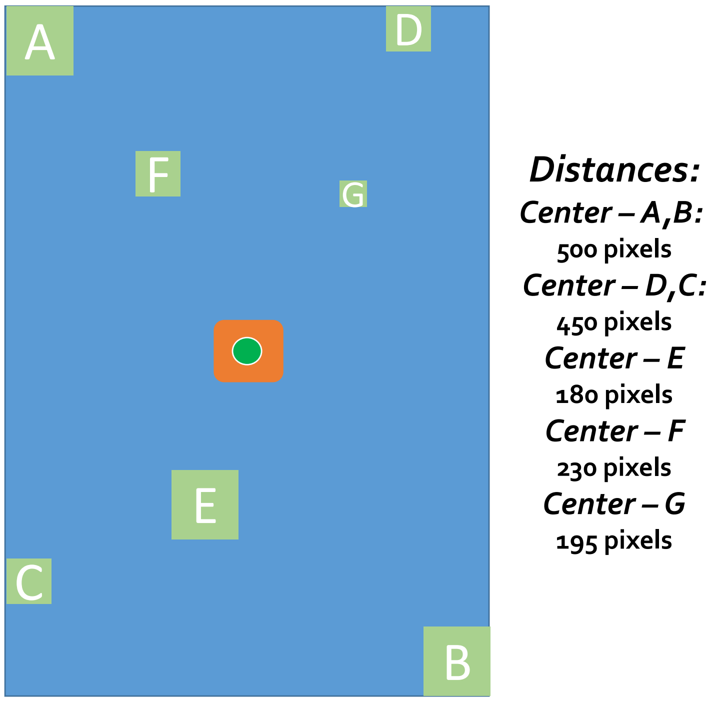


Figure 4.14: Target elements and their distances to the centre of the page. Large: A, B, E. Medium: C, D, F. Small: G.

when the ball collided with the element, the colour of the target changed from blue to orange (selecting), and if this state was maintained for one second, the colour would then change from orange to green (selected). In 0-s tasks, elements were selected as soon as the ball collided with the target. For this reason, users were not required to correct overshoot errors. In contrast, in 1-s tasks, if the user selected the right element but were not able to select it in time, we asked them to correct this error and go back to the element again until it was correctly selected.

Users could stop the ball from moving in two ways: by falling into a dead zone or by a touch-up event. As discussed in Section 3.2.1, a dead zone is a threshold under which the ball will not move. Furthermore, we decided to use hold touch gestures with continuous tilting interactions to give more control to the user.

Overall, the study considered the following independent variables: 3 Interaction techniques (constant move, balance board an mapped container), 2 selection delays (zero and one second), 7 different targets, selected 10 times each. For a total of 420 trials for every participant.

For each user, we shuffled the order of the interaction techniques, the selection delays and the order of the targets. We logged the trace of the indicator as well as the timestamp for every main event. Moreover, we recorded the study with a video camera. After each interaction technique tasks, we asked participants to rate the movement type. Finally, at the end of the study participants filled in a questionnaire asking personal background information.

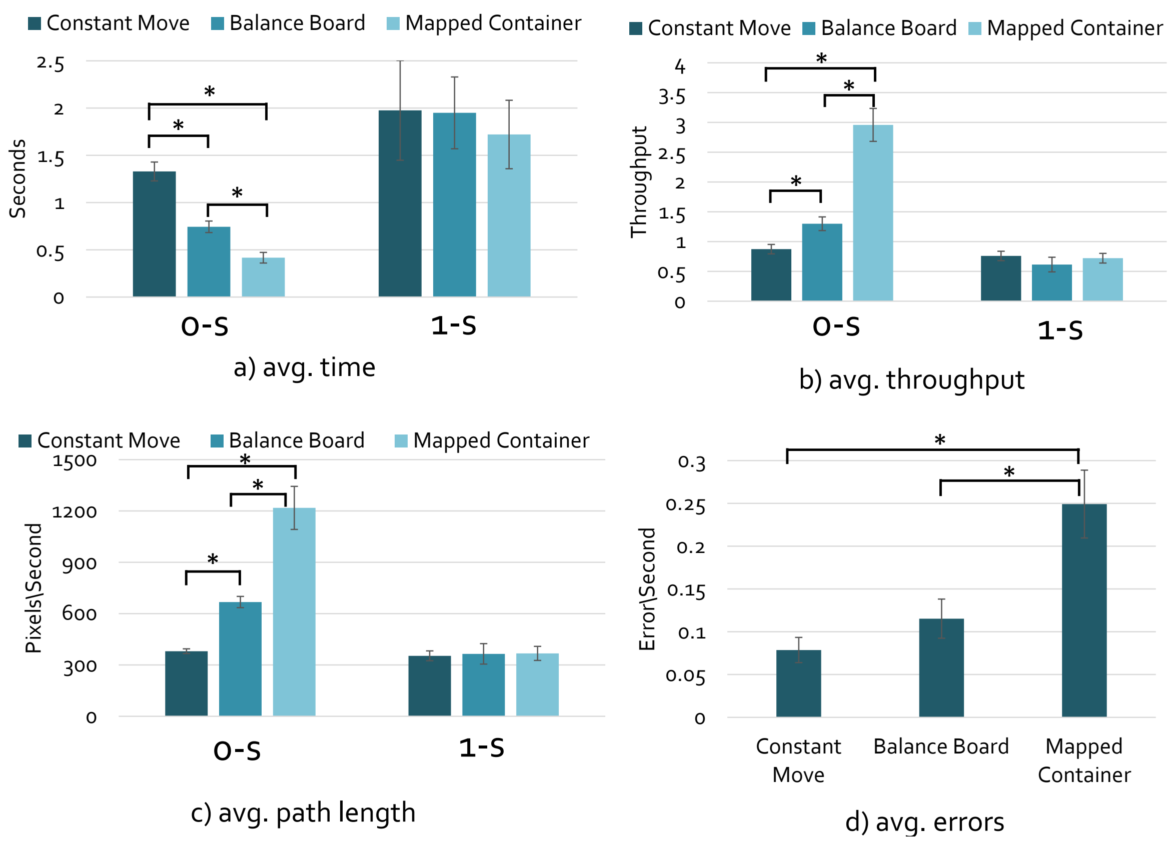


Figure 4.15: Performance results in terms of: a) Average time to select one element; b) Average throughput; c) Average path length in pixel/second; d) Average overshoots. Error bars represent standard errors. The * indicates a statistical significance among pairs ($p<0.05$)

4.3.2 Results and Discussion

We compared constant move, balance board and mapped container movement types under four different measures: the average time to select elements, the throughput, the number of pixels visited by the ball (path length) and the number of errors performed by the user (overshoots). In Figure 4.15, we summaries the results we obtained for each of the four measurements. Figure 4.15 also shows the repeated ANOVA test results that we executed for each series of tasks.

We calculate the throughput using the following Fitts' law formula adaptation proposed by Teather and MacKenzie [205].

$$\begin{aligned} TP &= \frac{ID}{MT} \\ ID &= \log_2\left(\frac{A}{W}+1\right) \end{aligned} \tag{4.1}$$

We divided the index of difficulty (ID) with the average time of each movement type (MT). The index of difficulty uses the distance to the target (A) and its width (W). Since elements A, B, C and D are placed in the corner of the page (see Figure 4.14) we defined their width to be infinite and therefore, their ID to be 0. For all the other elements, we considered their width as the diagonal distance between the centre of the container and the target.

In 0-s tasks, the mapped container movement type performed better in terms of time, throughput and path length. The second best continuous tilting variant was balance board, followed by constant move. Given the nature of 0-s tasks, overshoots were not possible since an element was considered selected as soon as the ball collided with the target.

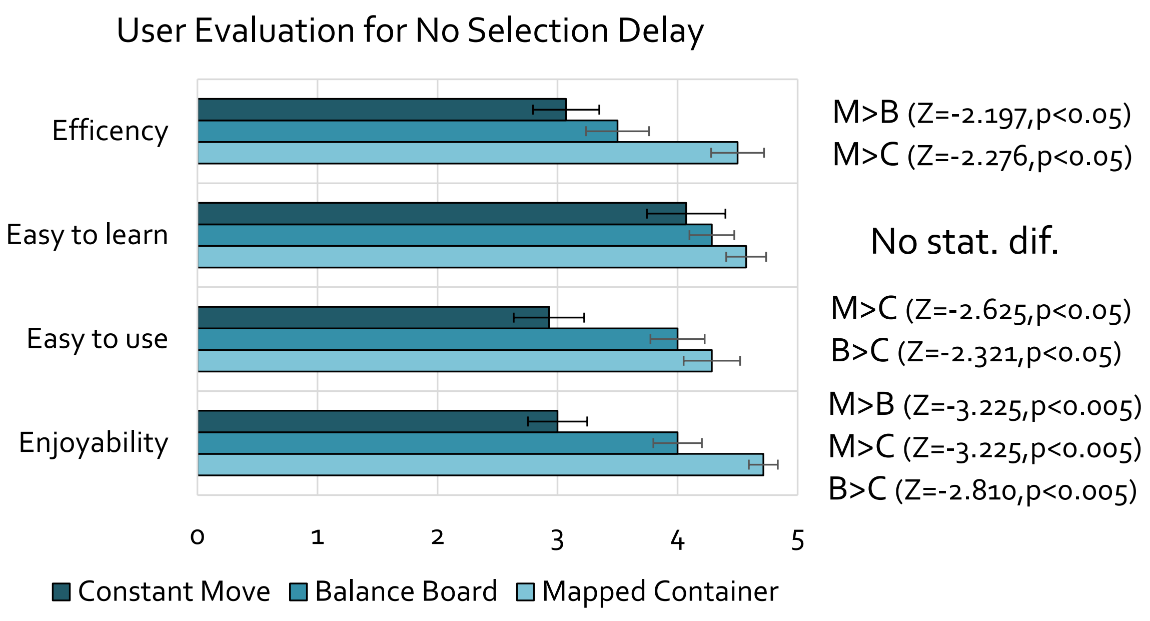
While we see a clear statistical difference between movement types in 0-s tasks, this factor is less prominent in 1-s cases. Overall, participants were required to be more precise and, therefore, slower when targets had to be selected for a larger amount of time. On average, users were 45% faster in 0-s tasks, they visited more pixels in the page and had higher throughput. Constant move was the interaction technique that suffered the least in delay tasks. In contrast, the mapped container and the balance board movement types were considerably worse when users had to select targets for one second.

Concerning overshoots, we kept track of the number of times users had to re-select the target element and then divided this number by the total task time for all users. As we can be seen in Figure 4.15, mapped container was more prone to errors than balance board and constant speed. No statistical difference was found between balance board and constant speed. For all four measurements, we did not find any statistical differences between different element sizes and distances from the centre.

At the end of each task, we asked participants to rate on a scale from one (completely disagree) to five (completely agree) how efficient, easy to use, easy to learn and enjoyable the three movement types were. We then ran a Friedman test and, if possible, Wilcoxon tests on the results obtained from the questionnaire. We summarise these findings in Figure 4.16.

Mapped container and balance board were perceived as better interaction techniques than the constant move movement type. Although in terms of performance the three movements were similar in 1-s tasks, users preferred mapped container and balance board also when targets had to be selected for one second. However, we noticed some differences between user ratings in 0-s and 1-s delay questionnaires. In terms of enjoyability, we could find a distinction between the three movements in 0-s tasks: mapped container was the most preferred movement type, balance board the second and, finally, constant move was rated as the least preferred interaction. In contrast, in 1-s tasks, we did not find any statistical differences between balance board and mapped container. Similarly, mapped container was rated as more efficient than balance board in 0-s tasks; however, we did not find this distinction when the targets had to be selected for one second.

At the end of the study, we asked participants to give their opinion on the study and suggest possible applications where they would like to use continuous tilting interactions. Overall, users stated that the study, and the gestures proposed, were interesting. Among the proposed applications, participants expressed their interests in using motion gestures in web browsing or scrolling tasks on their TV. Moreover, users liked the use of tilting gestures in combination with tap hold interactions, stating that this gave them more control over the movement. Furthermore, we asked participants to give general feedback about each movement type. While commenting on the mapped container con-



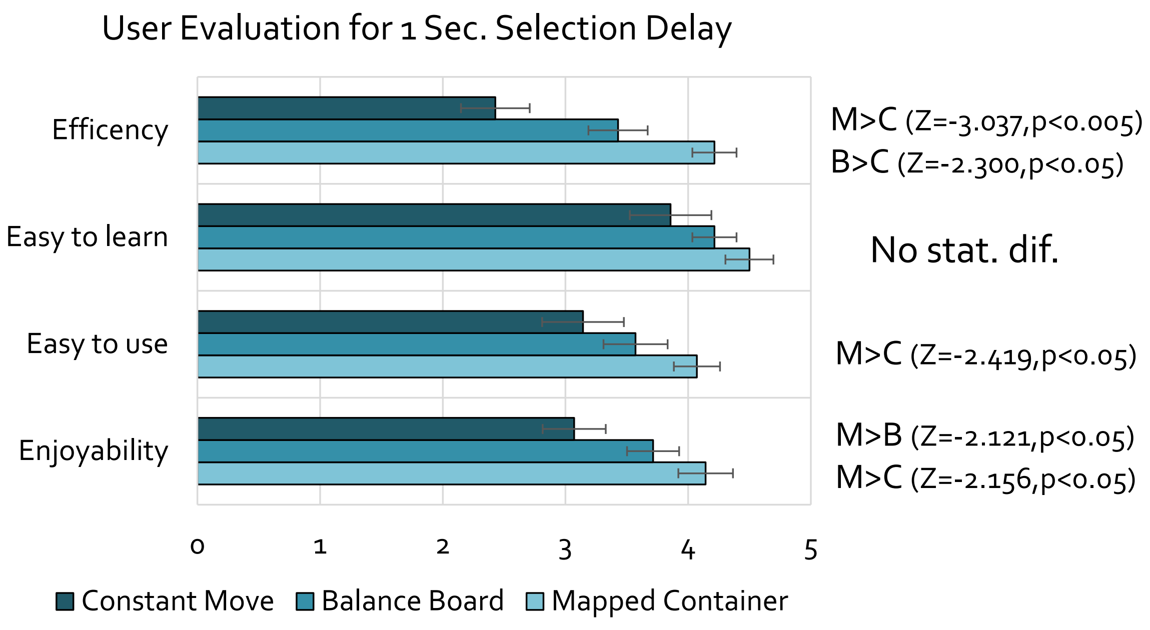


Figure 4.16: Average responses of post-task questionnaire for no selection delay and one second selection delay. Error bars represent standard errors. Charts also reports the results of the Wilcoxon tests for (C) constant move, (B) balance board and (M) mapped container.

tinuous tilting interaction, users stated: *"Fun, makes [me] want to master it [...]"*; *"[...] my favourite option because of its efficiency and predictability"*. Although mapped container did not perform better than balance board in 1-s condition, participants rated this movement as more enjoyable, efficient and easy to use than the other interaction techniques.

Similar implementations of our mapped container movement types have already been shown to be efficient in single device scenario [205]. Our study confirms these findings also in cross-device environments. However, the performance of mapped container and balance board dropped drastically when users had to select elements in 1-s tasks. These results differ with the findings discussed by Teather and MacKenzie [205]. In their study, mapped container performed better despite a longer delay in selection time.

Given these results, we believe that continuous tilting interactions in cross-device applications could be better suited to simulate hovering and scrolling actions. Selecting elements for a prolonged period of time can require too much precision from the user. Moreover, a small network delay can easily cause overshoots and therefore, negatively influence the overall user experience.

4.4 Discussion

In this chapter, we presented CTAT, a framework for the rapid development of tilting interactions in cross-device web applications. CTAT aims at encouraging developers in using alternative forms of interactions when more and diverse devices are involved in a communication.

A preliminary user study informed the design of the framework. The feedback received from participants, as well as experience gathered during the development of Tilt-Gallery, raised some challenges that we wanted to fix with CTAT. One of the primary goals of our framework was to allow easy experimentations of tilting gestures in cross-device scenarios. CTAT drastically diminishes the lines of codes required to develop tilting gestures, and it automatically adapts the interactions on the size of the mobile device involved. Furthermore, a visual tool allows users to generate *tilt links* among devices, without the need to write any lines of code.

For these reasons, we believe that CTAT could further push researchers in studying these alternative forms of interactions in cross-device applications.

With CTAT, we were also able to evaluate different variants of continuous tilting gestures in scenarios not yet studied. As similarly found by researchers [205, 162], our study shows that mapped container movements perform better than other continuous interactions. However, in contrast with Teather and MacKenzie results [205], in our cross-device scenario, we found no statistical differences among movements when targets had to be selected for a prolonged period of time.

5 MyoShare

The number of smart devices that people own is increasing dramatically. As a result, users often would like to copy data among their smartphones, tablets and laptops. For instance, users might be browsing the web on their smartphone on a train and would like to copy a link to their laptop at home in order to continue reading a news article later. While systems such as iCloud, Dropbox, Google Drive and messaging apps allow users to share data across their devices, they require the user to copy the desired content, switch the application window or directory, and then finally paste the data. Overall, these approaches do not offer a seamless sharing process.

The use of alternative approaches for information sharing tasks has been studied widely in research. In this context, tilting and mid-air gestures have been proposed as intuitive ways of interacting in cross-device scenarios [176, 220, 129]. Pointing or performing a wave gesture toward the desired device, are intuitive interactions for sharing data. An analysis of the research literature, revealed that mid-air gestures are mainly exploited when devices are physically close to each other. However, as shown by a survey conducted in our group [52], users do not always carry all their devices with them. In this scenario, the form of a mid-air gestures cannot be inferred by the physical proximity of the target device.

For these reasons, we decided to investigate on the use of gestures, typically applied in physically-aware scenarios, to target remote as well as co-located devices[1]. We started our investigation by exploring the use of mid-air gestures to share data across co-located and not co-located devices. Key questions that we wanted to address were first, how users would associate mid-air interactions with devices that are remote or co-located and what mental association users would make in mapping gestures to devices in this scenario. Second, we were interested in how mid-air gestures would perform when evaluated against more common interactions such as menu selections, tap or keyboard shortcuts and speech.

To answer the first question, we carried out an elicitation study with 16 participants

[1]Earlier versions of parts of this Chapter were originally published as Di Geronimo et al. [54, 55]

that discussed and suggested a set of mid-air gestures for copyng data among different devices. The study gave us the chance to understand the associations that users made to map interactions to co-located and remote devices. Furthermore, we developed MyoShare, a tool that allows users to share data among devices via mid-air gestures. As the name suggests, MyoShare uses the Myo armband to detect mid-air gestures. MyoShare also offers other forms of interaction techniques (menu selection, tap shortcuts, keyboard shortcuts and speech) that gave us the opportunity to compare mid-air gestures against them.

In Section 5.1, we start by discussing the design, the tasks and the results obtained by our elicitation study. We present the main concepts, architecture and implementation details of MyoShare in Section 5.2. Furthermore, we carried out two user studies to evaluate mid-air gestures when targeting and sharing data across devices, and we present the design and the results of these studies in Section 5.3. Finally, we conclude this chapter with final remarks in Section 5.4.

5.1 Elicitation Study

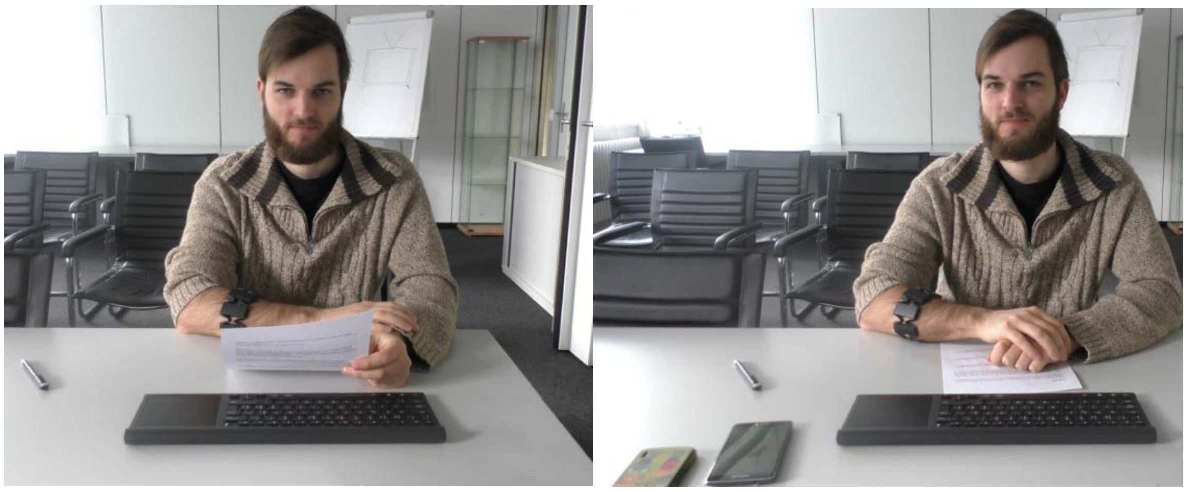
Via an elicitation study, we wanted to experiment the use of mid-air gestures when devices are not physically close to one another. Moreover, the results of the study and the feedback received from users, would inform the design of MyoShare. The elicitation study addressed the following research questions:

- (RQ1) How do users usually share content (images, links, text) among their devices?
- (RQ2) What functionality would users like to have in this scenario?
- (RQ3) Which mid-air gestures would users produce to share data across devices when they are remote or co-located?
- (RQ4) How do users associate the proposed gestures and the devices?
- (RQ5) Does the physical location of the device influence the proposed gestures?

In Section 5.1.1 we discuss the design and tasks of the elicitation study. We present the gesture set defined by users in Section 5.1.2, where we also report on the reasons users gave for their suggestions. Finally, in Section 5.1.3, we discuss the results obtained and report how the elicitation study informed the design of MyoShare.

5.1.1 Study Design and Tasks

The scenario we envisaged for the study was the following: the user is working on their desktop computer and would like to copy content to their mobile devices. To simulate this scenario, we asked participants to sit near a desk on which there was a keyboard with a trackpad; however, no monitors or computers were present (see Figure 5.1 (a)). We placed one or more mobile devices on the desk depending on the task to be performed (see Figure 5.1 (b)). We also asked users to wear the Myo armband on their non-dominant hand to inform participants about the boundaries of the gestures



(a) Setup for task 1, 2, 3 and 4 (b) Setup for task 6

Figure 5.1: Setup of the elicitation study.

and how the wearable device is able to detect mid-air gestures. However, the wearable device was turned off. Moreover, to avoid possible biases, we did not present the five predefined gestures of the Myo armband. A camera placed in front of the desk recorded the gestures that users proposed and their comments.

As similarly done in other elicitation studies reported in the literature [176, 121, 218], participants were not required to use any systems and the gestures they performed during the study did not trigger any actions. Users were encouraged to suggest their own desired gestures without considering the technical feasibility of the proposed interaction.

At the beginning of the study, the researcher explained the goal of the evaluation and asked users to fill in a questionnaire about their background and their current habits when copying data among their personal devices. After participants performed the six tasks of the study, they filled in a post-task questionnaire with their opinions about our scenario and on the use of mid-air gestures to target and send data to devices.

To answer the research questions defined above, we considered six tasks. In the first four tasks, we asked participants to propose gestures to copy an image from their desktop computer to: smartphone A (task 1); a second smartphone, smartphone B (task 2); a tablet C (task 3), and to all (broadcast, task 4). In each of these tasks, we assumed that the devices were not co-located and users did not know their exact physical location.

For task 5, we placed smartphone A on the desk and asked users to re-define, if desired, the gesture to copy the image from the desktop computer to the smartphone on the table. Similarly, in task 6, we placed smartphone B near to smartphone A (see Figure 5.1 (b)) and asked users to target smartphone B to copy the picture. It is important to note that we did not shuffle the order of the tasks among participants intentionally. After the first four tasks, participants had already associated devices to gestures and, to answer RQ5, we wanted to see if these associations would change if the devices were then co-located with the user. The scenario of the study and the description of tasks were printed and given to every participant.

