Technische Universität Berlin
Telekom Innovation Laboratories
Quality and Usability Lab
Fakultät 4

Master of Science Thesis

Spatial Input for 2D Peephole Navigation on Mobile Devices

Martin Schüßler
Boddinstr. 17a, 12053 Berlin, Deutschland
schuesslerm@acm.org

31.10.2016

Aufgabensteller:
Prof. Dr. Sebastian Möller
Tel 4

Ernst-Reuter-Platz 7
10587 Berlin
Sebastian.Moeller@telekom.de

Mitberichter & Betreuer:
Prof. Dr. Jörg Müller
Tel 18

Ernst-Reuter-Platz 7
10587 Berlin
joerg.mueller@acm.org

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbststandig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, 31.10.2016

. . . . . .

(Unterschrift Martin Schüßler)

Abstract

This thesis works towards establishing spatial movement as an input modality for mobile devices. The reader is introduced to related work of multiscale navigation with state of the art multi-touch gestures and spatially aware peephole displays. Motivated by several studies that showed the potential of 2D navigation using spatial movement of a device instead of touch, three informal lab studies were conducted. The results of the first study suggest that users are able to navigate even faster with peephole displays if their physical movements are amplified by a linear gain factor. In the second study an approach that allowed users to separately translate (pan) and scale (zoom) was found to jeopardise the usefulness of spatial input. In the third study participants used a peephole display to move otherwise unreachable items closer to their thumb to tap them without using their second hand. The data gathered in the latter study provided hints that peephole displays may perform equally well but not better than state of the art touch-based techniques for this type of task. Having gained insights from these studies and other works, it is argued that the potential of peephole displays lies in their multiscale navigation performance, their complimentary nature to other inputs and their single handed operability. In order to effectively exploit this potential, the author advises to use dynamic spatial orientation mapping, a clutching mechanism and fast, accurate six degree of freedom spatial tracking. As a technical contribution the most advanced version of a peephole display prototype is presented. It incorporates these insights, allowing single handed navigation via touch and spatial movement separated by a pressure based clutching mechanism.

Contents

1 Navigation using multi-touch and spatially aware mobile displays

This chapter provides the reader with a state of the art fundamental understanding of virtual 2D navigation with spatially aware mobile displays and multi-touch mobile displays as their baseline companions. In Section 1.1 the reader is introduced to **navigation in electronic worlds** on a meta level. The fundamental concepts of **translation and scaling** are formally described. Thereafter combined use both in **multiscale navigation** and in different categories of **multiscale interfaces** is shown. In Section 1.2 selected **multi-touch gestures** for navigation on mobile displays that extend the well established pinch-drag-flick baseline are presented. In Section 1.3 **dynamic peephole displays** (DP) and **zooming lenses** (ZL) are defined and related work is introduced. Additionally, a notation is derived that helps the reader to understand the different types of these displays and lenses. In the remainder of this chapter several **user studies** that have evaluated navigation performance on spatially aware mobile displays are presented.

1.1 Navigation in 2D electronic worlds

Whenever digital content exceeds the resolution of the displaying device, users need to decide what **section** of the information space they want to display [7, 6, 13].

> "In most computer applications, users need to interact with more information and with more interface components than can be conveniently displayed at one time on a single screen" [7]

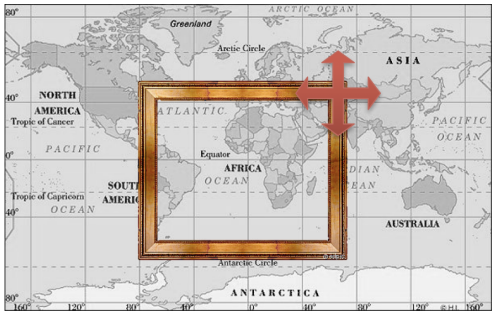
This is why **navigation** in electronic worlds is a very important topic of human-computer interaction (especially for mobile devices with their limited screen size) [13]. Commonly navigation is a primary task or requirement of another tasks and therefore a secondary task. **Navigation techniques** are mechanisms that facilitate navigation by allowing the user to steer the **traversal** [26] from one section to another. While

doing so, users might be **searching** for a known target or exploring and memorising the information by **browsing** through it[26]. Navigation can also be understood as a **feedback loop**: Users may have a certain objective such as *"I want to find the login button and tap on it with my finger"*. They try to achieve this objective by issuing commands using a navigation technique. At the same time they have to make **decisions in real time** [26]: Firstly, whether they issued the right commands (*"Oh I scrolled down instead of up"*), secondly, whether they have to adapt their strategy ("I found the button but it's too small to be taped, I need to zoom in further"), and thirdly, whether the goal has been reached ("The button is big enough now to be taped conveniently").

There are many other complex physiological aspects of "the cognitive process of determining and following a path" [27] but to mention them all is beyond the scope of this introduction.

1.1.1 Translation

> "The VDT [Visual Display Terminal] user views a representation of an area of computer memory. In most cases the portion of memory the user wishes to examine is much larger than that which will fit on the screen at one time. For this reason almost all VDT's are equipped with some sort of *'scroll function'* that allows the user to display data that is located beyond the limits of the screen." - Bury et al. [6], 1982



(a) Dynamic peephole ($p = -1$): user interaction (arrows) manipulates the position of viewport (frame)

(b) Static peephole ($p = +1$): user interaction manipulates the position of the content (world map)

Figure 1.1: Dynamic and static peephole control metaphors for translation

In early systems 2D navigation was limited to *scrolling*, *windowing* or *paging*. These

mechanisms substitute currently visible content with neighbouring invisible content using a **translation** $t_{xy} = (t_x, t_y)$. They allow the user to "either **move a fixed viewport** over the document, or **move the document** under the viewport" [13]. The first mentioned mapping is depicted in Figure 1.1(a). It uses a control metaphor that is also called *push-view* [22], *view-in-hand* [41], *scene-in-hand* [19] or **dynamic peephole** [60]. The second control metaphor shown in Figure 1.1(b) is referred to as *push-background* [22], *document-in-hand* [41] or **static peephole** [36]. There have been many studies investigating the superiority of one metaphor over the other (e.g., [22, 36]). However the author shares the opinion of Pahud et al. [41] that none of them is necessarily the best.

Whenever translation or any other navigation occurs, graphical hardware and software is used to compute how the visual output needs to be adapted. From a computer graphics point of view virtual worlds are commonly represented using vertices. These vertices are mainly described by vectors. One way of transforming these vectors to represent the new state of the system is to multiply them with affine transformations matrices (T). A translation can be described using such an affine transformation matrix and a metaphor factor $p \in [-1, +1]$ (see Fig. 1.1).

$$T_{t_{xy}} = \begin{pmatrix} 1 & 0 & p * t_x \\ 0 & 1 & p * t_y \\ 0 & 0 & 1 \end{pmatrix}$$

1.1.2 Scaling

If documents become very large, a lot of translation may be required [13]. This introduces a discontinuity between the information displayed at different times and places, which can cause cognitive and mechanical burdens for users [7]. A *zoom out* decreases the scale causing a content contraction. This results in more information being visible but with less