Our study had 16 participants, 3 females and 13 males with an average age of

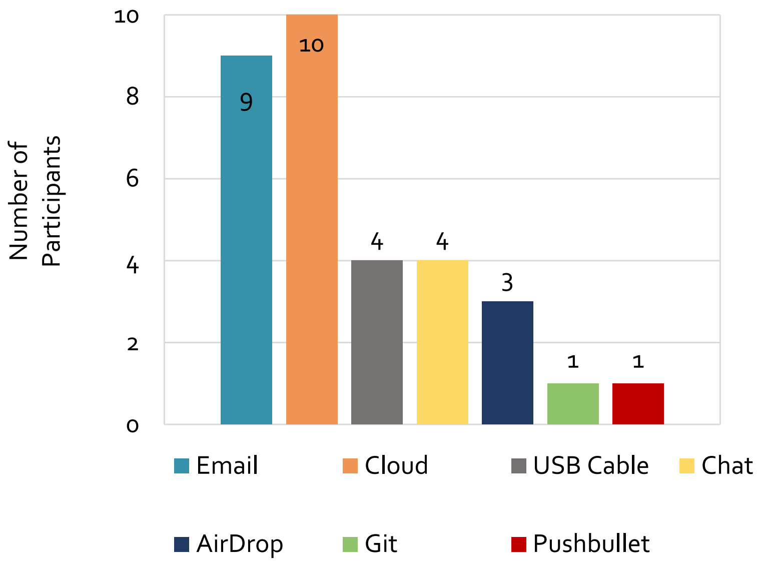


Figure 5.2: Tools that participants use to copy content among their devices.

26.8 (std. 4.4). In the background questionnaire, all participants stated that they use laptops and mobile devices several times per day. However, none of our participants owned a smartwatch or an armband, and only one user owned a fitness bracelet. While smartwatches and similar wearable devices are still not highly popular, it is important to note that the study was conducted in 2016, therefore, this data could be influenced by the period in which the study was carried out.

5.1.2 Results

In this section, we present the results of our study by first presenting users' habits when sharing data across their devices and then discussing the proposed gesture sets and the reasoning behind the suggested interactions.

User Habits

To tackle research questions RQ1 and RQ2, we asked participants how they currently copy content from their desktop or laptop computers to their mobile devices. In Figure 5.2, we can see all the tools users exploit to perform such tasks. Cloud services such as Dropbox, Google Drive and iCloud are the systems used most to copy data among devices. Some participants, also use chat or email systems to send data. For instance, users message themselves or close friends on Messenger[2] and WhatsApp[3] to access content on other devices.

We asked users if they feel that being able to target a specific device is an important feature when performing tasks to share information among their devices. 11 out of 16 participants felt that it is very or fairly important. Despite this need, however, as we can notice from Figure 5.2, most of the currently used tools only allow broadcast

[2]Facebook Messenger website: https://www.messenger.com. Accessed on May 2018.
[3]WhatsApp website: https://www.whatsapp.com. Accessed on May 2018.

solutions. Copying an image in a cloud directory, or sending the picture on a messaging system will copy data to all devices.

For these reasons, it is clear that there is a gap between users preferences and the currently available tools. Although copying resources between devices is a task performed mainly on a weekly or monthly basis (7 out of 16 participants copy data weekly, 6 less than on a weekly basis), we believe that these tasks might be performed less frequently than desired. By using cloud or chat systems, users need to stop their current workflow, switch app and copy-paste the desired content. The current tools available require too many steps for a simple copy operation and could discourage users from sharing content.

Mid-air Gesture Set

A total of 96 gestures (6 tasks for each of the 16 participants) were collected and classified into different macro-categories. We identified disjointed classes of gestures for the first three tasks (copy data to remote devices, smartphone A, B and tablet C), for the fourth task (broadcast data to all devices) and for tasks 5 and 6 (copy data to co-located devices, smartphone A and B). We manually classified gestures into different macro-categories by watching the recorded videos taken during the study.

As we can notice from Figure 5.3, for task [1, 2, and 3], we identified four different classes of gestures: poses, circular, directional and sequential. A pose gesture, is a static configuration of the hand (see Figure 5.4 (a)). Among all participants, only one user exploited the use of poses to target devices. Moreover, poses were the only static gestures proposed [110], all other gestures proposed required some movement of the hand, arm or fingers. Circular, directional and sequential gestures are all dynamic interactions [110]. The user who proposed poses for coping data to a target device stated that the fixed configuration of his hand and fingers simulated keyboard shortcuts that he uses on a daily basis.

Two participants proposed circular gestures to target and copy data to remote devices. We defined a circular gesture as an interaction that users perform by drawing an entire or partial imaginary circle in the air with their arm, hand or fingers (see Figure 5.4 (b)). Participants draw circles in clockwise or counterclowise direction, parallel or perpendicular to the table to distinguish between the three devices (smartphone A, smartphone B and tablet C). All participants started by proposing clockwise direction of the circles to target the first smartphone. Moreover, all participants performed circular gesture using their arm.

Eight out of 16 users of our study, suggested directional gestures as mid-air interactions for the first three tasks. We defined directional gestures as horizontal or vertical shifts of the users' hand or arm. Users exploited different direction of the interaction to distinguish between the devices. All participants that suggested directional gestures, mapped smartphone A with an *out* interaction or perpendicular motion of the hand towards the shoulder.

For instance, a wave gesture of the user hand will usually start from the left to the right, if right-handed, and from right to left if left-handed. Users commented that the importance of the device could influence the direction of the gesture. Mid-air interactions near to the participant dominant hand could indicate the most important device such as a personal smartphone (see Figure 5.5 task 1). In contrast, gestures towards

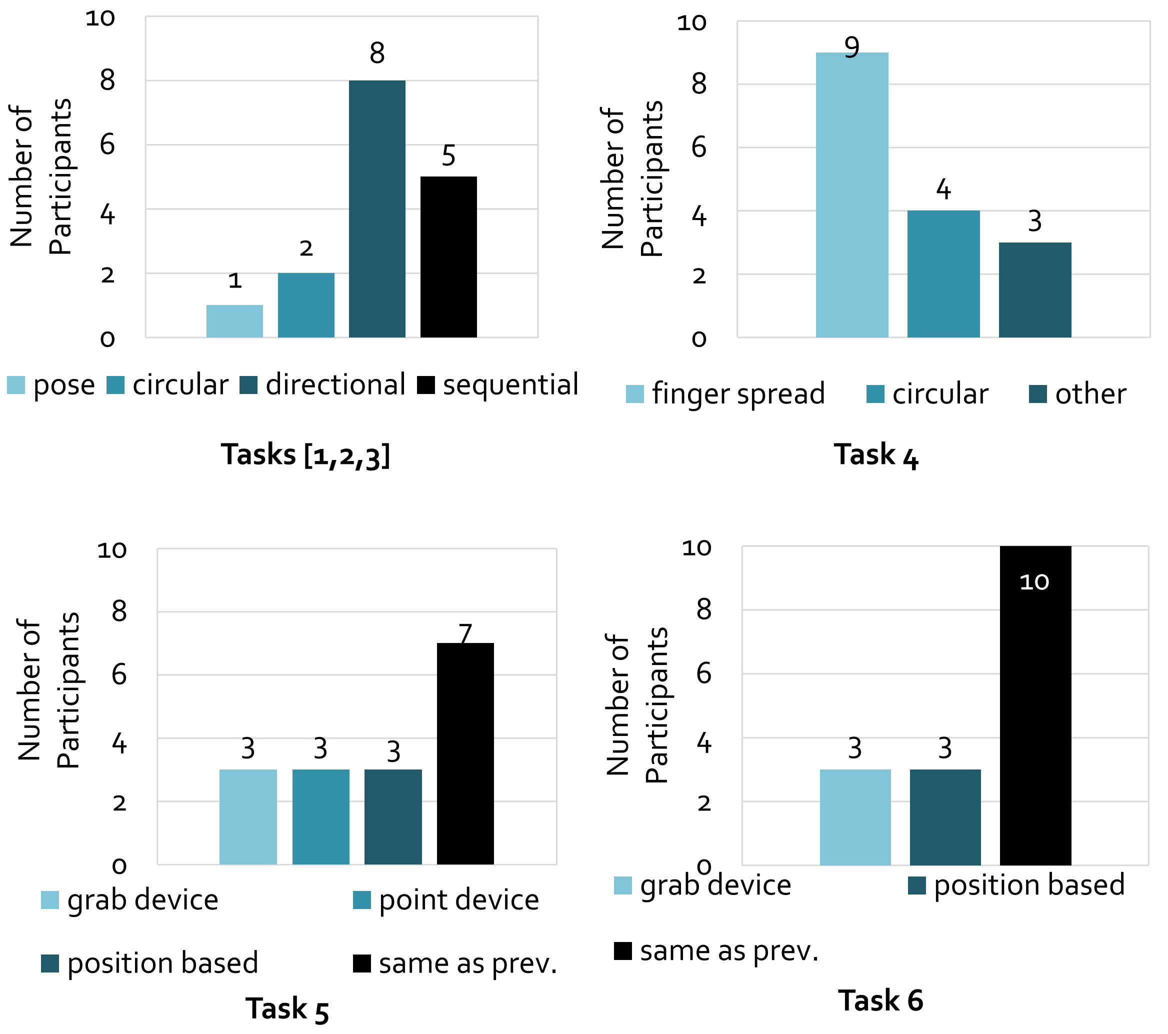


Figure 5.3: Participants suggested gestures for task [1,2,3], task 4, task 5 and task 6.

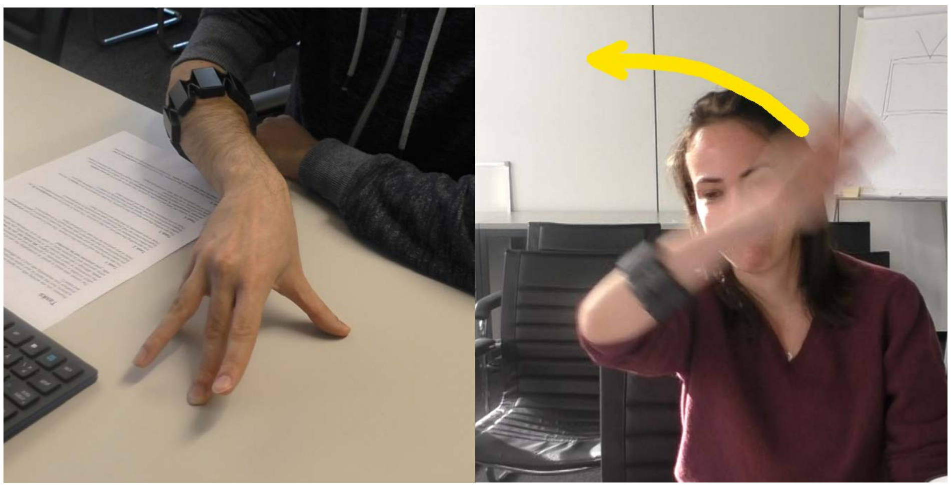

(a) Pose (b) Circular

Figure 5.4: Two examples of suggested gestures.

the non-dominant hand could indicate the least important devices (see Figure 5.5 task 2 and 3).

Five out of 16 participants, proposed sequential interactions for the first three tasks. We defined a mid-air gesture as a sequential interaction if it was part of a sequence. For instance, users have a ranking of the devices in their mind and associate numbers or letters to each target by counting it with their fingers or drawing a number or letter in the air. As similarly found for directional gestures, users associated smaller numbers (or letters, e.g. A, B) to more important devices while devices with a lower ranking were associated with larger numbers.

The majority of our participants (nine out of 16), suggested a finger spread gesture to send the image to all devices (see Figure 5.3 task 4). Circular gestures, such as drawing a spiral or complete circle in the air, were proposed by four of our users (see Figure 5.4 (b)). The remaining three participants suggested a wave gesture of the hand in some direction to broadcast the picture. We classified these gestures in the "other" category. With the first three tasks, we wanted to tackle research questions RQ3 and RQ4 and, therefore, understand user preferences in mapping mid-air gestures to remote

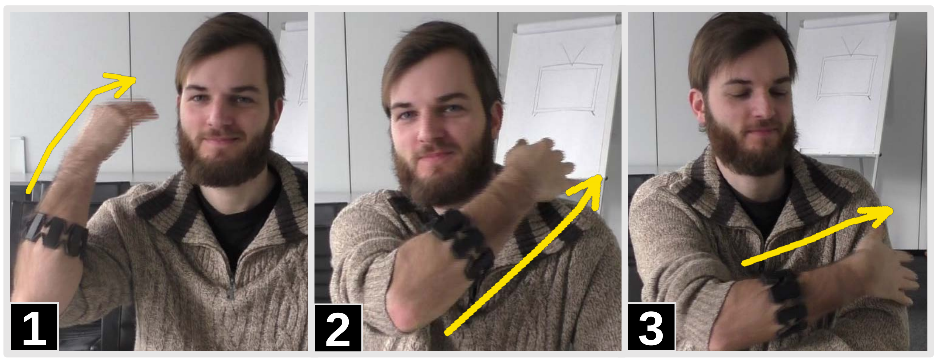


Figure 5.5: Example of directional gesture for task 1, 2 and 3.

devices. We considered co-located cases, and tackled RQ5, in the last two tasks. In task 5 (see Figure 5.3), seven participants decided not to change the gesture proposed for task 1 (copy image to smartphone A) although, this time, the device was placed on the desk and therefore, reachable by the user. These participants commented that they were not keen to learn new gestures for the tasks since their previously proposed mid-air interactions could work in both co-located and remote scenarios. On the other hand, the majority of our users proposed new gestures for this task. These new interactions are similar to the spatially-aware gestures suggested by Rädle et al. [176]. For instance, users grabbed the device, pointed with their finger towards it or waved their hand in the direction of the phone. However, when another device was placed close to smartphone A in task 6, some users changed their mind and reused the gesture proposed for task 2 (see Figure 5.3). Since devices could no longer be distinguishe with the new gestures defined, three users found the remote interactions more appropriate for completing the task. The majority of our participants (nine out of 16), suggested a finger spread gesture to send the image to all devices (see Figure 5.3 task 4). Circular gestures, such as drawing a spiral or complete circle in the air, were proposed by four of our users (see Figure 5.4 (b)). The remaining three participants suggested a wave gesture of the hand in some direction to broadcast the picture. We classified these gestures in the “other” category. With the first three tasks, we wanted to tackle research questions RQ3 and RQ4 and, therefore, understand user preferences in mapping mid-air gestures to remote devices. We considered co-located cases, and tackled RQ5, in the last two tasks. In task 5 (see Figure 5.3), seven participants decided not to change the gesture proposed for task 1 (copy image to smartphone A) although, this time, the device was placed on the desk and therefore, reachable by the user. These participants commented that they were not keen to learn new gestures for the tasks since their previously proposed mid-air interactions could work in both co-located and remote scenarios. On the other hand, the majority of our users proposed new gestures for this task. These new interactions are similar to the spatially-aware gestures suggested by Rädle et al. [176]. For instance, users grabbed the device, pointed with their finger towards it or waved their hand in the direction of the phone. However, when another device was added near to smartphone A in task 6, some users changed their mind and reused the gesture proposed for task 2 (see Figure 5.3). Since devices could not be distinguished any more with the new gestures defined, three users found the remote interactions more appropriate to complete the task.

5.1.3 Discussion and Input for MyoShare

All participants were enthusiastic about our envisioned scenario, and all stated that they would like to use mid-air gestures to share content in the future. Overall, all users were able to complete the tasks and did not find any particular problems during the study. These results informed the design of MyoShare.

Among all the mid-air gestures proposed, we picked the most popular interactions to target and copy data to devices. Since the majority of our participants suggested gestures in the directional category, we decided to pick these types of interactions to copy data to a specific device. For broadcast tasks, as suggested by users, we decided to allow the use of the finger spread gesture.

At the end of the study, we asked participants to propose possible features that

MyoShare could have. Some users expressed the need to group devices into categories. For example, users could label one or more smartphones or tablets as work or personal devices. For this reason, MyoShare allows the registration of devices into groups and users are then free to map groups to a specific gesture. We also asked participants if the type of data sent (text, image, video or link) could influence the gesture. Since the majority of our users did not find this feature relevant, MyoShare does not differentiate between types of data and the gestures are only influenced by the target device. In the next sections, we will discuss more details of the features offered by MyoShare and how directional mid-air gestures performed against more common interaction techniques such as menu selections, tap or keyboard shortcuts and speech.

5.2 The System

MyoShare is a system that allows users to select content on web applications and perform mid-air gestures to send data to co-located or remote devices. MyoShare also offers other interaction techniques (menu selection, speech, shortcuts, Leap gestures and tilting interactions) to target and copy content among devices. With MyoShare, users are free to map their desired interaction to a single device or a group of devices. After selecting images, text, videos, links or phone numbers, users can copy the data to devices by performing the associated gesture to them. MyoShare is able to distinguish different types of content and allow special actions for each category. For instance, if a phone number was selected and sent to a smartphone, once the user accesses this data on the target device by simply clicking on the item, MyoShare will redirect the user to the phone dialogue with the number already copied and ready to be used.

Our system is composed of four main components: an Android application that allows users to send data (via a web view) and to browse content received by other devices; a Chrome extension to send data from desktop computers, a web application, and a central server.

In Section 5.2.1, we present our cross-scenario in detail and discuss the advantages of MyoShare. In Section 5.2.2, we describe our approach by showing the mobile and desktop applications from the user perspective. Finally, in Section 5.2.3, we present the architecture and implementation details of our system.

5.2.1 Scenario

The scenario that we envisaged when studying the use of mid-air gestures in cross-device web applications can be summarised as follows:

> *Maria is a web designer, and she is using her desktop computer at work when she finds the website of an artist that she likes. The website contains some pictures of some of the artists' latest works, text describing the biography of the author and contact information. Maria would like to read the biography later at home and, therefore, selects the link of the web page and performs a wave-out gesture to target her personal desktop computer at home. Maria also found the artists' latest works inspiring, and she thinks that this might give her some ideas for her next project. For this reason,*

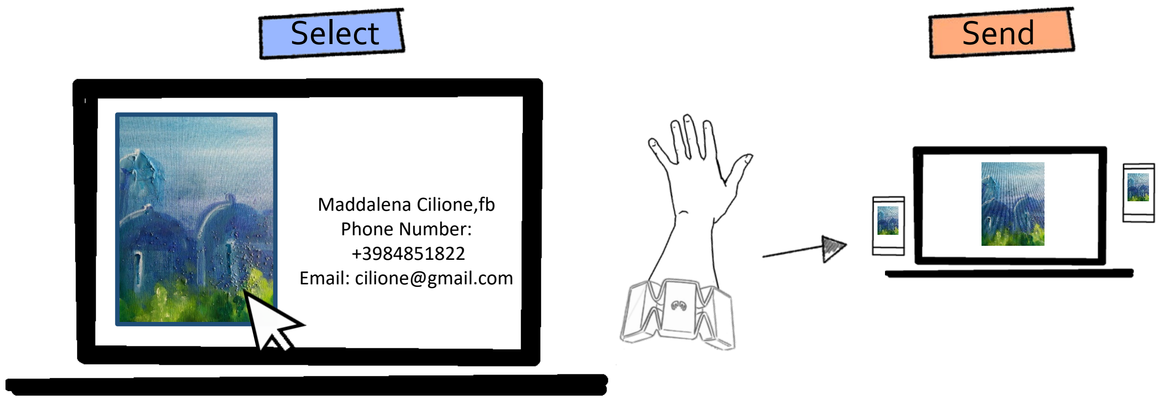


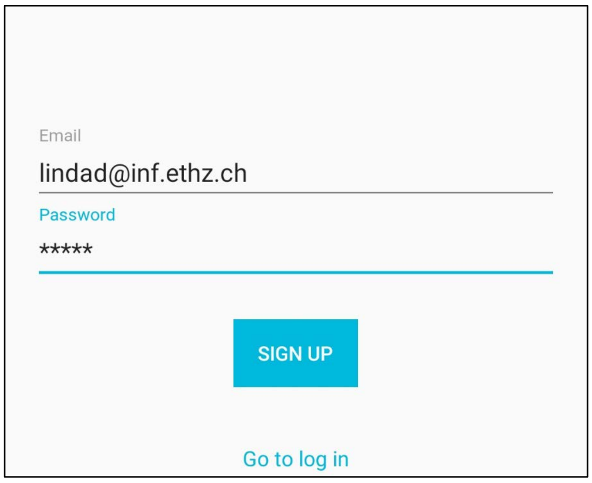
Figure 5.6: Visual representation of the MyoShare system when an image is selected on a laptop and sent to all devices via a finger-spread interaction.

> *Maria selects the pictures she wants to access later, and performs a finger-spread gesture to broadcast the image to all her devices. Finally, Maria wants to contact the artist to ask further information on their work. Maria selects the phone number in the page and performs a wave-in gesture to send this content to her smartphone. At this point, a notification will appear on the mobile phone. Maria clicks on the notification and sees the number shared. By clicking on the number, the system redirects Maria to the phone dialogue of the mobile device.*

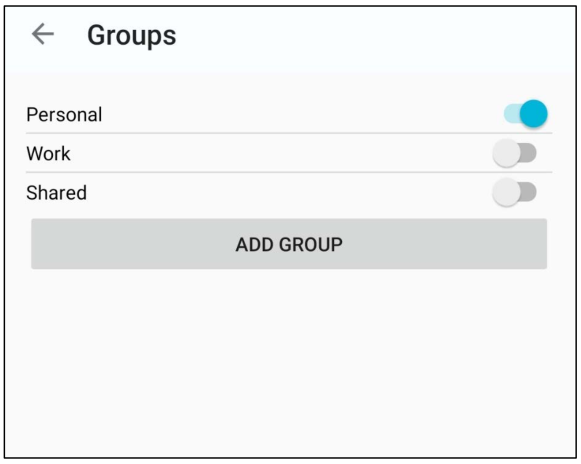
In Figure 5.6, we visually represent some parts of the scenario discussed above. As discussed in Section 5.1.2, currently users often send emails to themselves or send messages to some of their friends on Messager or WhatsApp to copy content. Alternatively, cloud applications are also used to perform such tasks. Although all these methods allow users to access data across different devices, they were not made with these scenarios in mind. For this reason, they require the user to stop their current workflow, copy the desired data, open the messaging app or cloud directory and, finally, paste the content. In contrast with these solutions, MyoShare allows users to send data without the necessity to open other windows or folders. It, therefore, offers an unobtrusive and fast way to share data while minimising possible loss of concentration.

5.2.2 Approach

As introduced at the beginning of this chapter, MyoShare involves the use of a Chrome extension, and an Android and a web application. In this section, we will introduce each of these components from the user perspective, following the interaction flow of the scenario discussed above: users select data from the source device, perform the desired interaction and then access the copied content on the target device. Some of the features offered by MyoShare were informed by the feedback we received from participants of the elicitation study (see Section 5.1.3). We discuss and motivate these functionalities in the next sections.



(a) Login



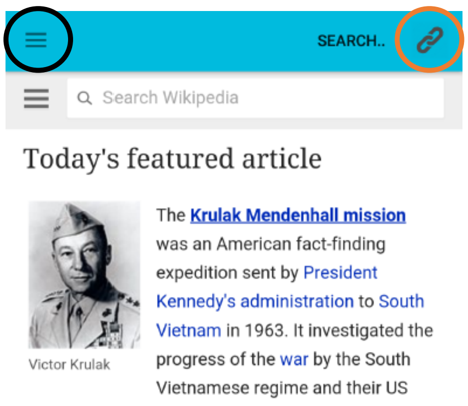
(b) Groups

Figure 5.7: Login phase and group creations on the MyoShare Android application.

Setup and Content Selection

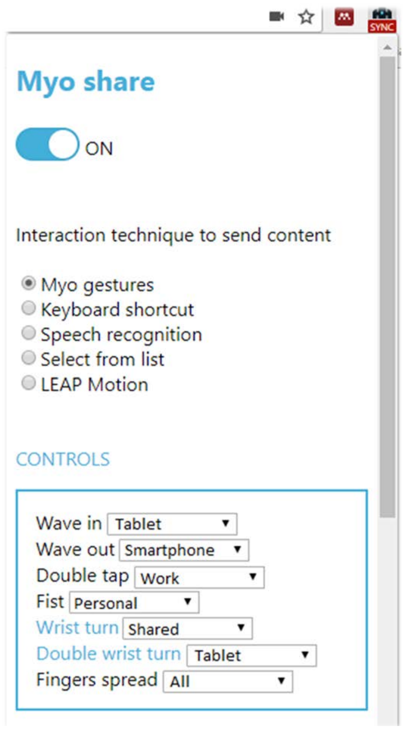
Via a login phase, MyoShare is capable of storing user preferences on the mapping between devices and interactions, as well as all the devices owned and registered by users. When the MyoShare Android app is launched for the first time on a user's phone or tablet, it asks the users to login or to create a new account (see Figure 5.7 (a)). Additionally, the system asks the user to name the current device. For instance, in Figure 5.7 (b), the user names the device "Smartphone" and associates it with the "Personal" category via the setting page of the App. The Android application acts as a regular browser that is extended with additional features of MyoShare. Users can enter URLs or keywords for Google search using the blue address bar on top of the page, while the settings of MyoShare can be accessed by tapping on the menu on the top left of the page (black circle in Figure 5.8). On desktop computers, users need to first add the MyoShare extension and then enter their username and password. Moreover, users have to set ON in the extension (see Figure 5.9).

At this point, users are free to select and send content. MyoShare allows users to



The Krulak Mendenhall mission was an American fact-finding expedition sent by President Kennedy's administration to South Vietnam in 1963. It investigated the progress of the war by the South Vietnamese regime and their US

Figure 5.8: Screenshot of the Android application while browsing a web page.


Myo share

ON

Interaction technique to send content

- [✓ Myo gestures]
- Keyboard shortcut
- Speech recognition
- Select from list
- LEAP Motion

CONTROLS

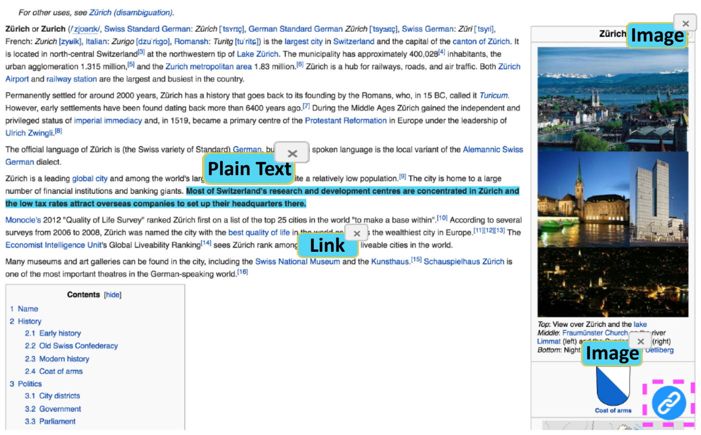
Wave in	Tablet
Wave out	Smartphone
Double tap	Work
Fist	Personal
Wrist turn	Shared
Double wrist turn	Tablet
Fingers spread	All

Figure 5.9: Partial screenshot of the MyoShare Chrome extension.

select plain text, links, pictures, videos and the URL of the currently visited web page. On desktop computers, users can select plain text using click and drag interactions. Once some text is selected, this will be highlighted in blue, and a "plain text" label will appear near to the content (see Figure 5.10). If the selected text contains a phone number, the label will inform the user that the selected text is a phone number. No additional confirmations from the user are needed. A long left click can also select phone numbers. Users can deselect the content by using the ESC key of their keyboard, or by clicking on the "X" button in the label. Similarly, on mobile devices, users can select plain text via tap and drag interactions. Via hold tap gestures, users can select images, videos and links. In contrast with the desktop solution, on the mobile device, no labels are shown to avoid cluttering the screen. A double tap anywhere on the display will deselect any content.

On desktop computers, to select links, images and videos, users are required to perform a long left click on the desired content. While a simple right click could have been a nice alternative for this type of interaction, unfortunately, it is not possible to extend the default behaviour of the browser when the user right clicks in the web page.

As we can see from Figure 5.10, a blue icon in the bottom right of the website is shown to the user. This icon is also available in the Android app, and it is displayed near to the address bar (orange circle in Figure 5.8). This icon allows users to select the current URL of the page. By clicking on the icon, the system will copy the URL in the clipboard.



For other uses, see Zürich (disambiguation).

Zürich or **Zurich** (/ˈzjʊərɪk/, Swiss Standard German: *Zürich* [ˈtsyːrɪç], German Standard German *Zürich* [ˈtsyːʁɪç], Swiss German: *Züri* [ˈtsyri], French: *Zurich* [zyʁik], Italian: *Zurigo* [dzuˈriːgo], Romansh: *Turitg* [tuˈritɕ]) is the largest city in Switzerland and the capital of the canton of Zürich. It is located in north-central Switzerland[3] at the northwestern tip of Lake Zürich. The municipality has approximately 400,028[4] inhabitants, the urban agglomeration 1.315 million,[5] and the Zurich metropolitan area 1.83 million.[6] Zürich is a hub for railways, roads, and air traffic. Both Zürich Airport and railway station are the largest and busiest in the country.

Permanently settled for around 2000 years, Zürich has a history that goes back to its founding by the Romans, who, in 15 BC, called it *Turicum*. However, early settlements have been found dating back more than 6400 years ago.[7] During the Middle Ages Zürich gained the independent and privileged status of imperial immediacy and, in 1519, became a primary centre of the Protestant Reformation in Europe under the leadership of Ulrich Zwingli.[8]

The official language of Zürich is (the Swiss variety of Standard) German, bu spoken language is the local variant of the Alemannic Swiss German dialect.

Zürich is a leading global city and among the world's larg ite a relatively low population.[9] The city is home to a large number of financial institutions and banking giants. **Most of Switzerland's research and development centres are concentrated in Zürich and the low tax rates attract overseas companies to set up their headquarters there.**

Monocle's 2012 "Quality of Life Survey" ranked Zürich first on a list of the top 25 cities in the world "to make a base within".[10] According to several surveys from 2006 to 2008, Zürich was named the city with the best quality of life in s the wealthiest city in Europe.[11][12][13] The Economist Intelligence Unit's Global Liveability Ranking[14] sees Zürich rank among liveable cities in the world.

Many museums and art galleries can be found in the city, including the Swiss National Museum and the Kunsthaus.[15] Schauspielhaus Zürich is one of the most important theatres in the German-speaking world.[16]

Contents [hide]

1 Name
2 History
2.1 Early history
2.2 Old Swiss Confederacy
2.3 Modern history
2.4 Coat of arms
3 Politics
3.1 City districts
3.2 Government
3.3 Parliament

Zürich

Top: View over Zürich and the lake
Middle: Fraumünster Church river
Limmat (left) a (right)
Bottom: Night Uetliberg

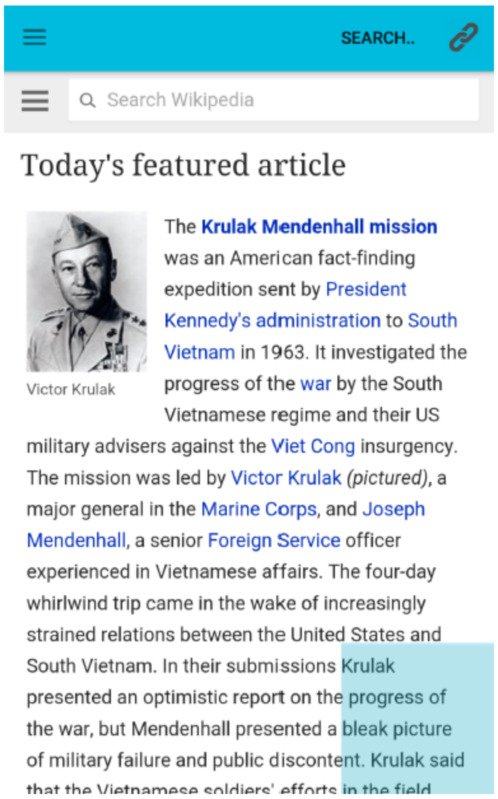
Figure 5.10: Content selection on a desktop computer. Text is zoomed for readability purposes.

Interaction Techniques

Once the desired content is selected, users are free to perform the interaction and copy the data to a device or group of devices. Once the desired content is selected, users are free to perform the interaction and copy the data to a device or group of devices. To map interactions and devices, users can open the Chrome extension on desktop computers by clicking on the Myo icon close to the address bar. MyoShare will show a list of available interaction technique and their mapping to the registered devices of the user (see Figure 5.9). Similarly, in the Android application, these settings are available by going to the settings page. The mapping between gestures and interactions is saved globally, meaning that the association will be used for every registered phone, desktop or tablet. We will now present each available technique. It is important to note that while we have compared mid-air gestures only to a subset of the following interactions, we will present all the studied techniques.

- **Myo Gestures**

  As the name suggests, Myo gestures are mid-air interactions made available via the Myo armband. To use these gestures, users are required to wear the armband and perform the sync gesture of the Myo. The sync gesture is a long wave-out of the user's arm. An icon in the Chrome extension will indicate the state of the Myo ("Off", "Sync", "On"). Similarly, on the Android application, a green (on), red (synching) or grey (off) icon will indicate the state of the armband. As we can see from Figure 5.9, users can associate devices or group of devices to the gestures. Once the gesture is performed, MyoShare will show a label informing the user that the data has been sent and to which device. These labels will be shown for all the interaction techniques. By default, we associate the first registered device or group of devices to the wave-out gesture. This design choice was influenced by the findings of our elicitation study where people preferred directional gestures to target remote devices, and started suggesting out gestures to identify most relevant phones (see Section 5.1.3). For the same reason, we associate the finger-spread interaction to the broadcast action. The other available gestures are wave-in, fist and double tap. All associations of gestures and devices can be changed



Today's featured article

The **Krulak Mendenhall mission** was an American fact-finding expedition sent by President Kennedy's administration to South Vietnam in 1963. It investigated the progress of the war by the South Vietnamese regime and their US military advisers against the Viet Cong insurgency. The mission was led by Victor Krulak *(pictured)*, a major general in the Marine Corps, and Joseph Mendenhall, a senior Foreign Service officer experienced in Vietnamese affairs. The four-day whirlwind trip came in the wake of increasingly strained relations between the United States and South Vietnam. In their submissions Krulak presented an optimistic report on the progress of the war, but Mendenhall presented a bleak picture of military failure and public discontent. Krulak said that the Vietnamese soldiers' efforts in the field

Victor Krulak

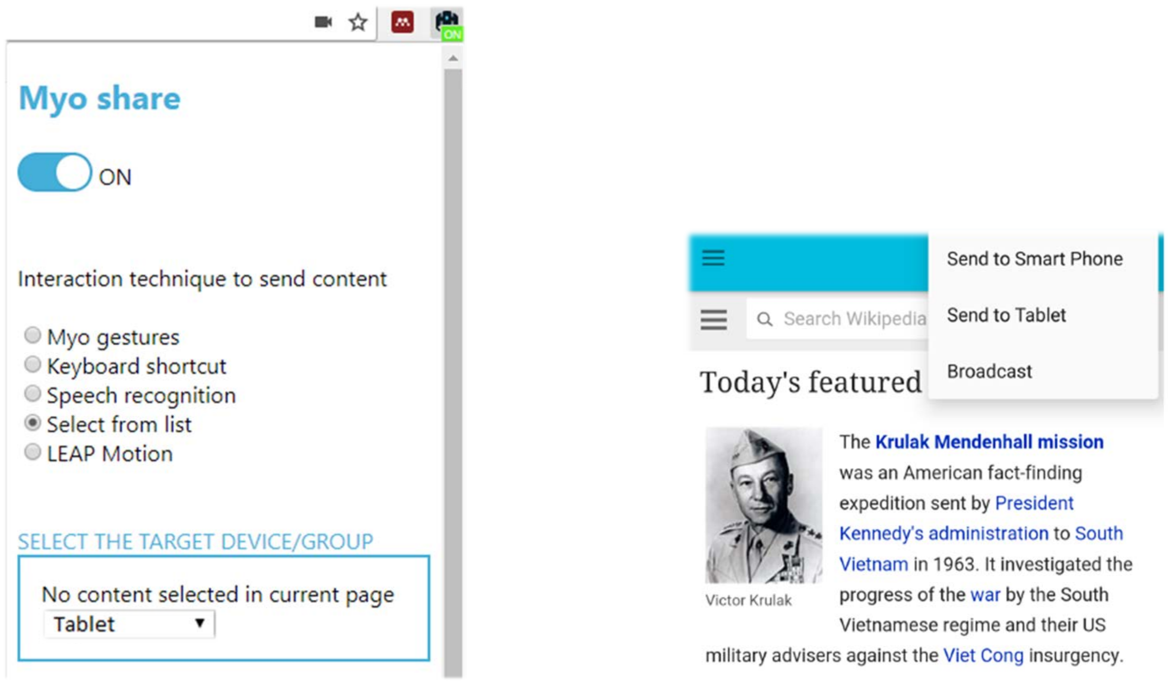
Figure 5.11: Screenshot of MyoShare Android application when tap shortcuts are used as the interaction technique.

via the settings page on the Android application and the Chrome extension on desktop computers.

- **Shortcuts**

  On desktop computers, users can associate devices or group of devices with keyboard shortcuts. For instance, a SHIFT+1 combination of keys could send the selected data to the smartphone of the user and a SHIFT+A (A for all) could share the content to all the registered devices. On the Android applications, we emulate keyboard shortcuts with tap shortcuts. When the user selects this type of interaction technique, a blue rectangle square appears on the bottom right of the page (see Figure 5.11). The number of taps performed will infer the desired target device. For example, one tap could be mapped to the smartphone of the user, two taps to a tablet, and three taps could broadcast the data to all.

  By default, we associate the lower rankings (SHIFT+1, and one tap) to the first registered device. Higher numbers of taps or number on the keyboard are associated with the least important devices. This design choice was also made by taking into considerations user feedback in the elicitation study. By default, we associate the lower rankings (SHIFT+1, and one tap) to the first registered device. Higher numbers of taps or number on the keyboard are associated to least important devices. This design choice was also made by taking into considerations users' feedback in the elicitation study.



Myo share

ON

Interaction technique to send content

- ( ) Myo gestures
- ( ) Keyboard shortcut
- ( ) Speech recognition
- [✓ Select from list]
- ( ) LEAP Motion

SELECT THE TARGET DEVICE/GROUP

No content selected in current page

Tablet ▾

Send to Smart Phone

Send to Tablet

Broadcast

Search Wikipedia

Today's featured

Victor Krulak

The **Krulak Mendenhall mission** was an American fact-finding expedition sent by President Kennedy's administration to South Vietnam in 1963. It investigated the progress of the war by the South Vietnamese regime and their US military advisers against the Viet Cong insurgency.

(a) Chrome Extension (b) Android App

Figure 5.12: Menu selection for MyoShare Chrome extention and Android app.

- **Speech**

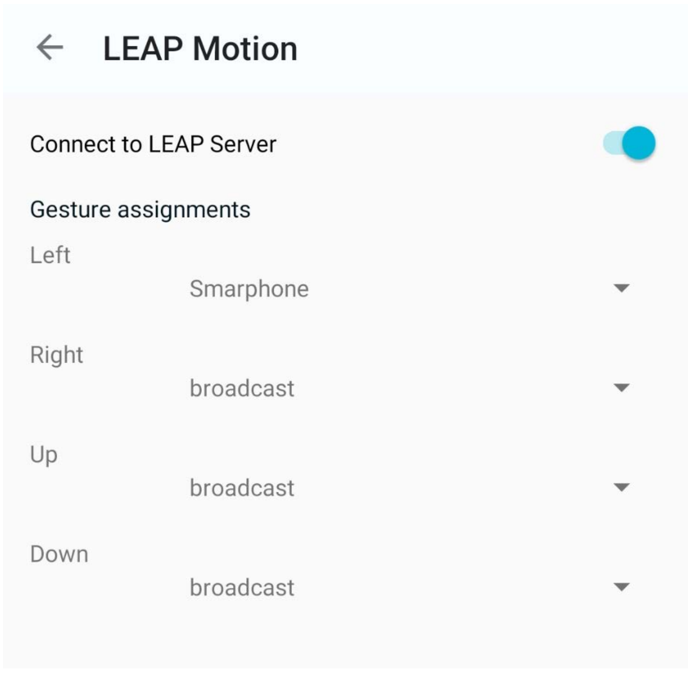
  With the speech interaction technique, we allow users to say a keyword that is mapped to a device or group of devices. For instance, by saying "smartphone" after some content has been selected, MyoShare will send the data to the corresponding device. On desktop computers, when some text is selected, a label on the top right of the page will appear informing the user that the system is "listening". Similarly, on the Android application, a informative label on the bottom of the page will notify the user that MyoShare is waiting for the right word.

- **Menu Selection**

  Users are also free to send data by browsing and selecting devices in a dropdown menu. On desktop machines, the menu is available by opening the Chrome extension on the top right of the page (see Figure 5.12 (a)). On mobile, users can send data to a specific target device by clicking on the menu icon on the top right of the app (see Figure 5.12 (b))

- **Tilting Interactions**

  On the Android applications, users can also send selected text via tilting interactions. MyoShare allows users to associate a device or group of devices to jerk motion gestures as offered by Tilt-and-Tap. For instance, a tilt left gesture could send data to the users' smartphone, and a tilt up could broadcast the selected content to all registered devices. Moreover, tilting gestures can also be combined with touch interactions. In this case, a button will appear at the bottom of the page. Users can perform motion gestures while tapping on this UI element.


LEAP Motion

Connect to LEAP Server

Gesture assignments

Left
Smarphone

Right
broadcast

Up
broadcast

Down
broadcast

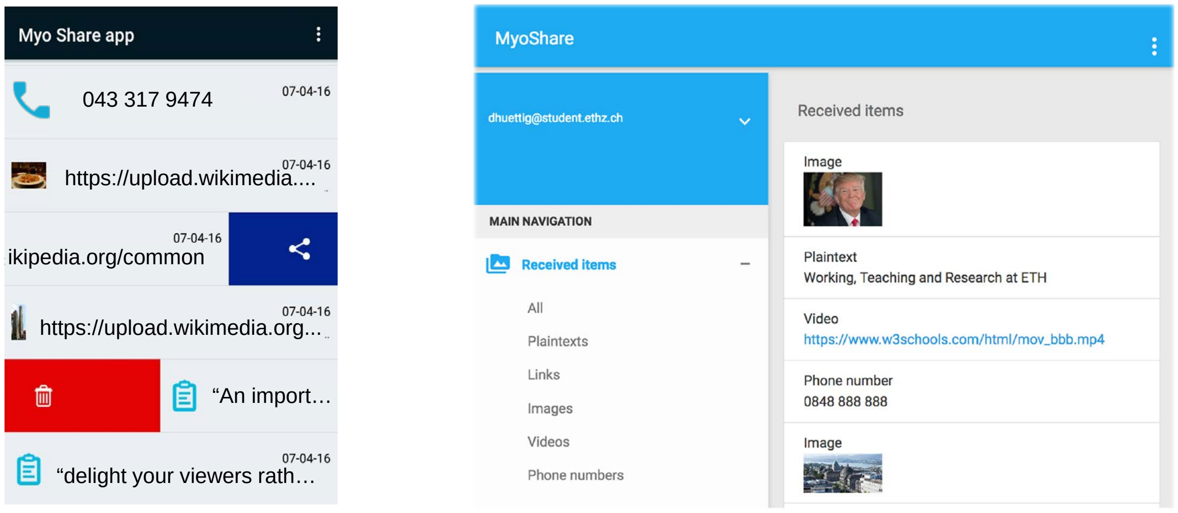
Figure 5.13: Leap toggle and devices mappings on the Android app setting page.

- **Leap Gestures**

  In addition to Myo Gestures, MyoShare also supports mid-air interactions via the Leap motion. Since it is not currently possible to easily connect a mobile phone to this sensor, these interaction techniques were only made available to users in front of a computer with the Leap motion connected to a desktop or laptop machine. In the Chrome extension, users can start using these techniques by simply selecting this interaction type in the pop-up window. On mobile devices, users have first to toggle the "Connect to Leap Server" button in the settings page of the Android device (see Figure 5.13). When the connection has been established, a toast will appear informing the user that they are now free to perform gestures on the Leap motion. The available interactions are wave-in, wave-out, wave-up and wave-down.

Accessing Content on Target Device

Once the data has been selected and sent, users can access it on the target device via the Android app or the web application. On mobile devices, MyoShare pushes notifications as soon as it receives content. By clicking on the notification, the app will open showing the shared data. The different type of data (plain text, video, picture, link, phone number) will be associated with a different icon (see Figure 5.14 (a)). By default, data are ordered depending on when they were shared. Most recent content will be displayed at the top. Each item can be deleted or shared (see Figure 5.14 (a)). As mentioned in Section 5.2.1, if users tap on a phone number, the system will copy it on the phone dialogue directly. Links and plain text can be saved in the clipboard. The web application offers the same features as the Android app (see Figure 5.14 (b)).



Myo Share app

043 317 9474 07-04-16

https://upload.wikimedia.... 07-04-16

ikipedia.org/common 07-04-16

https://upload.wikimedia.org... 07-04-16

“An import...

“delight your viewers rath... 07-04-16

MyoShare

dhuettig@student.ethz.ch

MAIN NAVIGATION

Received items

All

Plaintexts

Links

Images

Videos

Phone numbers

Received items

Image

Plaintext
Working, Teaching and Research at ETH

Video
https://www.w3schools.com/html/mov_bbb.mp4

Phone number
0848 888 888

Image

(a) Android app (b) Web app

Figure 5.14: Screenshots of the Android and web app when accessing received content.

5.2.3 Architecture and Implementation

The main goal of MyoShare was to allow users to share data across devices via a number of different interactions as well as comparing these techniques among each other. However, to reach these goals, MyoShare needed to be composed by a number of different components: an Android application, a web application, a Chrome extension and a remote Node.js server that also exploits a Firebase Cloud Server[4] (see Figure 5.15). While the users is only required to download the Android application and\or the Chrome extension, MyoShare needs two additional servers to detect gestures and keep track of the users preferences and shared data.

More in detail, the cloud server has the responsibility to manage user authentications and to store the data the user has shared. The Chrome extension, the mobile and web application communicate with the Firebase server for authenticating users.

The remote server has three primary roles: it hosts the web application shown in Figure 5.14 (b), it allows the Leap motion to connect to mobile devices via Socket.IO and, finally, it serves push notifications when data are sent to devices.

While the Myo gestures were implemented via the APIs offered by the Myo[5], the Leap motion interactions were developed using thresholds on the position of the users hands on the sensor. Speech detections were made possible via Google APIs[6]. Menu selection and shortcuts were implemented via JavaScript on the Chrome extension and Java on the Android app.

Concerning content detection, to understand the type of data selected we employed a pre-processing algorithm when the page is loaded. In this step, we scan the page to search for phone numbers and remember their location in the DOM. For performance reasons, we keep this data in memory. Every time the user performs a long left click, or hold tap gesture on a mobile device, we run a detection algorithm. This algorithm goes

[4]Firebase website: https://firebase.google.com. Accessed on May 2018.
[5]Myo SDK: https://goo.gl/LyJZQE. Accessed on May 2018.
[6]Google Speech API: https://cloud.google.com/speech-to-text. Accessed on May 2018.

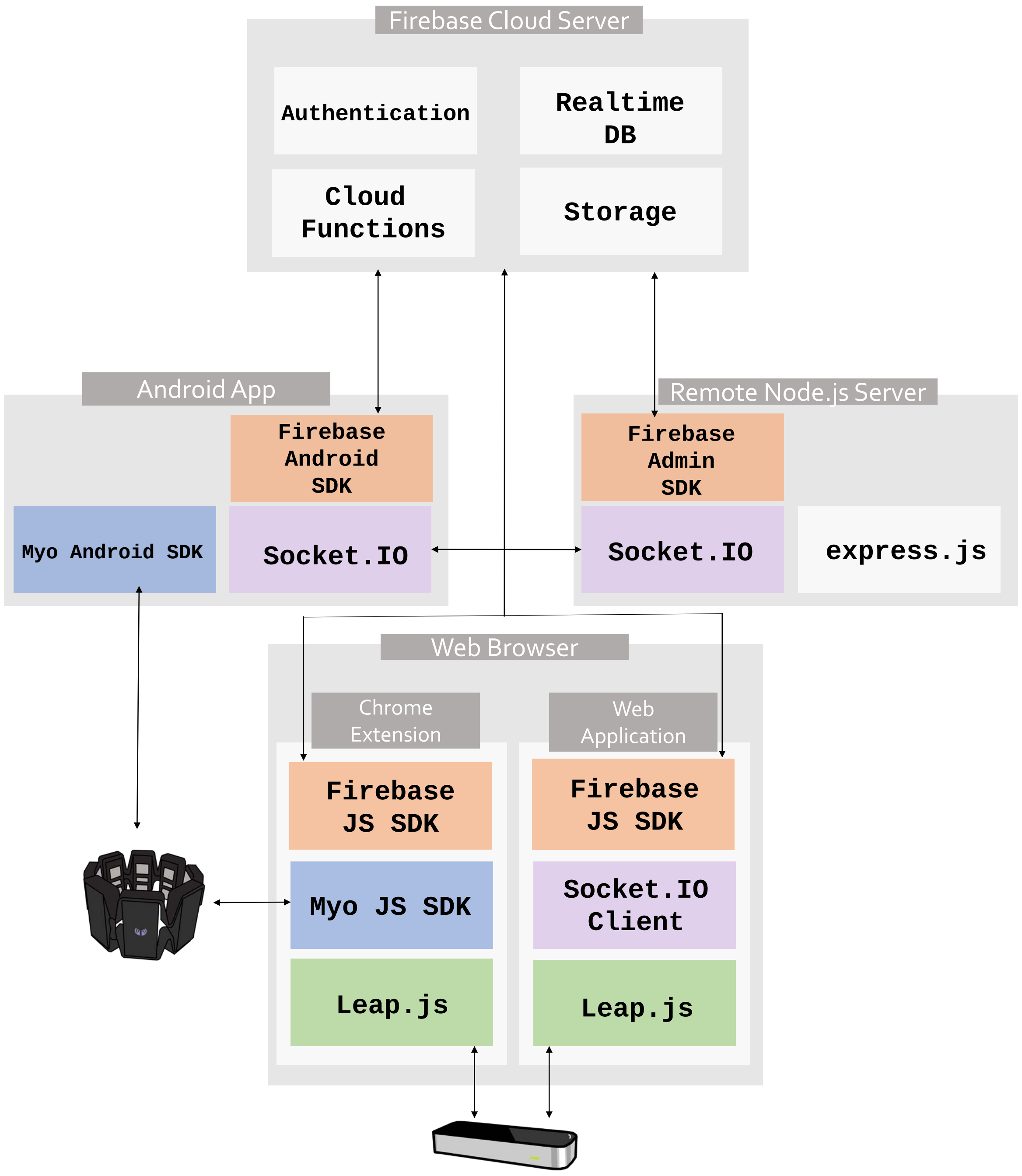
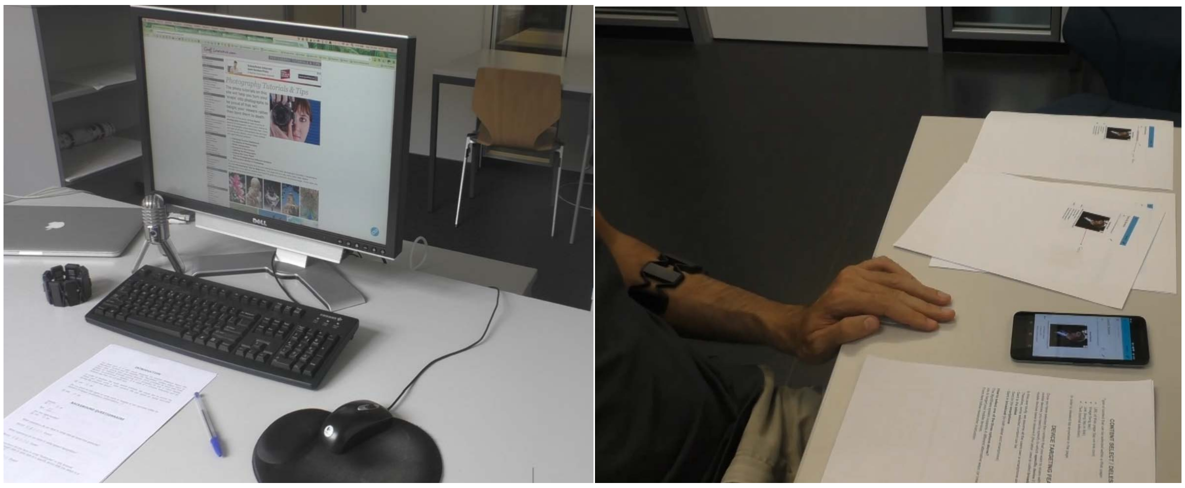


Figure 5.15: Architecture of MyoShare.



(a) First user study (b) Second user study

Figure 5.16: Setup of the two evaluations: when data is sent from a desktop computer (first study) and when data is sent from mobile phones (second study).

through the DOM nodes under the coordinates of the event starting from the element on the top. For every level, we check if the node is a `A`, `IMG`, or `VIDEO` type and if the element contains a previously detected phone number. This check stops as soon as it finds a link, image, video or a phone number. In this way, even if the user selected an element that is nested into other nodes, we are capable of detecting the desired data. Currently, MyoShare is not capable of detecting pictures that are specified in the CSS of the page. For instance, if a `DIV` element has the background-image CSS rule defined in the style of the website, MyoShare will not be able to detect that data. To add this feature, MyoShare should be extended to also scan the CSS files of the web page.

Our content detection solution can be considered an approach of DOM parsing. MyoShare is capable of detecting content without the need of special HTML formats or users input, however, more advanced techniques on Semantic Web could be considered in the future to enlarge the set of content type supported [71, 66].

5.3 Evaluation

To evaluate mid-air gestures in our scenario, we conducted two user studies that compared Myo gestures against shortcuts, menu selection and speech. In the first study, participants were required to copy content from a desktop computer, while in the second evaluation, users shared data from a mobile device. We decided to conduct two separate studies to contain the length of the evaluation and to avoid possible biases. The design and structure of the two evaluations were the same, and we present it in Section 5.3.1. We then report on the findings obtained from the two studies in Section 5.3.2 and discuss the results in Section 5.3.2.

5.3.1 Study Design and Tasks

During the first study, participants were seated close to a desk which was equipped with a computer, a 24” monitor, a keyboard, a mouse and a microphone (see Figure 5.16 (a)). In the second study, users were seated and were asked to use a Nexus 5X smartphone to select data from the MyoShare Android app (see Figure 5.16 (b)). In both studies, a camera placed on the desk recorded the users using our system.

In both studies, users were required to perform a series of tasks using Myo gestures, shortcuts, menu selection and speech. For each technique, we repeated the following structure:

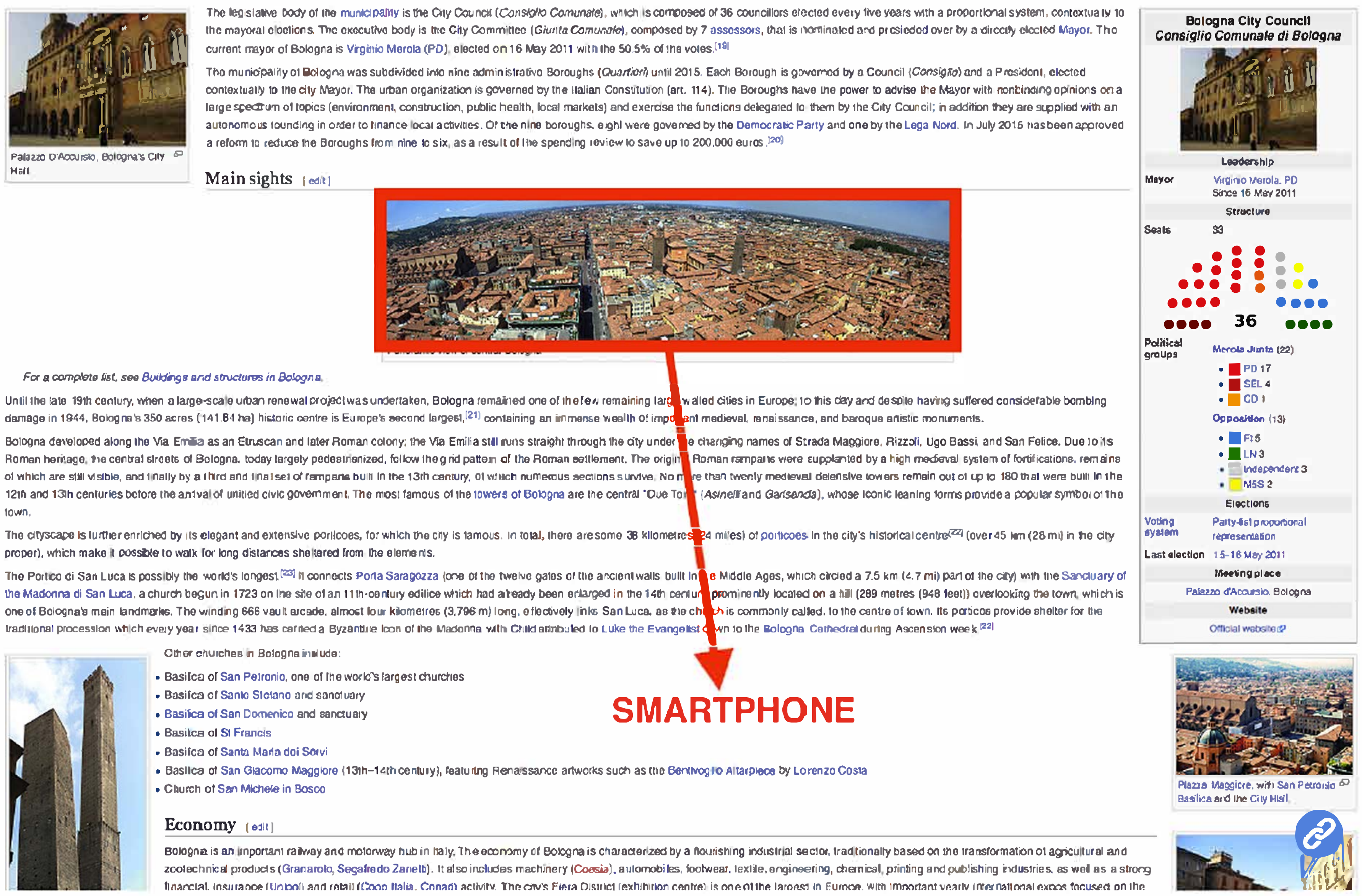
- **Communication phase**: the researcher explained the goal of the next tasks to be performed as well as informing the user about the mapping between the interactions and target devices.
- **Practice phase**: users were required to get accustomed to the interactions by performing a series of tasks on a test web page.
- **Task phase**: We asked participants to perform 12 subtasks. For each subtask, users were required to copy a specific item (current URL, link, image, plain text), in a web page we provided, and perform the interaction to send the data to a target device. A printed sheet informed the user about the item and target to select (see Figure 5.17). Sheets were presented one after the other for each subtask. Users could move to the next subtask only if they had successfully sent the right data to the correct device or set of devices. When errors were performed, participants were required to repeat the subtask.
- **Post-task questionnaire**: users were asked to complete a questionnaire about their rating of the interaction used.

All subtasks were performed on Wikipedia[7] pages for both studies. We picked this website for two main reasons: first, Wikipedia offers the type of media we wanted to tackle (links, images, plain text); second, participants are already accustomed to the web application, and therefore, we could minimise possible biases given by the website in use. To avoid possible learning biases, we shuffled the order of interaction techniques and the order of the devices targeted (smartphone, tablet, broadcast). Moreover, we also changed the Wikipedia web pages for each mode of interaction.

In both evaluations at the end of the study, additional questions were asked to participants in the form of a post-study questionnaire. In the questionnaire, participants were asked to give general comments on our system.

The mapping between interactions and target devices was kept fixed for all participants and all techniques. For Myo gestures, users could target the smartphone via a wave-out interaction, the tablet via a wave-in gesture, and a finger-spread interaction would broadcast the data to both devices. Participants wore the Myo on their non-dominant arm, while their dominant hand was used to select content via a mouse (first study) or the mobile phone (second study). Keyboard shortcuts for the first study were the following: SHIFT+1 to target the smartphone, SHIFT+2 to target the tablet and

[7]Wikipedia website: https://www.wikipedia.org. Accessed on May 2018.



The legislative body of the municipality is the City Council (*Consiglio Comunale*), which is composed of 36 councillors elected every five years with a proportional system, contextually to the mayoral elections. The executive body is the City Committee (*Giunta Comunale*), composed by 7 assessors, that is nominated and presided over by a directly elected Mayor. The current mayor of Bologna is Virginio Merola (PD), elected on 16 May 2011 with the 50.5% of the votes.[19]

The municipality of Bologna was subdivided into nine administrative Boroughs (*Quartieri*) until 2015. Each Borough is governed by a Council (*Consiglio*) and a President, elected contextually to the city Mayor. The urban organization is governed by the Italian Constitution (art. 114). The Boroughs have the power to advise the Mayor with nonbinding opinions on a large spectrum of topics (environment, construction, public health, local markets) and exercise the functions delegated to them by the City Council; in addition they are supplied with an autonomous founding in order to finance local activities. Of the nine boroughs, eight were governed by the Democratic Party and one by the Lega Nord. In July 2015 has been approved a reform to reduce the Boroughs from nine to six, as a result of the spending review to save up to 200.000 euros.[20]

Palazzo D'Accursio, Bologna's City Hall

Bologna City Council
Consiglio Comunale di Bologna

Leadership	
Mayor	Virginio Merola, PD Since 16 May 2011
Structure	
Seats	33
Political groups	Merola Junta (22) • 🟥 PD 17 • 🟥 SEL 4 • 🟧 CD 1 Opposition (13) • 🟦 FI 5 • 🟩 LN 3 • ⬜ Independent 3 • 🟨 M5S 2
Elections	
Voting system	Party-list proportional representation
Last election	15–16 May 2011
Meeting place	
	Palazzo d'Accursio, Bologna
Website	
	Official website

Main sights [edit]

For a complete list, see Buildings and structures in Bologna.

Until the late 19th century, when a large-scale urban renewal project was undertaken, Bologna remained one of the few remaining larg walled cities in Europe; to this day and despite having suffered considerable bombing damage in 1944, Bologna's 350 acres (141.64 ha) historic centre is Europe's second largest,[21] containing an immense wealth of imp ant medieval, renaissance, and baroque artistic monuments.

Bologna developed along the Via Emilia as an Etruscan and later Roman colony; the Via Emilia still runs straight through the city unde e changing names of Strada Maggiore, Rizzoli, Ugo Bassi, and San Felice. Due to its Roman heritage, the central streets of Bologna, today largely pedestrianized, follow the grid pattern of the Roman settlement. The origin Roman ramparts were supplanted by a high medieval system of fortifications, remains of which are still visible, and finally by a third and final set of ramparts built in the 13th century, of which numerous sections survive. No m re than twenty medieval defensive towers remain out of up to 180 that were built in the 12th and 13th centuries before the arrival of unified civic government. The most famous of the towers of Bologna are the central "Due To (*Asinelli* and *Garisenda*), whose iconic leaning forms provide a popular symbol of the town.

The cityscape is further enriched by its elegant and extensive porticoes, for which the city is famous. In total, there are some 38 kilometres 24 miles) of porticoes in the city's historical centre[22] (over 45 km (28 mi) in the city proper), which make it possible to walk for long distances sheltered from the elements.

The Portico di San Luca is possibly the world's longest.[23] It connects Porta Saragozza (one of the twelve gates of the ancient walls built in e Middle Ages, which circled a 7.5 km (4.7 mi) part of the city) with the Sanctuary of the Madonna di San Luca, a church begun in 1723 on the site of an 11th-century edifice which had already been enlarged in the 14th centu prominently located on a hill (289 metres (948 feet)) overlooking the town, which is one of Bologna's main landmarks. The winding 666 vault arcade, almost four kilometres (3,796 m) long, effectively links San Luca, as the ch h is commonly called, to the centre of town. Its porticos provide shelter for the traditional procession which every year since 1433 has carried a Byzantine icon of the Madonna with Child attributed to Luke the Evangelist wn to the Bologna Cathedral during Ascension week.[22]

Other churches in Bologna include:

- Basilica of San Petronio, one of the world's largest churches
- Basilica of Santo Stefano and sanctuary
- Basilica of San Domenico and sanctuary
- Basilica of St Francis
- Basilica of Santa Maria dei Servi
- Basilica of San Giacomo Maggiore (13th–14th century), featuring Renaissance artworks such as the Bentivoglio Altarpiece by Lorenzo Costa
- Church of San Michele in Bosco

Piazza Maggiore, with San Petronio Basilica and the City Hall

Economy [edit]

Bologna is an important railway and motorway hub in Italy. The economy of Bologna is characterized by a flourishing industrial sector, traditionally based on the transformation of agricultural and zootechnical products (Granarolo, Segafredo Zanetti). It also includes machinery (Coesia), automobiles, footwear, textile, engineering, chemical, printing and publishing industries, as well as a strong financial, insurance (Unipol) and retail (Coop Italia, Conad) activity. The city's Fiera District (exhibition centre) is one of the largest in Europe, with important yearly international expos focused on the

Figure 5.17: Example sheet presented to the user for a subtask.

SHIFT+A to send the data to both devices. On the smartphone, one tap on the blue rectangle at the bottom of the page would copy the selected item to the smartphone, two taps to the tablet and three taps to all devices. For the speech mode of interaction, the keywords "smartphone", "tablet" and "all", sent the data to the corresponding devices in both studies. In the first study, we always left the pop-up menu open for the menu selection technique (see Figure 5.12 (a)) to allow a faster selection of the desired target.

We recruited 16 participants for the first study, 3 females and 13 males with an average age of 25.3 (std.: 2.8). For the second study, 13 users participated in the evaluation, 4 females and 9 males, with an average age of 30.9 (std.: 10.8). Since we made use of speech recognition algorithms for one of the modes of interaction studied, we asked users their fluency in the English language. In the first study, two of our participants were native English speakers, while the remaining 14 users stated that they were fluent English speakers. Similarly, in the second study, two of our participants were native English speakers, and the remaining were fluent.

5.3.2 Results and Discussion

We collected data for 12 subtasks for 4 different modes of interactions. This gave us a total of 48 trials per participant. Participants performed at least 758 commands, in the first study, and 624 in the second. We present the results obtained by the two users studies in terms of time and error performances and in terms of qualitative results found from the qualitative survey filled out by participants. We will summarise the results of the two studies together.

Time and Errors

To calculate the performance in terms of timing for each mode of interaction, we studied the performance of the interactions in terms of timing without considering the selection process. For this reason, we kept track of the time interval between the selection of data and when the feedback was shown. In Figure 5.18, we can see the average time in seconds to perform a command for each interaction technique for the first and second study with errors included.

When the source device was a desktop computer, users were faster when using keyboard shortcuts and Myo gestures (ANOVA tests with Greenhouse-Geisser corrections, $F(1.077,16.153) = 17.565$, $p<0.01$). These two interactions were statistically similar in terms of time, and both performed better than speech and menu selection. Selecting the right item in the menu, or saying the keyword for speech recognition, was three times slower than using Myo gestures or keyboard shortcuts.

As found for the first study, directional mid-air gestures and tap shortcuts in the second mobile study were the fastest modes of interactions ($F(1.112,13.340) = 17.565$, $p<0.01$). In both studies, speech was particularly slow. This interaction requires the user to first think the right keyword to pronounce, then pronounce it and wait for the system to detect the keyword and finally inform the user what it understood. This overhead has influenced the performance in terms of timing. While the menu selection technique was slower than the Myo and shortcut interactions in both studies, it did perform better in the second study than the first. This difference can be explained by

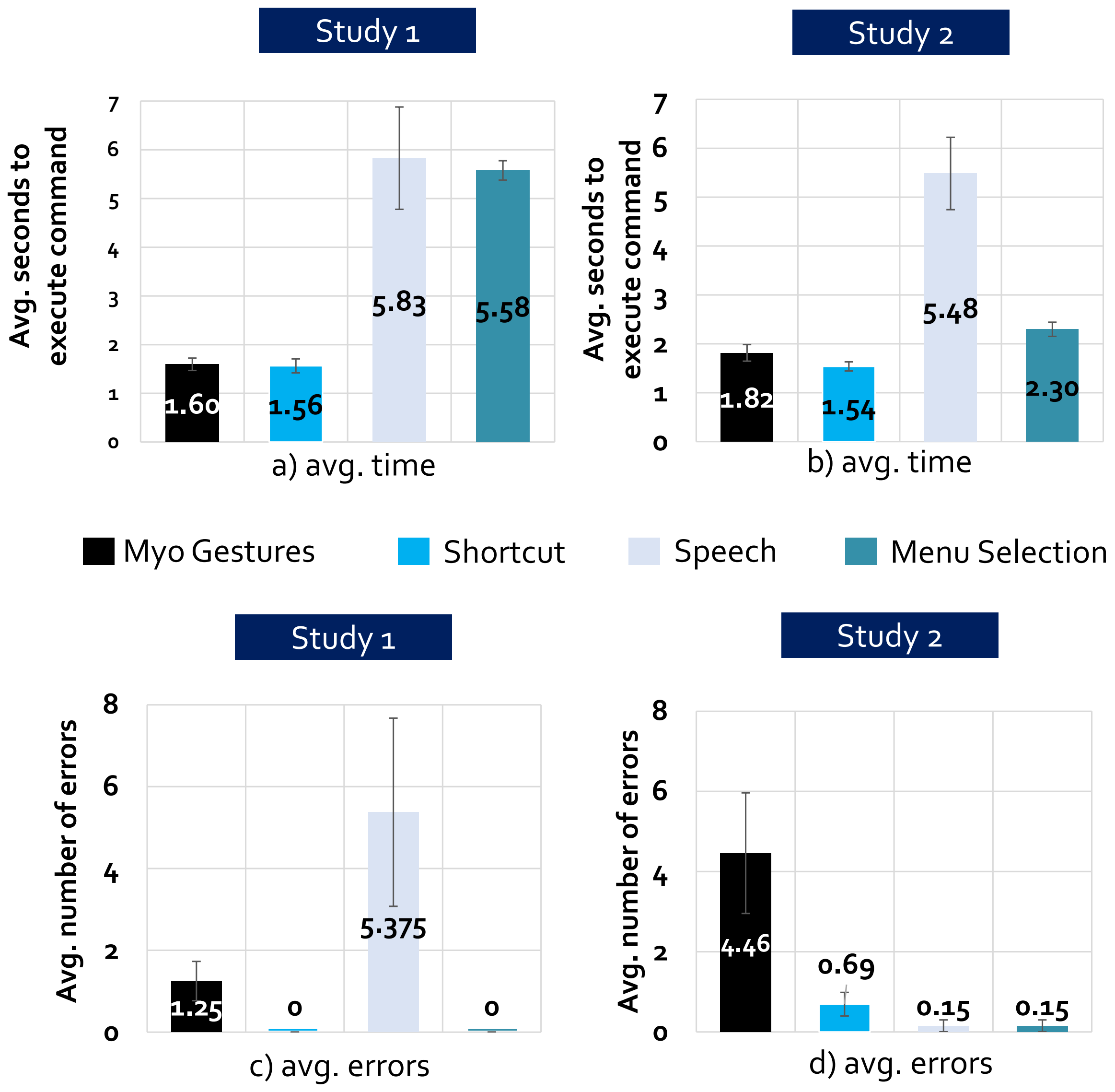


Figure 5.18: Time and error performance for the first and second study. Error bars represent the standard error.

the position of the dropdown menu in the page. On the smartphone, the menu is easier to reach than its counterpart on desktop machines (see Figure 5.12 (a) and Figure 5.12 (b)).

In terms of errors, we kept track of the number of times users had to repeat a command due to either the user performing an incorrect interaction or the system misclassifying the gesture performed or the word pronounced by the user. In Figure 5.18, we can see the average number of errors for the first and second study. Speech and Myo gestures were the interaction techniques with the higher error rates in the first study, while, Myo gestures were more prone to mistakes in the second study. Despite these differences, we did not find any statistically significant differences among interaction techniques for both evaluations with and without including outliers. In the second evaluation, two users performed 800% and 300% more errors than the average users when using Myo gestures. Although the gesture recognition algorithm can be improved by a calibration phase, overall, Myo gestures are not as free from errors as keyboard shortcuts. This factor can be particularly problematic for users with big or small arms. On the other hand, despite errors, Myo gestures were fast interactions, suggesting that users were able to easily overcome errors and repeat the command when the gesture was misclassified by the system.

As shown in Figure 5.18, participants had some issues with the speech mode of interaction in the first study. More in detail, two users had to repeat the same keywords several times until the system finally recognised the word pronounced. While with Myo gestures errors did not influence the time to complete the command in the second study, mistakes had a larger impact for speech in the first study (from 3.3 second without errrors to 5.83 seconds on average with errors)

Qualitative Survey

At the end of the study, for both evaluations, we asked users to rate each interaction techniques in terms of enjoyability, intuitiveness, efficiency, easiness to use and to learn on a scale from one (low rate) to five (high rate). In Figure 5.19, we can see the average responses of participants for each of the four modes of interaction in both studies. The figure also shows the results obtained from Wilcoxon test results. Wilcoxon was executed only if the Friedman tests provided any differences among the interactions for each aspect.

Myo gestures were perceived as enjoyable, efficient and easy to use as keyboard shortcuts when users had to send data from desktop computers. Mid-air interactions were rated as more enjoyable and efficient than speech and menu selection. Moreover, speech was rated as less easy to use than Myo gestures and keyboard shortcuts. This result could have been influenced by the high number of errors performed by participants when using this mode of interaction in the first study.

While we could see more differences among techniques in desktop tasks, this factor is less prominent in the second evaluation. All four interactions performed similarly in terms of enjoyability and easiness to use. Although Myo gestures were rated as less efficient than tap shortcuts, mid-air interactions were perceived as easier to learn than remembering the right amount of taps to target a specific device or group of devices.

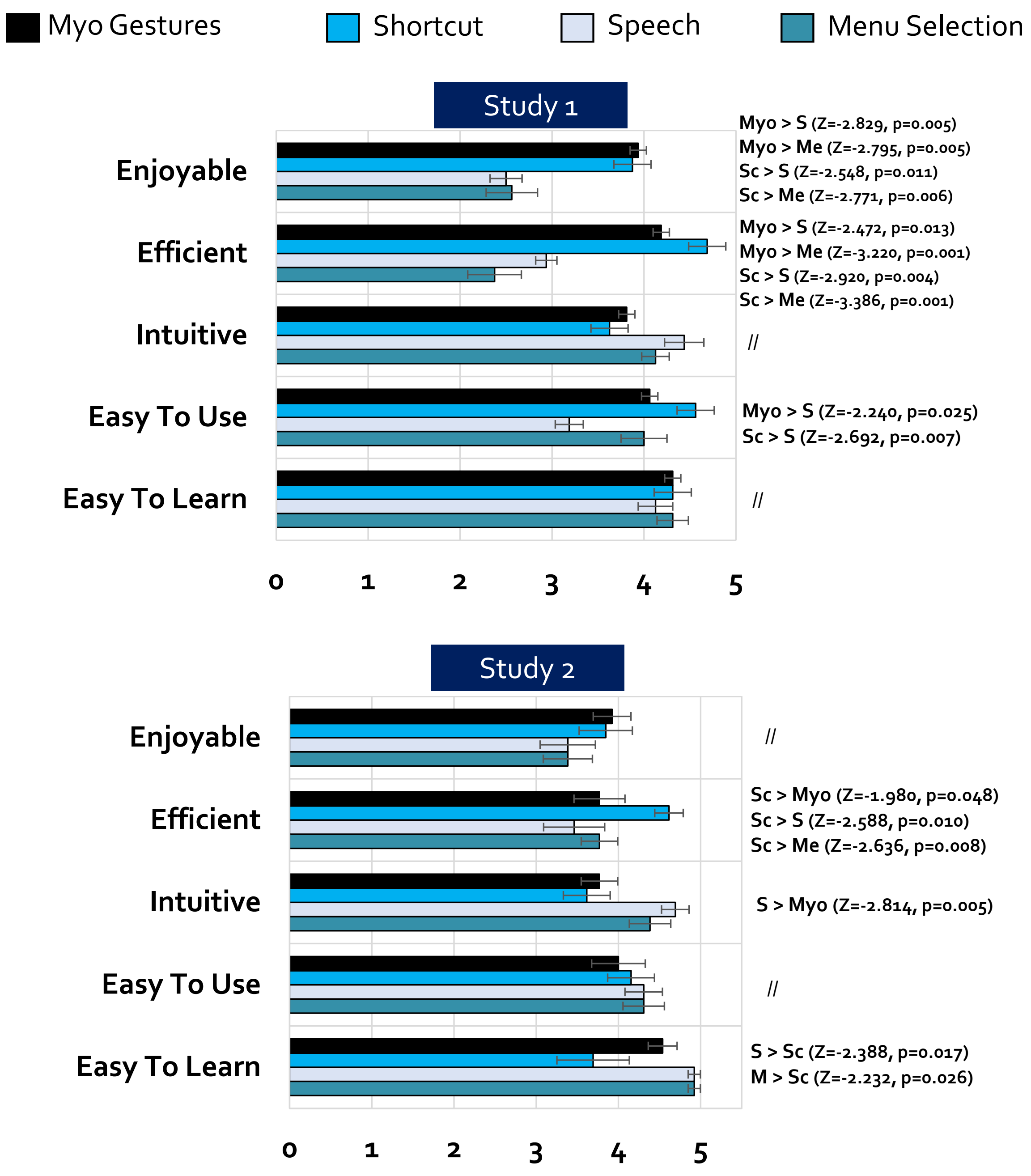


Figure 5.19: Average users' ratings for each interaction technique for the first and second study. The figure also summarise Wilcoxon pairwise results between techniques (Myo) Myo gestures, (Sc) shortcut, (S) speech and (Me) menu selection.

Discussion

In both studies, Myo gestures were as fast as keyboard and tap shortcuts and faster than speech and menu selection. Although we found that Myo interactions can be prone to misclassification, users recovered more easily from errors using mid-air gestures than speech. Moreover, users rated Myo interactions enjoyable and easy to use in both scenarios.

At the end of the study, we asked participants to give their opinions on the wearable device. One participant stated that he would see himself using Myo gestures in any situation if there was no need of additional hardware. Similarly, many other participants discussed the inconvenience of wearing the Myo. While the smart armband gave us the chance to easily experiment our scenario, in the future, alternative and more comfortable devices could improve the user experience. We discuss more on the limitation of the Myo armband and possible future improvements in Section 6.

Since MyoShare allows users to switch between interactions whenever they desire, we asked participants to discuss where, and in which situation, they see themselves using each mode of interaction. Shortcuts and menu selection techniques were rated as the most comfortable interaction technique to perform in public or private places.

Myo gestures and speech were considered more suited to private scenarios, at home or in the office. Participants were concerned about the possibility of performing gestures unfamiliar to other people. However, they also commented that if the mode of interaction became popular, the awkwardness would diminish and, therefore, they would not have any issues in performing mid-air gestures in public places. On the other hand, other users expressed their enthusiasm for using the gestures by saying that they felt futuristic, innovative and cool in doing "*Minority report-style*" gestures to send data across their devices. These results are in line with the discussion made by Rico and Brewster [182]. Although our implementation of speech recognition could be improved by using other more precise APIs, many participants did not like these type of interaction and commented that they do not enjoy using this technique independently by their physical location.

With menu selection, speech and keyboard shortcuts, users could potentially add and associate a large number of devices. On the other hand, tap shortcuts and Myo gestures can support only a limited number of possible devices. While grouping devices into categories can minimise this scalability problem, users of our elicitation study explicitly stated that they like to target specific devices when sending data. However, as found by Hamilton and Wigdor [80], it is hard for users to keep track of more than seven devices at a time. For this reason, users could map interactions to more relevant and used devices, such as a personal phone, or a laptop at home, while other less frequently used devices could receive data only when a broadcast command is executed.

Concerning the content selection on desktop computers, we asked participants of the first study what they thought about the long left click interaction to select links, videos and pictures. Some users reported incoherency between this technique and the single click to select the current URL of the web page (see the blue icon in the bottom right in Figure 5.10). One user suggested supporting both interactions, a single and long left click to select the URL. Lastly, we asked participants what they thought about the organisation of received data on the receiving devices. All users agreed that this was the right technique to access the content sent by other devices.

5.4 Discussion

In this chapter, we presented our work on the use of mid-air interactions to share data across co-located and remote devices. An elicitation study gave us the chance to analyse user needs when passing data across their phones or tablets. Participants also commented and suggested a set of possible gestures that could be used to target devices. Overall, the majority of our users proposed interactions that followed a personal ranking of the devices and directional gestures were the most popular interactions proposed in scenarios with remote devices.

Although the physical position of the target influenced users when we added two devices near to each other, the majority of our participants preferred to use spatially-agnostic techniques to target devices. These users commented that their proposed gesture for remote tasks could also work for co-located scenarios without the need to learn a new additional mid-air interaction. At the end of the elicitation study, all participants commented that they would like to exploit these type of gestures in the scenario envisioned. Moreover, the feedback received from the study informed the design of the MyoShare system.

MyoShare allows users to select and send web content from any page on mobile devices or desktop computers to other devices via mid-air interactions and other techniques, such as menu selections, shortcuts, tilting gestures and speech. Devices can be organised in groups and users are free to customise the mappings between single or sets of devices as well as the interaction techniques.

To evaluate mid-air gestures against other modes of interaction, we carried out two users studies. The first study considered the use of MyoShare from a desktop computer, while in the second study, users shared data from a smartphone. Overall, participants of both studies liked the system and commented positively on the scenarios proposed. Furthermore, Myo gestures performed well concerning timing and qualitative performance. Mid-air interactions were as fast as keyboard shortcuts, and although they were more prone to errors, users easily corrected their mistakes using these gestures.

6 Conclusion

The main goal of this thesis was to improve the mobile web user experience by enriching the number of possible interactions available. While on mobile applications, researchers have already proposed the use of tilting and mid-air gestures as a possible alternatives to touch, on the web, little research has been done on alternative modes of interactions.

In this thesis, we reached the overall goal of enriching the set of possible interactions on the web, via a number of contributions. Moreover, to test and evaluate our approach, we developed frameworks and visual tools that are summarised in Figure 6.1

To better discuss our approach and contributions, let us revisit the research questions from Chapter 1.

- **RQ 1**: *How can we support developers and end-users in building single and cross-device web applications that use tilting interactions?*
- **RQ 2**: *Which single and cross-device applications can benefit from tilting or mid-air gestures?*

To start our research in the area, we decided to tackle the use of tilting interaction on the web. We focused on this type of interaction for their potential advantages in mobile contexts such as their one-hand and eye-free features [15, 217, 205, 67, 221]. For our **first contribution**, we first performed an in-depth analysis of related works in the field, and found a large number of different types of tilting interactions. To address RQ1 for single device cases, we developed Tilt-and-Tap (TAT), a JavaScript framework that offers a set of APIs to implement tilting interactions on the web. TAT was then later extended to TAT 2.0, a framework that supports a more significant number of features. Since the interactions supported by TAT and TAT 2.0 were inspired by related research, the frameworks also offer a catalogue of previously proposed gestures by applying them on the web.

To evaluate TAT, we carried out a developer and a competition study. The developer study had the goal of assessing the usability of the framework and receiving feedback from developers. The competition study aimed at evaluating if the framework could

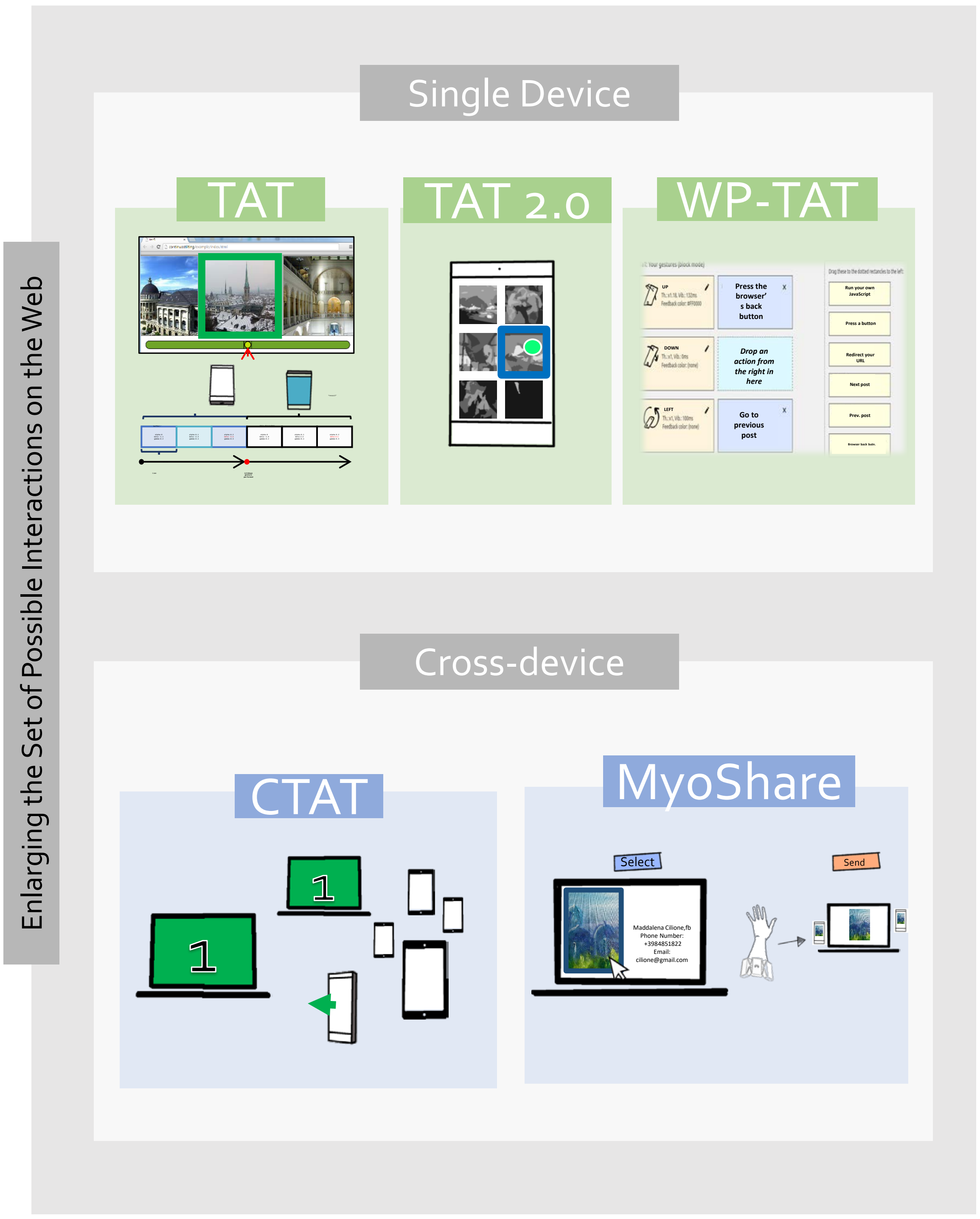


Figure 6.1: Summary of the frameworks and tools presented in this thesis.

improve the creativity of developers. The results of the first study showed that the framework was easy to use and to learn for developers with an average knowledge of CSS, HTML and JS languages. Moreover, all participants were capable of completing the tasks in the allowed time. In the competition study, we received three web applications that, together with TiltZoo and YouTap (see Section 3.1.2), demonstrate the flexibility of the framework.

Although TAT pushes the boundaries of possible interactions on the web, our work is not free from limitations that could be tackled in future research. While TAT and TAT 2.0 are able to minimise the differences among different platforms and devices, browsers continuously evolving and could potentially change their implementation of the motion events in the future. For instance, Chrome has recently decided to modify the `DeviceOrientationChange` and `DeviceMotionChange` events[1], making them incompatible with other browsers. While this cannot be fixed easily until all major browsers follow the standard, a central framework, such as TAT, leave the developers free from the responsibility of checking these inconsistencies. The API will be updated by its authors, and developers simply need to install the new version. However, test the framework(s) on different platforms and different devices over time can be a challenging task. For this reason, TAT and its extension are open source and they are available on GitHub[2] allowing the community to report and eventually fix issues that may arise in the future.

We note that differences among browsers are also given by the lack of applications that use motion gestures. We believe that browsers will more strictly follow standards if more applications would exploit tilting interactions in the future.

With TAT and TAT 2.0 we wanted to target web developers and encourage them to experiment with alternative interaction techniques in their web applications while categorising and supporting a broad set of tilting interactions proposed by previous research. In contrast, with our **second contribution**, we tackle RQ1 for end-user cases. We reached this goal by applying the use of the drag and drop paradigm to add motion interactions to web applications. To test our approach, we developed WP-TAT, a WordPress extension for the rapid prototyping of tilting gestures on the web. While WordPress enables users to easily pick a theme of their choice and add content, the CMS does not support any customisation of the interaction techniques used by the web application. WP-TAT bridges this gap and offers users the possibility to extend the set of interactions in their web application via a visual interface.

With WP-TAT, users can add global actions to tilting gestures via a visual interface. We defined a global action as an action that affects the entire website. For instance, users visiting the website could go to the next or previous posts by tilting the device to the right or to the left, respectively. Similarly, users could search on Google for some text displayed on the website by selecting it and tilting the device up. We again developed a number of web applications to show use cases for WP-TAT. A preliminary user study was conducted to compare two variants of our WordPress extension, a classic visual interface where buttons and pop-up menus allow users to associate global actions to tilting gestures, and a block UI, where users can drag and drop actions into motion gestures. Overall, participants preferred the drag and drop approach, however, one user

[1]Motion events changes for Chrome: https://goo.gl/8g2cNU. Accessed on June 2018.

[2]TAT GitHub page: https://github.com/lindig11/tiltandtap. Accessed on September 2018.

also enjoyed using the classic UI.

While the set of proposed global actions demonstrated various use cases of tilting gestures on the web, currently, WP-TAT only exploits jerk tilting interactions. In the future, continuous tilting gestures could also be integrated in the tool. In this scenario, alternative actions should be applied to such interactions. For instance, continuous tilting gestures could be suited for scrolling tasks on gallery of images and text.

With our **third contribution** we answer RQ2, by listing a series of design guidelines on the use of motion interactions on the web. Among all the design observations discussed in this thesis, we report that tilting gestures can make websites more interactive and fun to use.

We found that tilting interactions can be used in scenarios where solo-touch gestures may fail (e.g. small buttons). However, if not informed, users might be unaware of the presence of motion gestures. Similarly, on many mobile devices, such as iPhones and iPads, a number of multi-touch gestures are available; however, users are often not aware of these possibilities. Although some of these interactions might be extremely useful, they are often referred as hidden or secret gestures[3]. In our work, we propose the use of buttons and labels displayed on web pages as possible cues to inform the user of the presence of tilting interactions. Although these methods might gather the attention of the user, we did not study how efficient these techniques are. In the future, a user study could evaluate how fast users are in detecting tilting gestures on web applications and which method is best. Moreover, consistency among different web applications could improve the memorability of the gestures. In our example applications, we mapped tilting interactions to different actions arising consistency issues. For instance, a tilt down closed the cookie information label on one website, while the same interaction was used to search for text on Google in another web application. Possible future work could involve an analysis of the mapping between specific motion gestures and actions. Furthermore, if tilting interactions become in common use, browsers might support them natively giving a consistent user experience among different websites.

We continued our research on the use of alternative gestures on the web, by studying tilting interactions in cross-device scenarios. For our **fourth contribution**, we designed a series of requirements to improve the development process as well as the user experience on the use of motion gestures in cross-device applications. With this contribution, we tackle RQ1 for cross-device scenarios. To reach this goal, we employed the use of TAT with Socket.IO to implement a simple application to remotely control a cursor on a big screen via means of motion gestures on a mobile device. We then used this application to compare motion gestures to touch interactions via a preliminary user study. While we did not notice significant differences among touch and tilting gestures in terms of timing, we found that motion interactions allowed users to focus on the big screen while controlling the indicator on the phone or the tablet.

Inferred by the results of the study, as well as the experience gathered during the development of the application, we designed and implemented CTAT, a cross-device framework for the rapid development of tilting gestures on cross-device applications. With CTAT, developers can define a sender of the interaction, the receiver and the type of desired gesture via JavaScript, without the need to manage the detection of the interaction, recognising the right devices involved in the communication and deal with

[3]Example article on secret gestures: https://goo.gl/tPWBJw. Accessed on June 2018.

the server. Moreover, a visual interface, VCTAT, allows developers to create tilt links among devices without the necessity to write any code.

With the combination of CTAT and TAT 2.0, we reach our **fifth contribution** by comparing different variants of continuous tilting interactions (slow movements of the device in some direction) in cross-device scenarios. While previous research has already studied these gestures in single device use cases [162, 205], via a detailed user study, we evaluated the performance of these interactions when they are used to control a cursor on a larger display. This contribution tackled RQ2 for cross-devices use cases.

Concerning CTAT, we allow developers to associate to each tilting gesture a sender and a receiver; however, the framework is not aware of the physical position of the devices. Intuitively, a tilt left interaction could trigger actions only on the targets that are located on the left. While this spatial awareness feature might be particularly interesting for tilting gestures, being aware of the position of the devices requires additional hardware, such as Kinect cameras, and a more challenging architecture [176] since the Doppler effect, or similar approaches may be insufficient when devices can be located anywhere in a room. CTAT aimed at helping developers to easily exploit motion interactions on cross-device applications without the need of a complex setup. For this reason, we did not consider the physical location of the devices when users perform gestures. However, researchers are now refining systems that are able to detect the position of devices when indoors without the need of additional hardware installed in the room [127]. In the future, when the position of devices can be detected reliably, the concepts of CTAT could be extended to also involve this new data.

Moreover, while TAT 2.0 supports a number of alternative continuous interaction techniques, we only compared a subset of these variants in our cross-device user study. For time constraints, we could not evaluate all the supported implementations, and we picked the continuous interactions that were previously studied in single device scenarios. In the future, a user study could compare a more broad set of continuous motion gestures also in 1D scenarios. Furthermore, an in-the-wild study on aCrossETH could give interesting insights on how tilting interactions could improve user engagement with public screens.

We continued our research in cross-device interactions by exploring the use of mid-air gestures. In an elicitation study, we wanted to understand what sets of gestures users would naturally produce to send data across devices if the targets were remote or co-located. We found that users have a ranking of their own devices and associate gestures to the targets depending on their importance. Moreover, when more devices are near to each other, users preferred to use spatially-agnostic gestures despite their vicinity to the target. Overall, the set of gestures suggested by users, as well as the reasoning behind these interactions, represent the **sixth contribution** of this thesis.

The elicitation study informed the design of MyoShare, a system that allows users to select web data from any sites and perform mid-air interactions to copy content to other devices. For our **seventh and last contribution**, mid-air gestures were later compared to more common interaction techniques via two user studies. In the first evaluation, we studied mid-air gestures when data were sent from desktop computers and compared them to keyboard shortcuts, speech and menu selection interaction techniques. In contrast, in the second study, we evaluated the use of mid-air gestures to copy data from a mobile device and, similarly to the first study, we compared them to tap shortcuts, speech and menu selection. Overall, mid-air gestures were as fast as keyboard shortcuts, and participants enjoyed using this technique and felt that they would like to use MyoShare in the future. Although mid-air gestures were more prone to errors, we found that users easily recover from their mistakes. With these last two contributions, we tackled RQ2 and used mid-air gestures to enlarge the set of possible interactions on the web.

With MyoShare we question the importance of the devices' physical location when sharing data via mid-air interactions. While our elicitation study proved that spatially-aware gestures were not always preferred by our participants, we did not implement all possible interactions suggested by users but focused only on the most popular ones. In remote scenarios, directional gestures were the favourite type of gestures; however, participants proposed different variants of these mid-air interactions. MyoShare does not support all these alternatives and only allows waving gestures. In the future, circular, pose, sequential, as well as tilting gestures and all variants of directional interactions, could be compared to further study how they also perform in terms of memorability.

Concerning the use of the Myo armband, we recognise that the wearable device might be cumbersome to use. Some participants of our user studies felt that the Myo was uncomfortable to wear because it felt too tight. In contrast with smartwatches, the Myo is placed in an unfamiliar position of the users' arm, its form factor is not *fashionable*, and it does not allow users to wear shirts with long sleeves since the device has to be in direct contact with their skin. Moreover, the recognition of the gesture may be faulty for users with small or big arms. However, the Myo allowed us to easily experiment with gestures that cannot be currently detected with smartwatches. In the future, alternative wearable devices as well as more advanced technologies [77] for detecting mid-air gestures could be explored in the future to improve the overall user satisfaction.

Overall, while our approaches push the boundaries of the interactions available on single and cross-device web applications, we believe that future research in the field could tackle the limitations discussed in this section as well as expand the findings of this thesis.

Acknowledgements

I am currently writing the conclusion of my thesis and I thought that writing the acknowledgement would be a nice excuse to take a break.

As far as my english allows me, I'll try to express my feelings for colleagues, family and friends that helped me reach this goal. However, it is very hard to summarise almost five years of projects, trials and errors, advices and coding nights. I'll try to start from the easy part. Professor Moira Norrie, was not only my supervisor but a real mentor. She guided me on the hard roads, and pushed me just enough to let me feel independent on the professional and personal level. Although I still play Pokmon at times, I believe that I started my PhD as a young adult, and ended it as a woman (still nerdy, but it is too late to fix that). I know that this is true also because of her. For these reasons, thank you Moira, thank you a lot.

Special thanks also go to all my colleagues that shared with me some of these years and helped me go trough this long path. Among all, I think that Matthias and Maria were the ones that had to deal with my temper the most. I know there is a *cultural barrier*, so thank you for dealing with this random south italian girl one time too often. Thank you Tillman, David and Karl for the nice talks and the occasional beers after work. I wish I had more of those, I hope there is still time to atone. Thank you Christoph, Fabrice and Michael for the long chats and the nice discussions we had. A big thanks also to my co-examinares for the time and work they had to spend on my thesis, it is really appreciated.

If I had this opportunity was also thanks to my home university, and especially to Professor Filomena Ferrucci and Professor Federica Sarro. Thank you for believing in me since the beginning.

While working on papers and dealing with reviewers is not easy, bureaucracy is not a joke either. However, Beate, Claudia and Denise helped me trough the jungle of administrative stuff, with always a smile and a nice word for me. Thank you all.

Leaving the food... I mean Italy was not easy. I was afraid that with cheap and great pizzas I would have lost also some friends on the way. Sadly this was partially true. There are some relationships that need physical vicinity to last, others instead can get stronger with time and adversities such as nostalgia and absence. Mamma, Enzo, Papa', Monica, Carolina, Alessia, Sandra, Felice, Antonella, Zio Walter, Zia Annalisa, Zia Mariagrazia, Zia Silvana, Nonna Linda, Nonna Maddalena, Federica, Giulia, Michele, Giovanni, Chiara, Denise, Giulio, Jessica, Fabrizio, Ele, Stani, Angela, Luca, Claudio and Adriano thank you for beeing in the second category.

Switzerland also gave me the change to make new friends in strange ways (like meeting in public bathrooms, eh Eva?), to the point that one starts questioning the existence of fate. Many thanks to Elina, Lorenzo and, of course, Chanel, for the laughs and chats. Speaking of fate, I met Fabio Sorrentino four years ago, and we were

devastated when he left Switzerland, however, he had another gift to give us, it was you Martina. Thank you for being here, I will always be there for you.

"We started this thing together, we end this together". Thank you Alfonso for saying these words also when it felt as the hardest thing to do.

Ringraziamenti

È il terzultimo giorno qui all' ETH e per quanto sia felice di iniziare un nuovo percorso, provo nostalgia nel ricordare questi ultimi anni. Partire non è stato semplice, in realtà, non scappavo da nulla. Non aspiravo a cambiare mondo, perchè il mio mondo non aveva nulla di sbagliato. Cinque anni fa, ero solo in cerca di qualcosa, non sapevo neanche di preciso cosa. Indipendenza, certamente, lavoro, senza ombra di dubbio, ma, amore? No, quello non mi è mai mancato. Per quanto spesso ho la sensazione di non meritarlo, capisco che in qualche maniera sono in grado di ricambiare l'affetto che ricevo, non so bene come, ma se così non fosse penso non sarei circondata da persone splendide come quelle che sto per citare.

Grazie mamma, grazie Enzo, per avermi insegnato cosa significa vivere, e farlo al meglio delle nostre possibilitá. Grazie anche a Nikla, per aver vissuto la sua vita con noi, avrai sempre un posto speciale nel mio cuore.

Grazie papà, grazie Monica, Carolina ed Alessia. Zurigo mi ha portata un poó piú vicino a voi, e anche se il tempo insieme non basta mai, sapete che ci saró sempre. Vi voglio bene, davvero. Grazie Sandra, Felice, Antonella, la banda Murolo e Zio Mimmo, per far parte della mia famiglia allargata. Grazie di tutto.

Grazie Zio Walter e Zia Annalisa, per avermi insegnato cos'è la leggerezza. Mi mancate ogni giorno di piú. Grazie Nonne per l'infanzia fantastica che siete riuscite a donarmi. Grazie Giulia, Michele e Giovanni, Zia Mariagrazia, Zia Silvana e Federica per aver sempre creduto in me. Grazie anche se riusciamo a viverci con il contagocce.

Grazie alle amiche di sempre. Quelle che subiscono ore di messaggi vocali, e li ascoltano anche. Quelle che ci sono sempre state e sempre ci saranno. Denise e Chiara, siete parte di me. Grazie Dario per tutti i ricordi. Grazie per aver reso le superiori e l'università un posto vivibile.

Se mi vedete a volte assente quando sono con voi è perché spesso vorrei fermare il tempo. Fermarlo durante una partita a Mice e vivere fino infondo l'unione speciale che ci lega. Fabrizio, Jessica, grazie di esistere.

Infine, Nonno Bruno, Nonno Giulio e Paolo, questa tesi è dedicata a voi. Grazie per avermi insegnato a non dare mai nulla per scontanto.

P.S.: Scrivere doppi ringraziamenti non è stato semplice. Ci sono sentimenti per alcune persone che posso esprimere solo in italiano, altre solo in inglese. Ho deciso quindi di fare di tutto un poò e dividerli in doppia lingua. Se stai leggendo questi ringraziamenti, probabilmente il tuo nome è in uno di questi.

A
Student Contributions

A number of students have contributed to the technical realisation of the tools and showcase applications presented in this thesis. The following students have made a significant contribution as part of their Master's, Bachelor's or lab project. All students have been personally supervised by this thesis' author.

- **Ersan aras** has implemented the second prototype of Tilt-and-Tap [7].
- **Andreas Blchliger** has developed the 3DTiltGallery application [22].
- **Andrea Canonica** has developed the first prototype of Tilt-and-Tap 2.0 [29].
- **Sandro Kalbermatter** has experimented with tilting interactions on WordPress to build WP-TAT [109].
- **Can Türk** has implemented TiltGallery and carried out the associated preliminary user study [207].
- **Abhimanyu Patel** has implemented the first prototype Cross-Tilt-and-Tap [168].
- **Marica Bertarini** has implemented the first version of MyoShare for desktop computer and carried out the associated elicitation and user study [21].
- **Julia Badertscher** has developed the first version of MyoShare for mobile and carried out the associated user study [14].
- **Dirk Hüttig** has refined the implementation and architecture of MyoShare [103].

Bibliography

[1] R. Acerbis, A. Bongio, M. Brambilla, S. Butti, S. Ceri, and P. Fraternali. Web applications design and development with webml and webratio 5.0. In R. F. Paige and B. Meyer, editors, *Objects, Components, Models and Patterns*, pages 392–411, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-69824-1. Cited on page 20.

[2] S. Aghaee, C. Pautasso, and A. D. Angeli. Natural end-user development of web mashups. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 111–118, Sept 2013. doi: 10.1109/VLHCC.2013.6645253. Cited on page 21.

[3] R. Aigner, D. Wigdor, H. Benko, M. Haller, D. Lindbauer, A. Ion, S. Zhao, and J. Koh. Understanding mid-air hand gestures: A study of human preferences in usage of gesture types for hci. *Microsoft Research TechReport MSR-TR-2012-111*, 2, 2012. Cited on page 30.

[4] A. Alapetite. Dynamic 2d-barcodes for multi-device web session migration including mobile phones. *Personal and Ubiquitous Computing*, 14(1):45–52, Jan 2010. ISSN 1617-4917. doi: 10.1007/s00779-009-0228-5. URL `https://doi.org/10.1007/s00779-009-0228-5`. Cited on page 40.

[5] F. Alt, A. S. Shirazi, T. Kubitza, and A. Schmidt. Interaction techniques for creating and exchanging content with public displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1709–1718, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466226. URL `http://doi.acm.org/10.1145/2470654.2466226`. Cited on page 37.

[6] H. Anzures and S. Mendoza. Multi-user interaction with public screens using mobile devices. In *2011 8th International Conference on Electrical Engineering, Computing Science and Automatic Control*, pages 1–5, Oct 2011. doi: 10.1109/ICEEE.2011.6106647. Cited on page 37.

[7] E. Aras. Development framework for tilt-and-tap interaction with web-based applications. Master's thesis, ETH Zurich, 2015. Cited on page 145.

[8] C. Ardito, M. F. Costabile, and H.-C. Jetter. Gestures that people can understand and use. *Journal of Visual Languages & Computing*, 25(5):572 – 576, 2014. ISSN 1045-926X. doi: https://doi.org/10.1016/j.jvlc.2014.07.002. URL `http://www.sciencedirect.com/science/article/pii/S1045926X14000639`. Cited on page 30.

[9] S. S. Arefin Shimon, C. Lutton, Z. Xu, S. Morrison-Smith, C. Boucher, and J. Ruiz. Exploring non-touchscreen gestures for smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3822–3833, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858385. URL `http://doi.acm.org/10.1145/2858036.2858385`. Cited on page 2.

[10] I. Aslan, A. Krischkowsky, A. Meschtscherjakov, M. Wuchse, and M. Tscheligi. A leap for touch: Proximity sensitive touch targets in cars. In *Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '15, pages 39–46, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3736-6. doi: 10.1145/2799250.2799273. URL `http://doi.acm.org/10.1145/2799250.2799273`. Cited on pages 31 and 32.

[11] A. Attenberger and K. Buchenrieder. Remotehand: A wireless myoelectric interface. In M. Kurosu, editor, *Human-Computer Interaction. Advanced Interaction Modalities and Techniques*, pages 3–11, Cham, 2014. Springer International Publishing. ISBN 978-3-319-07230-2. Cited on page 34.

[12] M. T. I. Aumi, S. Gupta, M. Goel, E. Larson, and S. Patel. Doplink: Using the doppler effect for multi-device interaction. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 583–586, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1770-2. doi: 10.1145/2493432.2493515. URL `http://doi.acm.org/10.1145/2493432.2493515`. Cited on pages 9 and 40.

[13] S. K. Badam and N. Elmqvist. Polychrome: A cross-device framework for collaborative web visualization. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 109–118, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2587-5. doi: 10.1145/2669485.2669518. URL `http://doi.acm.org/10.1145/2669485.2669518`. Cited on pages 5, 13, and 20.

[14] J. A. Badertscher. Mobile-myoshare: Exploiting hand-free gestures to share data among devices. Bachelor's thesis, ETH Zurich, 2016. Cited on page 145.

[15] M. Baglioni, E. Lecolinet, and Y. Guiard. Jerktilts: Using accelerometers for eight-choice selection on mobile devices. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, ICMI '11, pages 121–128, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0641-6. doi: 10.1145/2070481.2070503. URL `http://doi.acm.org/10.1145/2070481.2070503`. Cited on pages 2, 25, 26, 42, 46, 48, 49, 81, and 135.

[16] M. Baldauf, F. Adegeye, F. Alt, and J. Harms. Your browser is the controller: Advanced web-based smartphone remote controls for public screens. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays*, PerDis '16, pages 175–181, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4366-4. doi: 10.1145/2914920.2915026. URL `http://doi.acm.org/10.1145/2914920.2915026`. Cited on pages 21, 37, 38, and 83.

[17] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In A. Ferscha and F. Mattern, editors, *Pervasive Computing*, pages 1–17, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24646-6. Cited on page 24.

[18] J. F. Bartlett. Rock 'n' scroll is here to stay [user interface]. *IEEE Computer Graphics and Applications*, 20(3):40–45, May 2000. ISSN 0272-1716. doi: 10.1109/38.844371. Cited on page 25.

[19] D. Baur, S. Boring, and S. Feiner. Virtual projection: Exploring optical projection as a metaphor for multi-device interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1693–1702, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2208297. URL `http://doi.acm.org/10.1145/2207676.2208297`. Cited on page 37.

[20] D. Benslimane, S. Dustdar, and A. Sheth. Services mashups: The new generation of web applications. *IEEE Internet Computing*, 12(5):13–15, Sept 2008. ISSN 1089-7801. doi: 10.1109/MIC.2008.110. Cited on page 21.

[21] M. Bertarini. Myoshare: Exploiting hand-free gestures to share data among devices. Bachelor's thesis, ETH Zurich, 2016. Cited on page 145.

[22] A. Blchliger. Tilt-and-tap in 3d: Design and development of tilting interactions in 3dweb applications. Bachelor's thesis, ETH Zurich, 2015. Cited on page 145.

[23] S. Boring, M. Jurmu, and A. Butz. Scroll, tilt or move it: Using mobile phones to continuously control pointers on large public displays. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, OZCHI '09, pages 161–168, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-854-4. doi: 10.1145/1738826.1738853. URL `http://doi.acm.org/10.1145/1738826.1738853`. Cited on pages 5, 6, 9, 37, 38, 70, 81, 83, 84, and 101.

[24] S. Boring, D. Baur, A. Butz, S. Gustafson, and P. Baudisch. Touch projector: Mobile interaction through video. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2287–2296, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753671. URL `http://doi.acm.org/10.1145/1753326.1753671`. Cited on pages 38 and 39.

[25] T. Brezmes, J.-L. Gorricho, and J. Cotrina. Activity recognition from accelerometer data on a mobile phone. In S. Omatu, M. P. Rocha, J. Bravo, F. Fernández, E. Corchado, A. Bustillo, and J. M. Corchado, editors, *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pages 796–799, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-02481-8. Cited on page 24.

[26] G. Broll, W. Reithmeier, P. Holleis, and M. Wagner. Design and evaluation of techniques for mobile interaction with dynamic nfc-displays. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*,

TEI '11, pages 205–212, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0478-8. doi: 10.1145/1935701.1935743. URL `http://doi.acm.org/10.1145/1935701.1935743`. Cited on page 37.

[27] C. Buenaflor and H. cheol Kim. Six human factors to acceptability of wearable computers, 2013. Cited on page 33.

[28] L. Buethe, C. Vogt, L. Petti, N. Muenzenrieder, C. Zysset, G. A. Salvatore, and G. Troester. A mechanically flexible tilt switch on kapton foil with microspheres as a pendulum. In *Sensors and Measuring Systems 2014; 17. ITG/GMA Symposium*, pages 1–4, June 2014. Cited on page 34.

[29] A. Canonica. Re-engineering tilt-and-tap: Design and development of tilting interactions on theweb. Bachelor's thesis, ETH Zurich, 2016. Cited on page 145.

[30] B. Caramiaux, M. Donnarumma, and A. Tanaka. Understanding gesture expressivity through muscle sensing. *ACM Trans. Comput.-Hum. Interact.*, 21 (6):31:1–31:26, Jan. 2015. ISSN 1073-0516. doi: 10.1145/2687922. URL `http://doi.acm.org/10.1145/2687922`. Cited on pages 29 and 34.

[31] C. Castelluccia and P. Mutaf. Shake them up!: A movement-based pairing protocol for cpu-constrained devices. In *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, MobiSys '05, pages 51–64, New York, NY, USA, 2005. ACM. ISBN 1-931971-31-5. doi: 10.1145/1067170.1067177. URL `http://doi.acm.org/10.1145/1067170.1067177`. Cited on page 40.

[32] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, and M. Matera. *Morgan Kaufmann series in data management systems: Designing data-intensive Web applications*. Morgan Kaufmann, 2003. Cited on page 20.

[33] S. Ceri, F. Daniel, M. Matera, and F. M. Facca. Model-driven development of context-aware web applications. *ACM Trans. Internet Technol.*, 7(1), Feb. 2007. ISSN 1533-5399. doi: 10.1145/1189740.1189742. URL `http://doi.acm.org/10.1145/1189740.1189742`. Cited on page 20.

[34] L. Chan, Y.-L. Chen, C.-H. Hsieh, R.-H. Liang, and B.-Y. Chen. Cyclopsring: Enabling whole-hand and context-aware interactions through a fisheye ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology*, UIST '15, pages 549–556, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3779-3. doi: 10.1145/2807442.2807450. URL `http://doi.acm.org/10.1145/2807442.2807450`. Cited on page 33.

[35] D. Chattopadhyay and D. Bolchini. Touchless circular menus: Toward an intuitive ui for touchless interactions with large displays. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 33–40, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2775-6. doi: 10.1145/2598153.2598181. URL `http://doi.acm.org/10.1145/2598153.2598181`. Cited on page 30.

[36] F. Chehimi and P. Coulton. Motion controlled mobile 3d multiplayer gaming. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pages 267–270, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-393-8. doi: 10.1145/1501750.1501813. URL `http://doi.acm.org/10.1145/1501750.1501813`. Cited on page 27.

[37] K.-Y. Chen, D. Ashbrook, M. Goel, S.-H. Lee, and S. Patel. Airlink: Sharing files between multiple devices using in-air gestures. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 565–569, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2968-2. doi: 10.1145/2632048.2632090. URL `http://doi.acm.org/10.1145/2632048.2632090`. Cited on pages 5, 9, and 40.

[38] X. A. Chen, T. Grossman, D. J. Wigdor, and G. Fitzmaurice. Duet: Exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 159–168, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2556955. URL `http://doi.acm.org/10.1145/2556288.2556955`. Cited on page 40.

[39] P.-Y. P. Chi and Y. Li. Weave: Scripting cross-device wearable interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3923–3932, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702451. URL `http://doi.acm.org/10.1145/2702123.2702451`. Cited on page 21.

[40] S.-J. Cho, R. Murray-Smith, and Y.-B. Kim. Multi-context photo browsing on mobile devices based on tilt dynamics. In *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '07, pages 190–197, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-862-6. doi: 10.1145/1377999.1378006. URL `http://doi.acm.org/10.1145/1377999.1378006`. Cited on pages 25, 27, 28, and 50.

[41] S. R. Choudhary, H. Versee, and A. Orso. Webdiff: Automated identification of cross-browser issues in web applications. In *2010 IEEE International Conference on Software Maintenance*, pages 1–10, Sept 2010. doi: 10.1109/ICSM.2010.5609723. Cited on page 19.

[42] S. R. Choudhary, M. R. Prasad, and A. Orso. Crosscheck: Combining crawling and differencing to better detect cross-browser incompatibilities in web applications. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pages 171–180, April 2012. doi: 10.1109/ICST.2012.97. Cited on page 19.

[43] S. Clinch, N. Davies, T. Kubitza, and A. Friday. Ownership and trust in cyber-foraged displays. In *Proceedings of The International Symposium on Pervasive Displays*, PerDis '14, pages 168:168–168:173, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2952-1. doi: 10.1145/2611009.2611010. URL `http://doi.acm.org/10.1145/2611009.2611010`. Cited on page 36.

[44] A. Cockburn, D. Ahlstrm, and C. Gutwin. Understanding performance in touch selections: Tap, drag and radial pointing drag with finger, stylus and mouse. *International Journal of Human-Computer Studies*, 70(3):218 – 233, 2012. ISSN 1071-5819. doi: https://doi.org/10.1016/j.ijhcs.2011.11.002. URL `http://www.sciencedirect.com/science/article/pii/S1071581911001546`. Cited on page 2.

[45] R. Dachselt and R. Buchholz. Natural throw and tilt interaction between mobile phones and distant displays. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 3253–3258, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-247-4. doi: 10.1145/1520340.1520467. URL `http://doi.acm.org/10.1145/1520340.1520467`. Cited on pages 5, 36, 38, 39, 40, 41, and 83.

[46] F. Daniel, S. Soi, S. Tranquillini, F. Casati, C. Heng, and L. Yan. Distributed orchestration of user interfaces. *Information Systems*, 37(6):539 – 556, 2012. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2011.08.001. URL `http://www.sciencedirect.com/science/article/pii/S0306437911001050`. BPM 2010. Cited on page 23.

[47] N. Davies, S. Clinch, and F. Alt. Pervasive displays: understanding the future of digital signage. *Synthesis Lectures on Mobile and Pervasive Computing*, 8 (1):1–128, 2014. doi: 10.2200/S00558ED1V01Y201312MPC011. URL `https://doi.org/10.2200/S00558ED1V01Y201312MPC011`. Cited on page 37.

[48] D. Dearman and J. S. Pierce. It's on my other computer!: Computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 767–776, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: 10.1145/1357054.1357177. URL `http://doi.acm.org/10.1145/1357054.1357177`. Cited on page 36.

[49] L. Di Geronimo and M. C. Norrie. Rapid development of web applications that use tilting interactions in single and multi-device scenarios. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '16, pages 354–355, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4131-8. doi: 10.1145/2909132.2926088. URL `http://doi.acm.org/10.1145/2909132.2926088`. Cited on page 45.

[50] L. Di Geronimo, E. Aras, and M. C. Norrie. Tilt-and-tap: Framework to support motion-based web interaction techniques. In P. Cimiano, F. Frasincar, G.-J. Houben, and D. Schwabe, editors, *Engineering the Web in the Big Data Era*, pages 565–582, Cham, 2015. Springer International Publishing. ISBN 978-3-319-19890-3. Cited on pages 45 and 47.

[51] L. Di Geronimo, A. Murolo, M. Nebeling, and M. C. Norrie. Mixing and mashing website themes. In P. Cimiano, F. Frasincar, G.-J. Houben, and D. Schwabe, editors, *Engineering the Web in the Big Data Era*, pages 34–51, Cham, 2015. Springer International Publishing. ISBN 978-3-319-19890-3. Cited on page 23.

[52] L. Di Geronimo, M. Husmann, and M. C. Norrie. Surveying personal device ecosystems with cross-device applications in mind. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays*, PerDis '16, pages 220–227, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4366-4. doi: 10.1145/2914920.2915028. URL `http://doi.acm.org/10.1145/2914920.2915028`. Cited on pages 5, 9, 13, 35, 36, 42, and 107.

[53] L. Di Geronimo, M. Husmann, A. Patel, C. Tuerk, and M. C. Norrie. Ctat: Tilt-and-tap across devices. In A. Bozzon, P. Cudre-Maroux, and C. Pautasso, editors, *Web Engineering*, pages 96–113, Cham, 2016. Springer International Publishing. ISBN 978-3-319-38791-8. Cited on page 83.

[54] L. Di Geronimo, M. Bertarini, J. Badertscher, M. Husmann, and M. C. Norrie. Exploiting mid-air gestures to share data among devices. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 35:1–35:11, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5075-4. doi: 10.1145/3098279.3098530. URL `http://doi.acm.org/10.1145/3098279.3098530`. Cited on page 107.

[55] L. Di Geronimo, M. Bertarini, J. Badertscher, M. Husmann, and M. C. Norrie. Myoshare: Sharing data among devices via mid-air gestures. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '17, pages 48:1–48:3, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5075-4. doi: 10.1145/3098279.3125436. URL `http://doi.acm.org/10.1145/3098279.3125436`. Cited on page 107.

[56] L. Di Geronimo, A. Canonica, M. Husmann, and M. C. Norrie. Continuous tilting interaction techniques on mobile devices for controlling public displays. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '17, pages 21–26, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5083-9. doi: 10.1145/3102113.3102120. URL `http://doi.acm.org/10.1145/3102113.3102120`. Cited on pages 45 and 83.

[57] L. Di Geronimo, S. Kalbermatter, and M. C. Norrie. End-user web development tool for tilting interactions. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '17, pages 9–14, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5083-9. doi: 10.1145/3102113.3102117. URL `http://doi.acm.org/10.1145/3102113.3102117`. Cited on page 45.

[58] N. K. Dim and X. Ren. Designing motion gesture interfaces in mobile phones for blind people. *Journal of Computer Science and Technology*, 29(5):812–824, Sep 2014. ISSN 1860-4749. doi: 10.1007/s11390-014-1470-5. URL `https://doi.org/10.1007/s11390-014-1470-5`. Cited on page 29.

[59] N. K. Dim, K. Kim, and X. Ren. Designing motion marking menus for people with visual impairments. *International Journal of Human-Computer Studies*, 109:79 – 88, 2018. ISSN 1071-5819. doi: https://doi.org/10.1016/j.ijhcs.2017.09.002. URL `http://www.sciencedirect.com/science/article/pii/S1071581917301283`. Cited on page 29.

[60] T. Dingler, T. Bagg, Y. Grau, N. Henze, and A. Schmidt. ucanvas: A web framework for spontaneous smartphone interaction with ubiquitous displays. In J. Abascal, S. Barbosa, M. Fetter, T. Gross, P. Palanque, and M. Winckler, editors, *Human-Computer Interaction – INTERACT 2015*, pages 402–409, Cham, 2015. Springer International Publishing. ISBN 978-3-319-22698-9. Cited on page 38.

[61] T. Dong, E. F. Churchill, and J. Nichols. Understanding the challenges of designing and developing multi-device experiences. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, DIS '16, pages 62–72, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4031-1. doi: 10.1145/2901790.2901851. URL `http://doi.acm.org/10.1145/2901790.2901851`. Cited on page 20.

[62] D. Efron and S. van Veen. *Gesture, race and culture*. Mouton, 1972. Cited on page 29.

[63] P. Ekman and W. V. Friesen. The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *semiotica*, 1(1):49–98, 1969. Cited on page 29.

[64] P. Eslambolchilar, J. Williamson, and R. Murray-Smith. Multimodal feedback for tilt controlled speed dependent automatic zooming. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software &#38; Technology*, UIST '04, New York, NY, USA, 2004. ACM. Cited on page 29.

[65] M. Feldmann, T. Nestler, K. Muthmann, U. Jugel, G. Hübsch, and A. Schill. Overview of an end-user enabled model-driven development approach for interactive applications based on annotated services. In *Proceedings of the 4th Workshop on Emerging Web Services Technology*, WEWST '09, pages 19–28, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-776-9. doi: 10.1145/1645406.1645410. URL `http://doi.acm.org/10.1145/1645406.1645410`. Cited on page 20.

[66] E. Ferrara, P. D. Meo, G. Fiumara, and R. Baumgartner. Web data extraction, applications and techniques: A survey. *Knowledge-Based Systems*, 70:301 – 323, 2014. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2014.07.007. URL `http://www.sciencedirect.com/science/article/pii/S0950705114002640`. Cited on page 125.

[67] S. Fitchett and A. Cockburn. Evaluating reading and analysis tasks on mobile devices: A case study of tilt and flick scrolling. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*, OZCHI '09, pages 225–232, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-854-4. doi: 10.1145/1738826.1738863. URL `http://doi.acm.org/10.1145/1738826.1738863`. Cited on pages 27 and 135.

[68] N. Freedman and S. P. Hoffman. Kinetic behavior in altered clinical states: Approach to objective analysis of motor behavior during clinical interviews. *Perceptual and Motor Skills*, 24(2):527–539, 1967. doi: 10.2466/pms.1967.24.2.527. URL `https://doi.org/10.2466/pms.1967.24.2.527`. PMID: 6040227. Cited on page 29.

[69] E. Freeman, S. Brewster, and V. Lantz. Do that, there: An interaction technique for addressing in-air gesture systems. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2319–2331, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858308. URL `http://doi.acm.org/10.1145/2858036.2858308`. Cited on page 9.

[70] L. Frosini and F. Paternò. User interface distribution in multi-device and multi-user environments with dynamically migrating engines. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '14, pages 55–64, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2725-1. doi: 10.1145/2607023.2607032. URL `http://doi.acm.org/10.1145/2607023.2607032`. Cited on page 20.

[71] M. Geel, T. Church, and M. C. Norrie. Sift: An end-user tool for gathering web content on the go. In *Proceedings of the 2012 ACM Symposium on Document Engineering*, DocEng '12, pages 181–190, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1116-8. doi: 10.1145/2361354.2361395. URL `http://doi.acm.org/10.1145/2361354.2361395`. Cited on page 125.

[72] M. Geel, D. Huguenin, and M. C. Norrie. Presishare: Opportunistic sharing and presentation of content using public displays and qr codes. In *Proceedings of the 2Nd ACM International Symposium on Pervasive Displays*, PerDis '13, pages 103–108, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2096-2. doi: 10.1145/2491568.2491591. URL `http://doi.acm.org/10.1145/2491568.2491591`. Cited on page 40.

[73] C. P. Gerba, A. L. Wuollet, P. Raisanen, and G. U. Lopez. Bacterial contamination of computer touch screens. *American Journal of Infection Control*, 44(3): 358–360, 2018/04/06 XXXX. ISSN 0196-6553. doi: 10.1016/j.ajic.2015.10.013. URL `http://dx.doi.org/10.1016/j.ajic.2015.10.013`. Cited on page 37.

[74] G. Ghiani, F. Paternò, and L. D. Spano. Creating mashups by direct manipulation of existing web applications. In M. F. Costabile, Y. Dittrich, G. Fischer, and A. Piccinno, editors, *End-User Development*, pages 42–52, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21530-8. Cited on page 21.

[75] P. Gilbertson, P. Coulton, F. Chehimi, and T. Vajk. Using &ldquo;tilt&rdquo; as an interface to control &ldquo;no-button&rdquo; 3-d mobile games. *Comput. Entertain.*, 6(3):38:1–38:13, Nov. 2008. ISSN 1544-3574. doi: 10.1145/1394021.1394031. URL `http://doi.acm.org/10.1145/1394021.1394031`. Cited on page 27.

[76] M. Goel, B. Lee, M. T. Islam Aumi, S. Patel, G. Borriello, S. Hibino, and B. Begole. Surfacelink: Using inertial and acoustic sensing to enable multi-device interaction on a surface. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1387–1396, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557120. URL `http://doi.acm.org/10.1145/2556288.2557120`. Cited on page 40.

[77] Google. Project soli 2016, Accessed on June 2018. URL https://atap.google.com/soli. Cited on page 140.

[78] S. A. Grandhi, G. Joue, and I. Mittelberg. Understanding naturalness and intuitiveness in gesture production: Insights for touchless gestural interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 821–824, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979061. URL http://doi.acm.org/10.1145/1978942.1979061. Cited on page 30.

[79] Y. Guiard. On fitts's and hooke's laws: Simple harmonic movement in upper-limb cyclical aiming. *Acta Psychologica*, 82(1):139 – 159, 1993. ISSN 0001-6918. doi: https://doi.org/10.1016/0001-6918(93)90009-G. URL http://www.sciencedirect.com/science/article/pii/000169189390009G. Cited on page 26.

[80] P. Hamilton and D. J. Wigdor. Conductor: Enabling and understanding cross-device interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2773–2782, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557170. URL http://doi.acm.org/10.1145/2556288.2557170. Cited on pages 5, 13, 40, 41, and 132.

[81] F. Haque, M. Nancel, and D. Vogel. Myopoint: Pointing and clicking using forearm mounted electromyography and inertial motion sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3653–3656, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702133. URL http://doi.acm.org/10.1145/2702123.2702133. Cited on page 39.

[82] R. Hardy and E. Rukzio. Touch & interact: Touch-based interaction of mobile phones with displays. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pages 245–254, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-952-4. doi: 10.1145/1409240.1409267. URL http://doi.acm.org/10.1145/1409240.1409267. Cited on pages 37 and 38.

[83] R. Hardy, E. Rukzio, P. Holleis, and M. Wagner. Mobile interaction with static and dynamic nfc-based displays. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '10, pages 123–132, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-835-3. doi: 10.1145/1851600.1851623. URL http://doi.acm.org/10.1145/1851600.1851623. Cited on page 37.

[84] B. Hartmann, L. Abdulla, M. Mittal, and S. R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 145–154, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: 10.1145/1240624.1240646. URL http://doi.acm.org/10.1145/1240624.1240646. Cited on page 19.

[85] B. Hartmann, L. Wu, K. Collins, and S. R. Klemmer. Programming by a sample: Rapidly creating web applications with d.mix. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 241–250, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-679-0. doi: 10.1145/1294211.1294254. URL `http://doi.acm.org/10.1145/1294211.1294254`. Cited on page 21.

[86] K. Hasan, D. Ahlström, and P. Irani. Sammi: A spatially-aware multi-mobile interface for analytic map navigation tasks. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 36–45, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3652-9. doi: 10.1145/2785830.2785850. URL `http://doi.acm.org/10.1145/2785830.2785850`. Cited on page 33.

[87] K. Hasan, D. Ahlström, J. Kim, and P. Irani. Airpanes: Two-handed around-device interaction for pane switching on smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 679–691, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3026029. URL `http://doi.acm.org/10.1145/3025453.3026029`. Cited on page 31.

[88] S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, pages 13:1–13:14, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2027-6. doi: 10.1145/2517351.2517367. URL `http://doi.acm.org/10.1145/2517351.2517367`. Cited on page 24.

[89] M. Hesenius, T. Griebe, S. Gries, and V. Gruhn. Automating ui tests for mobile applications with formal gesture descriptions. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices &#38; Services*, MobileHCI '14, pages 213–222, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3004-6. doi: 10.1145/2628363.2628391. URL `http://doi.acm.org/10.1145/2628363.2628391`. Cited on page 19.

[90] J. D. Hincapié-Ramos, X. Guo, P. Moghadasian, and P. Irani. Consumed endurance: A metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1063–1072, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557130. URL `http://doi.acm.org/10.1145/2556288.2557130`. Cited on page 31.

[91] K. Hinckley. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 149–158, New York, NY, USA, 2003. ACM. ISBN 1-58113-636-6. doi: 10.1145/964696.964713. URL `http://doi.acm.org/10.1145/964696.964713`. Cited on pages 39 and 40.

[92] K. Hinckley and H. Song. Sensor synaesthesia: Touch in motion, and motion in touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing*

Systems, CHI '11, pages 801–810, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979059. URL `http://doi.acm.org/10.1145/1978942.1979059`. Cited on pages 2, 5, 6, 29, 46, and 80.

[93] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz. Sensing techniques for mobile interaction. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, pages 91–100, New York, NY, USA, 2000. ACM. ISBN 1-58113-212-3. doi: 10.1145/354401.354417. URL `http://doi.acm.org/10.1145/354401.354417`. Cited on pages 24 and 29.

[94] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith. Stitching: Pen gestures that span multiple displays. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, pages 23–31, New York, NY, USA, 2004. ACM. ISBN 1-58113-867-9. doi: 10.1145/989863.989866. URL `http://doi.acm.org/10.1145/989863.989866`. Cited on pages 39 and 40.

[95] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl5, and H.-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In G. D. Abowd, B. Brumitt, and S. Shafer, editors, *Ubicomp 2001: Ubiquitous Computing*, pages 116–122, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45427-4. Cited on pages 39 and 40.

[96] C. Holz and P. Baudisch. Understanding touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2501–2510, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979308. URL `http://doi.acm.org/10.1145/1978942.1979308`. Cited on pages 25 and 42.

[97] S. Houben, S. Nielsen, M. Esbensen, and J. E. Bardram. Noosphere: An activity-centric infrastructure for distributed interaction. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pages 13:1–13:10, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2648-3. doi: 10.1145/2541831.2541856. URL `http://doi.acm.org/10.1145/2541831.2541856`. Cited on pages 15, 20, and 42.

[98] S. Houben, N. Marquardt, J. Vermeulen, C. Klokmose, J. Schöning, H. Reiterer, and C. Holz. Opportunities and challenges for cross-device interactions in the wild. *interactions*, 24(5):58–63, Aug. 2017. ISSN 1072-5520. doi: 10.1145/3121348. URL `http://doi.acm.org/10.1145/3121348`. Cited on page 39.

[99] Z. Huang, W. Li, and P. Hui. Ubii: Towards seamless interaction between digital and physical worlds. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 341–350, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3459-4. doi: 10.1145/2733373.2806266. URL `http://doi.acm.org/10.1145/2733373.2806266`. Cited on page 40.

[100] M. Husmann and M. C. Norrie. Xd-mvc: Support for cross-device development. In *1st Intl. Workshop on Interacting with Multi-Device Ecologies in the Wild (Cross-Surface 2015)*. ETH Zürich, 2015. Cited on pages 20 and 96.

[101] M. Husmann, M. Nebeling, and M. C. Norrie. Multimasher: A visual tool for multi-device mashups. In Q. Z. Sheng and J. Kjeldskov, editors, *Current Trends in Web Engineering*, pages 27–38, Cham, 2013. Springer International Publishing. ISBN 978-3-319-04244-2. Cited on page 23.

[102] M. Husmann, S. Chithambaram, and M. C. Norrie. Combining physical and social proximity for device pairing. In *Proceedings of Cross-Surface 2016: Workshop on Challenges and Opportunities of Spatial and Proxemic Interaction*, 2016. Cross-Surface 2016: Workshop on Challenges and Opportunities of Spatial and Proxemic Interaction; Conference Location: Niagara Falls, Canada; Conference Date: November 6-9, 2016; . Cited on page 40.

[103] D. Hüttig. Beyondmyoshare. Bachelor's thesis, ETH Zurich, 2017. Cited on page 145.

[104] S. Ismair, J. Wagner, T. Selker, and A. Butz. Mime: Teaching mid-air pose-command mappings. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pages 199–206, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3652-9. doi: 10.1145/2785830.2785854. URL `http://doi.acm.org/10.1145/2785830.2785854`. Cited on pages 30 and 31.

[105] G. Jancke, G. D. Venolia, J. Grudin, J. J. Cadiz, and A. Gupta. Linking public spaces: Technical and social issues. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, pages 530–537, New York, NY, USA, 2001. ACM. ISBN 1-58113-327-8. doi: 10.1145/365024.365352. URL `http://doi.acm.org/10.1145/365024.365352`. Cited on page 36.

[106] T. Jokela, J. Ojala, and T. Olsson. A diary study on combining multiple information devices in everyday activities and tasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3903–3912, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702211. URL `http://doi.acm.org/10.1145/2702123.2702211`. Cited on page 36.

[107] B. Jones, R. Sodhi, D. Forsyth, B. Bailey, and G. Maciocci. Around device interaction for multiscale navigation. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, pages 83–92, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1105-2. doi: 10.1145/2371574.2371589. URL `http://doi.acm.org/10.1145/2371574.2371589`. Cited on page 33.

[108] E. Jones, J. Alexander, A. Andreou, P. Irani, and S. Subramanian. Gestext: Accelerometer-based gestural text-entry systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 2173–2182, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753655. URL `http://doi.acm.org/10.1145/1753326.1753655`. Cited on page 2.

[109] S. Kalbermatter. End user development tool for tilt-and-tap. Bachelor's thesis, ETH Zurich, 2016. Cited on page 145.

[110] M. Karam and M. C. Schraefel. A taxonomy of gestures in human computer interactions, 2005. URL `https://eprints.soton.ac.uk/261149/`. Cited on pages 30, 31, and 111.

[111] K. Katsuragawa, K. Pietroszek, J. R. Wallace, and E. Lank. Watchpoint: Freehand pointing with a smartwatch in a ubiquitous display environment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '16, pages 128–135, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4131-8. doi: 10.1145/2909132.2909263. URL `http://doi.acm.org/10.1145/2909132.2909263`. Cited on page 39.

[112] A. Kendon. Gestures as illocutionary and discourse structure markers in southern italian conversation. *Journal of Pragmatics*, 23(3):247 – 279, 1995. ISSN 0378-2166. doi: https://doi.org/10.1016/0378-2166(94)00037-F. URL `http://www.sciencedirect.com/science/article/pii/037821669400037F`. Cited on page 29.

[113] W. Kienzle and K. Hinckley. Lightring: Always-available 2d input on any surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 157–160, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647376. URL `http://doi.acm.org/10.1145/2642918.2647376`. Cited on page 33.

[114] D. Kim, O. Hilliges, S. Izadi, A. D. Butler, J. Chen, I. Oikonomidis, and P. Olivier. Digits: Freehand 3d interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 167–176, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7. doi: 10.1145/2380116.2380139. URL `http://doi.acm.org/10.1145/2380116.2380139`. Cited on page 33.

[115] M. Kim and J. Y. Lee. Touch and hand gesture-based interactions for directly manipulating 3d virtual objects in mobile augmented reality. *Multimedia Tools and Applications*, 75(23):16529–16550, Dec 2016. ISSN 1573-7721. doi: 10.1007/s11042-016-3355-9. URL `https://doi.org/10.1007/s11042-016-3355-9`. Cited on page 32.

[116] J. Knibbe, D. Martinez Plasencia, C. Bainbridge, C.-K. Chan, J. Wu, T. Cable, H. Munir, and D. Coyle. Extending interaction for smart watches: Enabling bimanual around device control. In *Proceedings of the Extended Abstracts of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI EA '14, pages 1891–1896, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2474-8. doi: 10.1145/2559206.2581315. URL `http://doi.acm.org/10.1145/2559206.2581315`. Cited on page 30.

[117] W. A. König, R. Rädle, and H. Reiterer. Interactive design of multimodal user interfaces. *Journal on Multimodal User Interfaces*, 3(3):197–213, Apr 2010. ISSN

1783-8738. doi: 10.1007/s12193-010-0044-2. URL `https://doi.org/10.1007/s12193-010-0044-2`. Cited on page 23.

[118] D. Kovachev, D. Renzel, P. Nicolaescu, and R. Klamma. Direwolf - distributing and migrating user interfaces for widget-based web applications. In F. Daniel, P. Dolog, and Q. Li, editors, *Web Engineering*, pages 99–113, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-39200-9. Cited on page 23.

[119] S. Kratz and M. Rohs. Hoverflow: Expanding the design space of around-device interaction. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 4:1–4:8, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-281-8. doi: 10.1145/1613858.1613864. URL `http://doi.acm.org/10.1145/1613858.1613864`. Cited on page 33.

[120] S. Kratz, M. Rohs, D. Guse, J. Müller, G. Bailly, and M. Nischt. Palmspace: Continuous around-device gestures vs. multitouch for 3d rotation tasks on mobile devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 181–188, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1287-5. doi: 10.1145/2254556.2254590. URL `http://doi.acm.org/10.1145/2254556.2254590`. Cited on pages 31 and 32.

[121] C. Kray, D. Nesbitt, J. Dawson, and M. Rohs. User-defined gestures for connecting mobile phones, public displays, and tabletops. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '10, pages 239–248, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-835-3. doi: 10.1145/1851600.1851640. URL `http://doi.acm.org/10.1145/1851600.1851640`. Cited on pages 41, 42, and 109.

[122] T. Kuribara, B. Shizuki, and J. Tanaka. Hoverlink: Joint interactions using hover sensing capability. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 1651–1656, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3146-3. doi: 10.1145/2702613.2732822. URL `http://doi.acm.org/10.1145/2702613.2732822`. Cited on page 40.

[123] B. Laurel and S. J. Mountford, editors. *Gestures in human-computer comminication. In The Art of The Human-Computer Interface Design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990. ISBN 0201517973. Cited on page 29.

[124] D. Lee, J.-I. Hwang, G. J. Kim, and S. C. Ahn. 3d interaction using mobile device on 3d environments with large screen. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 575–580, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0541-9. doi: 10.1145/2037373.2037463. URL `http://doi.acm.org/10.1145/2037373.2037463`. Cited on pages 37 and 39.

[125] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. In K. P. Fishkin, B. Schiele, P. Nixon, and A. Quigley, editors, *Pervasive Computing*, pages 1–16, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33895-6. Cited on page 24.

[126] H. Li, X. Zhao, Z. Xing, L. Bao, X. Peng, D. Gao, and W. Zhao. amassist: In-ide ambient search of online programming resources. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 390–398, March 2015. doi: 10.1109/SANER.2015.7081849. Cited on page 17.

[127] Y. T. Li, G. Chen, and M. T. Sun. An indoor collaborative pedestrian dead reckoning system. In *2013 42nd International Conference on Parallel Processing*, pages 923–930, Oct 2013. doi: 10.1109/ICPP.2013.110. Cited on page 139.

[128] H. Lieberman, F. Paternò, M. Klann, and V. Wulf. *End-User Development: An Emerging Paradigm*, pages 1–8. Springer Netherlands, Dordrecht, 2006. ISBN 978-1-4020-5386-3. doi: 10.1007/1-4020-5386-X_1. URL `https://doi.org/10.1007/1-4020-5386-X_1`. Cited on pages 16 and 21.

[129] A. Lucero, J. Holopainen, and T. Jokela. Pass-them-around: Collaborative use of mobile phones for photo sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1787–1796, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979201. URL `http://doi.acm.org/10.1145/1978942.1979201`. Cited on pages 40 and 107.

[130] V. Mäkelä, M. Khamis, L. Mecke, J. James, M. Turunen, and F. Alt. Pocket transfers: Interaction techniques for transferring content from situated displays to mobile devices. Cited on page 37.

[131] N. Marquardt, R. Diaz-Marino, S. Boring, and S. Greenberg. The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 315–326, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047238. URL `http://doi.acm.org/10.1145/2047196.2047238`. Cited on page 21.

[132] N. Marquardt, T. Ballendat, S. Boring, S. Greenberg, and K. Hinckley. Gradual engagement: Facilitating information exchange between digital devices as a function of proximity. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '12, pages 31–40, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1209-7. doi: 10.1145/2396636.2396642. URL `http://doi.acm.org/10.1145/2396636.2396642`. Cited on page 40.

[133] N. Marquardt, K. Hinckley, and S. Greenberg. Cross-device interaction via micro-mobility and f-formations. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 13–22, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7. doi: 10.1145/2380116.2380121. URL `http://doi.acm.org/10.1145/2380116.2380121`. Cited on page 40.

[134] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp)iphone: Decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 551–562, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0948-6. doi: 10.1145/2046707.2046771. URL `http://doi.acm.org/10.1145/2046707.2046771`. Cited on page 24.

[135] T. L. Martin. Time and time again: parallels in the development of the watch and the wearable computer. In *Proceedings. Sixth International Symposium on Wearable Computers,*, pages 5–11, 2002. doi: 10.1109/ISWC.2002.1167212. Cited on page 33.

[136] F. Matulic, M. Husmann, S. Walter, and M. C. Norrie. Eyes-free touch command support for pen-based digital whiteboards via handheld devices. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, pages 141–150, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3899-8. doi: 10.1145/2817721.2817728. URL `http://doi.acm.org/10.1145/2817721.2817728`. Cited on page 37.

[137] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Transactions on Mobile Computing*, 8(6):792–806, June 2009. ISSN 1536-1233. doi: 10.1109/TMC.2009.51. Cited on page 40.

[138] D. McNeill. *Hand and mind: What gestures reveal about thought.* University of Chicago press, 1992. Cited on pages 29 and 30.

[139] A. Melro, B. Silva, and R. José. Media sharing in situated displays: Service design lessons from existing practices with paper leaflets. In J. Falcão e Cunha, M. Snene, and H. Nóvoa, editors, *Exploring Services Science*, pages 322–328, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-36356-6. Cited on page 36.

[140] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3): 311–324, May 2007. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.893280. Cited on page 30.

[141] L. Moreno, G. Bavota, M. D. Penta, R. Oliveto, and A. Marcus. How can i use this method? In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 880–890, May 2015. doi: 10.1109/ICSE.2015.98. Cited on page 17.

[142] S. Morrison-Smith, M. Hofmann, Y. Li, and J. Ruiz. Using audio cues to support motion gesture interaction on mobile devices. *ACM Trans. Appl. Percept.*, 13 (3):16:1–16:19, May 2016. ISSN 1544-3558. doi: 10.1145/2897516. URL `http://doi.acm.org/10.1145/2897516`. Cited on page 29.

[143] J. Müller, R. Walter, G. Bailly, M. Nischt, and F. Alt. Looking glass: A field study on noticing interactivity of a shop window. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 297–306, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/

2207676.2207718. URL `http://doi.acm.org/10.1145/2207676.2207718`. Cited on page 37.

[144] A. Murolo, F. Stutz, M. Husmann, and M. C. Norrie. Improved developer support for the detection of cross-browser incompatibilities. In J. Cabot, R. De Virgilio, and R. Torlone, editors, *Web Engineering*, pages 264–281, Cham, 2017. Springer International Publishing. ISBN 978-3-319-60131-1. Cited on pages 4 and 18.

[145] A. Mylonas, V. Meletiadis, L. Mitrou, and D. Gritzalis. Smartphone sensor data as digital evidence. *Computers & Security*, 38:51 – 75, 2013. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2013.03.007. URL `http://www.sciencedirect.com/science/article/pii/S0167404813000527`. Cybercrime in the Digital Economy. Cited on page 24.

[146] M. A. Nacenta, Y. Kamber, Y. Qiang, and P. O. Kristensson. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1099–1108, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466142. URL `http://doi.acm.org/10.1145/2470654.2466142`. Cited on page 41.

[147] M. Nancel, J. Wagner, E. Pietriga, O. Chapuis, and W. Mackay. Mid-air pan-and-zoom on wall-sized displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 177–186, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1978969. URL `http://doi.acm.org/10.1145/1978942.1978969`. Cited on page 30.

[148] M. Nancel, E. Pietriga, O. Chapuis, and M. Beaudouin-Lafon. Mid-air pointing on ultra-walls. *ACM Trans. Comput.-Hum. Interact.*, 22(5):21:1–21:62, Aug. 2015. ISSN 1073-0516. doi: 10.1145/2766448. URL `http://doi.acm.org/10.1145/2766448`. Cited on page 37.

[149] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota. Fingerio: Using active sonar for fine-grained finger tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1515–1525, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858580. URL `http://doi.acm.org/10.1145/2858036.2858580`. Cited on pages 2, 31, and 33.

[150] M. Nebeling. Xdbrowser 2.0: Semi-automatic generation of cross-device interfaces. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 4574–4584, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025547. URL `http://doi.acm.org/10.1145/3025453.3025547`. Cited on pages 41 and 42.

[151] M. Nebeling and M. Norrie. jqmultitouch: Lightweight toolkit and development framework for multi-touch/multi-device web interfaces. In *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '12, pages 61–70, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1168-7. doi: 10.1145/2305484.2305497. URL `http://doi.acm.org/10.1145/2305484.2305497`. Cited on pages 3 and 18.

[152] M. Nebeling and M. C. Norrie. Beyond responsive design: Adaptation to touch and multitouch. In S. Casteleyn, G. Rossi, and M. Winckler, editors, *Web Engineering: 14th International Conference, ICWE 2014, Toulouse, France, July 1-4, 2014. Proceedings*, pages 380–389, Cham, 2014. Springer International Publishing. ISBN 978-3-319-08245-5. doi: 10.1007/978-3-319-08245-5_23. URL `https://doi.org/10.1007/978-3-319-08245-5_23`. Cited on pages 2 and 42.

[153] M. Nebeling, A. Huber, D. Ott, and M. C. Norrie. Web on the wall reloaded: Implementation, replication and refinement of user-defined interaction sets. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, pages 15–24, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2587-5. doi: 10.1145/2669485.2669497. URL `http://doi.acm.org/10.1145/2669485.2669497`. Cited on page 37.

[154] M. Nebeling, T. Mintsi, M. Husmann, and M. Norrie. Interactive development of cross-device user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 2793–2802, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2556980. URL `http://doi.acm.org/10.1145/2556288.2556980`. Cited on page 23.

[155] M. Nebeling, E. Teunissen, M. Husmann, and M. C. Norrie. Xdkinect: Development framework for cross-device interaction using kinect. In *Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '14, pages 65–74, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2725-1. doi: 10.1145/2607023.2607024. URL `http://doi.acm.org/10.1145/2607023.2607024`. Cited on page 37.

[156] M. Negulescu, J. Ruiz, Y. Li, and E. Lank. Tap, swipe, or move: Attentional demands for distracted smartphone input. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 173–180, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1287-5. doi: 10.1145/2254556.2254589. URL `http://doi.acm.org/10.1145/2254556.2254589`. Cited on pages 25 and 81.

[157] M. Nielsen, M. Störring, T. B. Moeslund, and E. Granum. A procedure for developing intuitive and ergonomic gesture interfaces for hci. In A. Camurri and G. Volpe, editors, *Gesture-Based Communication in Human-Computer Interaction*, pages 409–420, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24598-8. Cited on page 41.

[158] T. Niikura, Y. Hirobe, A. Cassinelli, Y. Watanabe, T. Komuro, and M. Ishikawa. In-air typing interface for mobile devices with vibration feedback. In *ACM SIGGRAPH 2010 Emerging Technologies*, SIGGRAPH '10, pages 15:1–15:1, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0392-7. doi: 10.1145/1836821.1836836. URL `http://doi.acm.org/10.1145/1836821.1836836`. Cited on pages 31 and 32.

[159] M. Nitsche and S. Nayak. Cell phone puppets: Turning mobile phones into performing objects. In M. Herrlich, R. Malaka, and M. Masuch, editors, *Entertain-*

ment Computing - ICEC 2012, pages 363–372, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33542-6. Cited on page 27.

[160] M. C. Norrie, L. Di Geronimo, A. Murolo, and M. Nebeling. The forgotten many? a survey of modern web development practices. In S. Casteleyn, G. Rossi, and M. Winckler, editors, *Web Engineering: 14th International Conference, ICWE 2014, Toulouse, France, July 1-4, 2014. Proceedings*, pages 290–307, Cham, 2014. Springer International Publishing. ISBN 978-3-319-08245-5. doi: 10.1007/978-3-319-08245-5_17. URL `https://doi.org/10.1007/978-3-319-08245-5_17`. Cited on pages 4, 22, and 61.

[161] M. C. Norrie, M. Nebeling, L. Di Geronimo, and A. Murolo. X-themes: Supporting design-by-example. In S. Casteleyn, G. Rossi, and M. Winckler, editors, *Web Engineering*, pages 480–489, Cham, 2014. Springer International Publishing. ISBN 978-3-319-08245-5. Cited on page 23.

[162] I. Oakley and S. O'Modhrain. Tilt to scroll: evaluating a motion based vibrotactile mobile interface. In *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics Conference*, pages 40–49, March 2005. doi: 10.1109/WHC.2005.138. Cited on pages 6, 9, 27, 101, 106, and 139.

[163] I. Oakley and J. Park. Motion marking menus: An eyes-free approach to motion input for handheld devices. *International Journal of Human-Computer Studies*, 67(6):515 – 532, 2009. ISSN 1071-5819. doi: https://doi.org/10.1016/j.ijhcs.2009.02.002. URL `http://www.sciencedirect.com/science/article/pii/S1071581909000226`. Cited on pages 25, 27, and 81.

[164] T. Ohta and J. Tanaka. Pinch: An interface that relates applications on multiple touch-screen by 'pinching' gesture. In A. Nijholt, T. Romão, and D. Reidsma, editors, *Advances in Computer Entertainment*, pages 320–335, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34292-9. Cited on pages 39 and 40.

[165] A. Oulasvirta and L. Sumari. Mobile kits and laptop trays: Managing multiple devices in mobile information work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1127–1136, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: 10.1145/1240624.1240795. URL `http://doi.acm.org/10.1145/1240624.1240795`. Cited on page 35.

[166] J. Paay, D. Raptis, J. Kjeldskov, M. B. Skov, E. V. Ruder, and B. M. Lauridsen. Investigating cross-device interaction between a handheld device and a large display. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6608–6619, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025724. URL `http://doi.acm.org/10.1145/3025453.3025724`. Cited on page 37.

[167] S. Park, C. Gebhardt, R. Rädle, A. Feit, H. Vrzakova, N. Dayama, H.-S. Yeo, C. Klokmose, A. Quigley, A. Oulasvirta, et al. Adam: Adapting multi-

user interfaces for collaborative environments in real-time. *arXiv preprint arXiv:1803.01166*, 2018. Cited on page 20.

[168] A. Patel. Cross-tilt-and-tap: a framework for tilting interactions in a cross-device environment. Bachelor's thesis, ETH Zurich, 2016. Cited on page 145.

[169] P. Paudyal, A. Banerjee, and S. K. Gupta. Sceptre: A pervasive, non-invasive, and programmable gesture recognition technology. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, IUI '16, pages 282–293, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4137-0. doi: 10.1145/2856767.2856794. URL `http://doi.acm.org/10.1145/2856767.2856794`. Cited on pages 33 and 34.

[170] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007. doi: 10.2753/MIS0742-1222240302. URL `https://doi.org/10.2753/MIS0742-1222240302`. Cited on page 10.

[171] K. Pietroszek, L. Tahai, J. R. Wallace, and E. Lank. 3d interaction with networked public displays using mobile and wearable devices. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, UbiComp/ISWC'15 Adjunct, pages 787–788, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3575-1. doi: 10.1145/2800835.2807957. URL `http://doi.acm.org/10.1145/2800835.2807957`. Cited on page 39.

[172] K. Pietroszek, J. R. Wallace, and E. Lank. Tiltcasting: 3d interaction on large displays using a mobile device. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology*, UIST '15, pages 57–62, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3779-3. doi: 10.1145/2807442.2807471. URL `http://doi.acm.org/10.1145/2807442.2807471`. Cited on page 37.

[173] I. Poggi. Towards the alphabet and the lexicon of gesture, gaze and touch. In *Virtual Symposium on Multimodality of Human Communication. http://www. semioticon. com/virtuals/index. html*, 2002. Cited on page 29.

[174] H. Pohl and M. Rohs. Around-device devices: My coffee mug is a volume dial. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices &#38; Services*, MobileHCI '14, pages 81–90, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3004-6. doi: 10.1145/2628363.2628401. URL `http://doi.acm.org/10.1145/2628363.2628401`. Cited on page 32.

[175] F. Quek, D. McNeill, R. Bryll, S. Duncan, X.-F. Ma, C. Kirbas, K. E. McCullough, and R. Ansari. Multimodal human discourse: Gesture and speech. *ACM Trans. Comput.-Hum. Interact.*, 9(3):171–193, Sept. 2002. ISSN 1073-0516. doi: 10.1145/568513.568514. URL `http://doi.acm.org/10.1145/568513.568514`. Cited on page 30.

[176] R. Rädle, H.-C. Jetter, M. Schreiner, Z. Lu, H. Reiterer, and Y. Rogers. Spatially-aware or spatially-agnostic?: Elicitation and evaluation of user-defined cross-device interactions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3913–3922, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702287. URL http://doi.acm.org/10.1145/2702123.2702287. Cited on pages 40, 41, 42, 107, 109, 114, and 139.

[177] M. Rahman, S. Gustafson, P. Irani, and S. Subramanian. Tilt techniques: Investigating the dexterity of wrist-based input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1943–1952, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518997. URL http://doi.acm.org/10.1145/1518701.1518997. Cited on pages 27 and 28.

[178] G. Ramos, K. Hinckley, A. Wilson, and R. Sarin. Synchronous gestures in multi-display environments. *HumanComputer Interaction*, 24(1-2):117–169, 2009. doi: 10.1080/07370020902739288. URL https://doi.org/10.1080/07370020902739288. Cited on pages 39 and 40.

[179] J. Rekimoto. Tilting operations for small screen interfaces. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, UIST '96, pages 167–168, New York, NY, USA, 1996. ACM. ISBN 0-89791-798-7. doi: 10.1145/237091.237115. URL http://doi.acm.org/10.1145/237091.237115. Cited on pages 2, 25, 46, 50, 70, and 81.

[180] J. Rekimoto. A multiple device approach for supporting whiteboard-based interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '98, pages 344–351, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-30987-4. doi: 10.1145/274644.274692. URL http://dx.doi.org/10.1145/274644.274692. Cited on page 37.

[181] J. Rekimoto. Gesturewrist and gesturepad: unobtrusive wearable interaction devices. In *Proceedings Fifth International Symposium on Wearable Computers*, pages 21–27, 2001. doi: 10.1109/ISWC.2001.962092. Cited on page 33.

[182] J. Rico and S. Brewster. Usable gestures for mobile interfaces: Evaluating social acceptability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 887–896, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753458. URL http://doi.acm.org/10.1145/1753326.1753458. Cited on pages 31 and 132.

[183] M. Romano, A. Bellucci, and I. Aedo. Understanding touch and motion gestures for blind people on mobile devices. In J. Abascal, S. Barbosa, M. Fetter, T. Gross, P. Palanque, and M. Winckler, editors, *Human-Computer Interaction – INTERACT 2015*, pages 38–46, Cham, 2015. Springer International Publishing. ISBN 978-3-319-22701-6. Cited on page 29.

[184] S. Roy Choudhary, M. R. Prasad, and A. Orso. X-pert: Accurate identification of cross-browser issues in web applications. In *Proceedings of the 2013 International*

Conference on Software Engineering, ICSE '13, pages 702–711, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4673-3076-3. URL `http://dl.acm.org/citation.cfm?id=2486788.2486881`. Cited on page 19.

[185] J. Ruiz and Y. Li. Doubleflip: A motion gesture delimiter for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2717–2720, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979341. URL `http://doi.acm.org/10.1145/1978942.1979341`. Cited on page 29.

[186] J. Ruiz, Y. Li, and E. Lank. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 197–206, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1978971. URL `http://doi.acm.org/10.1145/1978942.1978971`. Cited on pages 25, 26, and 27.

[187] K. Ryall, M. R. Morris, K. Everitt, C. Forlines, and C. Shen. Experiences with and observations of direct-touch tabletops. In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP '06)*, pages 8 pp.–, Jan 2006. doi: 10.1109/TABLETOP.2006.12. Cited on page 37.

[188] S. Santosa and D. Wigdor. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 63–72, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1770-2. doi: 10.1145/2493432.2493476. URL `http://doi.acm.org/10.1145/2493432.2493476`. Cited on pages 5 and 36.

[189] S. Santosa and D. Wigdor. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 63–72, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1770-2. doi: 10.1145/2493432.2493476. URL `http://doi.acm.org/10.1145/2493432.2493476`. Cited on page 36.

[190] N. Sawadsky and G. C. Murphy. Fishtail: From task context to source code examples. In *Proceedings of the 1st Workshop on Developing Tools As Plug-ins*, TOPI '11, pages 48–51, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0599-0. doi: 10.1145/1984708.1984722. URL `http://doi.acm.org/10.1145/1984708.1984722`. Cited on page 17.

[191] C. Schmidt, J. Müller, and G. Bailly. Screenfinity: Extending the perception area of content on very large public displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1719–1728, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466227. URL `http://doi.acm.org/10.1145/2470654.2466227`. Cited on page 40.

[192] D. Schmidt, J. Seifert, E. Rukzio, and H. Gellersen. A cross-device interaction style for mobiles and surfaces. In *Proceedings of the Designing Interactive Systems Conference*, DIS '12, pages 318–327, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1210-3. doi: 10.1145/2317956.2318005. URL `http://doi.acm.org/10.1145/2317956.2318005`. Cited on page 37.

[193] M. Schreiner, R. Rädle, H.-C. Jetter, and H. Reiterer. Connichiwa: A framework for cross-device web applications. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '15, pages 2163–2168, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3146-3. doi: 10.1145/2702613.2732909. URL `http://doi.acm.org/10.1145/2702613.2732909`. Cited on page 20.

[194] J. Seifert, A. Bayer, and E. Rukzio. Pointerphone: Using mobile phones for direct pointing interactions with remote displays. In P. Kotzé, G. Marsden, G. Lindgaard, J. Wesson, and M. Winckler, editors, *Human-Computer Interaction – INTERACT 2013*, pages 18–35, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40477-1. Cited on pages 37 and 39.

[195] T. Seyed, C. Burns, M. Costa Sousa, F. Maurer, and A. Tang. Eliciting usable gestures for multi-display environments. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '12, pages 41–50, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1209-7. doi: 10.1145/2396636.2396643. URL `http://doi.acm.org/10.1145/2396636.2396643`. Cited on pages 39 and 41.

[196] T. Seyed, A. Azazi, E. Chan, Y. Wang, and F. Maurer. Sod-toolkit: A toolkit for interactively prototyping and developing multi-sensor, multi-device environments. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, pages 171–180, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3899-8. doi: 10.1145/2817721.2817750. URL `http://doi.acm.org/10.1145/2817721.2817750`. Cited on page 21.

[197] X. Shen, P. Eades, S.-H. Hong, and A. V. Moere. Intrusive and non-intrusive evaluation of ambient displays. In *Ambient Information Systems*, 2007. Cited on page 36.

[198] S. Siddhpuria, K. Katsuragawa, J. R. Wallace, and E. Lank. Exploring at-your-side gestural interaction for ubiquitous environments. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, DIS '17, pages 1111–1122, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4922-2. doi: 10.1145/3064663.3064695. URL `http://doi.acm.org/10.1145/3064663.3064695`. Cited on page 39.

[199] B. Signer, A. de Spindler, and M. C. Norrie. A framework for link sharing in cooperative cross-media information spaces. In Y. Luo, editor, *Cooperative Design, Visualization, and Engineering*, pages 185–192, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04265-2. Cited on page 39.

[200] J. Song, G. Sörös, F. Pece, S. R. Fanello, S. Izadi, C. Keskin, and O. Hilliges. In-air gestures around unmodified mobile devices. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 319–329, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647373. URL `http://doi.acm.org/10.1145/2642918.2647373`. Cited on pages 5, 30, 31, and 32.

[201] S. Sridhar, A. M. Feit, C. Theobalt, and A. Oulasvirta. Investigating the dexterity of multi-finger input for mid-air text entry. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3643–3652, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702136. URL `http://doi.acm.org/10.1145/2702123.2702136`. Cited on page 30.

[202] J. Stasko, M. Doo, B. Dorn, and C. Plaue. Explorations and experiences with ambient information systems. In *Proceedings of the Workshop for Ambient Information Systems at the 5th International Conference on Pervasive Computing (PERVASIVE 2007)*, pages 36–41, 2007. Cited on page 36.

[203] S. Stellmach and R. Dachselt. Look &#38; touch: Gaze-supported target acquisition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2981–2990, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2208709. URL `http://doi.acm.org/10.1145/2207676.2208709`. Cited on pages 36 and 38.

[204] S. Stellmach, S. Stober, A. Nürnberger, and R. Dachselt. Designing gaze-supported multimodal interactions for the exploration of large image collections. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, NGCA '11, pages 1:1–1:8, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0680-5. doi: 10.1145/1983302.1983303. URL `http://doi.acm.org/10.1145/1983302.1983303`. Cited on page 38.

[205] R. J. Teather and I. S. MacKenzie. Position vs. velocity control for tilt-based interaction. In *Proceedings of Graphics Interface 2014*, GI '14, pages 51–58, Toronto, Ont., Canada, Canada, 2014. Canadian Information Processing Society. ISBN 978-1-4822-6003-8. URL `http://dl.acm.org/citation.cfm?id=2619648.2619658`. Cited on pages 6, 9, 27, 63, 68, 70, 81, 101, 103, 106, 135, and 139.

[206] M. Ten Koppel, G. Bailly, J. Müller, and R. Walter. Chained displays: Configurations of public displays can be used to influence actor-, audience-, and passer-by behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 317–326, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2207720. URL `http://doi.acm.org/10.1145/2207676.2207720`. Cited on page 37.

[207] C. Türk. Cross-tilt-and-tap: Design and development of tilting interactions in a cross-device environment. Bachelor's thesis, ETH Zurich, 2015. Cited on page 145.

[208] T. Vajk, P. Coulton, W. Bamford, and R. Edwards. Using a mobile phone as a wii-like controller for playing games on a large public display. *International Journal of Computer Games Technology*, 2008, 2008. doi: 10.1155/2008/539078. URL `http://dx.doi.org/10.1155/2008/539078`. Cited on page 38.

[209] B. van Tonder and J. Wesson. Is tilt interaction better than keypad interaction for mobile map-based applications? In *Proceedings of the 2010 Annual Research*

Conference of the South African Institute of Computer Scientists and Information Technologists, SAICSIT '10, pages 322–331, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-950-3. doi: 10.1145/1899503.1899539. URL http://doi.acm.org/10.1145/1899503.1899539. Cited on page 27.

[210] B. van Tonder and J. Wesson. Intellitilt: An enhanced tilt interaction technique for mobile map-based applications. In P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque, and M. Winckler, editors, *Human-Computer Interaction – INTERACT 2011*, pages 505–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-23771-3. Cited on page 27.

[211] D. Vogel and R. Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, UIST '05, pages 33–42, New York, NY, USA, 2005. ACM. ISBN 1-59593-271-2. doi: 10.1145/1095034.1095041. URL http://doi.acm.org/10.1145/1095034.1095041. Cited on page 39.

[212] W. S. Walmsley, W. X. Snelgrove, and K. N. Truong. Disambiguation of imprecise input with one-dimensional rotational text entry. *ACM Trans. Comput.-Hum. Interact.*, 21(1):4:1–4:40, Feb. 2014. ISSN 1073-0516. doi: 10.1145/2542544. URL http://doi.acm.org/10.1145/2542544. Cited on pages 25, 27, and 50.

[213] R. Walter, G. Bailly, and J. Müller. Strikeapose: Revealing mid-air gestures on public displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 841–850, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2470774. URL http://doi.acm.org/10.1145/2470654.2470774. Cited on pages 30 and 37.

[214] R. Walter, G. Bailly, N. Valkanova, and J. Müller. Cuenesics: Using mid-air gestures to select items on interactive public displays. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices &#38; Services*, MobileHCI '14, pages 299–308, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3004-6. doi: 10.1145/2628363.2628368. URL http://doi.acm.org/10.1145/2628363.2628368. Cited on page 37.

[215] M. Weigel, T. Lu, G. Bailly, A. Oulasvirta, C. Majidi, and J. Steimle. iskin: Flexible, stretchable and visually customizable on-body touch sensors for mobile computing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 2991–3000, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702391. URL http://doi.acm.org/10.1145/2702123.2702391. Cited on page 34.

[216] M. Weiser. The computer for the 21st century: Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence. In R. M. BAECKER, J. GRUDIN, W. A. BUXTON, and S. GREENBERG, editors, *Readings in HumanComputer Interaction (Second Edition)*, Interactive Technologies, pages 933 – 940. Morgan Kaufmann, second edition edition, 1995. ISBN 978-0-08-051574-8. doi: https://doi.org/10.1016/B978-0-08-051574-8.50097-2. URL https://www.

sciencedirect.com/science/article/pii/B9780080515748500972. Cited on page 36.

[217] D. Wigdor and R. Balakrishnan. Tilttext: Using tilt for text input to mobile phones. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pages 81–90, New York, NY, USA, 2003. ACM. ISBN 1-58113-636-6. doi: 10.1145/964696.964705. URL http://doi.acm.org/10.1145/964696.964705. Cited on pages 25, 27, and 135.

[218] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: 10.1145/1518701.1518866. URL http://doi.acm.org/10.1145/1518701.1518866. Cited on page 109.

[219] K. Wolf and N. Henze. Comparing pointing techniques for grasping hands on tablets. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices &#38; Services*, MobileHCI '14, pages 53–62, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3004-6. doi: 10.1145/2628363.2628371. URL http://doi.acm.org/10.1145/2628363.2628371. Cited on page 86.

[220] K. Yatani, K. Tamura, K. Hiroki, M. Sugimoto, and H. Hashizume. Toss-it: Intuitive information transfer techniques for mobile devices. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1881–1884, New York, NY, USA, 2005. ACM. ISBN 1-59593-002-7. doi: 10.1145/1056808.1057046. URL http://doi.acm.org/10.1145/1056808.1057046. Cited on pages 5, 40, and 107.

[221] H.-S. Yeo, X.-S. Phang, S. J. Castellucci, P. O. Kristensson, and A. Quigley. Investigating tilt-based gesture keyboard entry for single-handed text entry on large devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 4194–4202, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025520. URL http://doi.acm.org/10.1145/3025453.3025520. Cited on pages 25, 27, and 135.

[222] S. H. Yoon, Y. Zhang, K. Huo, and K. Ramani. Tring: Instant and customizable interactions with objects using an embedded magnet and a finger-worn device. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 169–181, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4189-9. doi: 10.1145/2984511.2984529. URL http://doi.acm.org/10.1145/2984511.2984529. Cited on page 33.

[223] S. Yousefi, F. A. Kondori, and H. Li. Experiencing real 3d gestural interaction with mobile devices. *Pattern Recognition Letters*, 34(8):912 – 921, 2013. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2013.02.004. URL http://www.sciencedirect.com/science/article/pii/S0167865513000482. Computer Analysis of Images and Patterns. Cited on pages 32 and 33.

[224] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, and M. Matera. A framework for rapid integration of presentation components. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 923–932, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242697. URL `http://doi.acm.org/10.1145/1242572.1242697`. Cited on page 21.

[225] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang. A framework for hand gesture recognition based on accelerometer and emg sensors. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(6):1064–1076, Nov 2011. ISSN 1083-4427. doi: 10.1109/TSMCA.2011.2116004. Cited on page 34.

Curriculum Vitae

Particulars

Name	Linda Di Geronimo
Date of Birth	June 7, 1989
Birthplace	Battipaglia, SA, Italy

Education

2008–2013	Study of Computer Science at Università degli Studi di Salerno (Italy)
2011	Computer Science Bachelor Degree *cum Laude* at Università degli Studi di Salerno (Italy)
2013	Exchange semester at the Globis group, ETH Zürich, Switzerland
2013	Computer Science Master Degree *cum Laude* at Università degli Studi di Salerno (Italy)
2013–2018	Doctoral studies in the Globis group at ETH Zürich, Switzerland

Work Experience (since 2013)

12/2013–09/2018	Research and teaching assistant with various groups at ETH Zürich, Switzerland
2014 – 2017	Teaching assistant and Main Teaching assistant for the Web Engineering Class at ETH Zürich, Switzerland
2015 – 2018	Teaching assistant and Main Teaching assistant for the HCI Class at ETH Zürich, Switzerland
2015 – 2018	Teaching assistant and Main Teaching assistant for the Mobile and Personal Information System Class at ETH Zürich, Switzerland
2014 – 2018	Teaching assistant for the Information System Lab at ETH Zürich, Switzerland
2014 – 2018	Teaching assistant for the CSCW Seminar at ETH Zürich, Switzerland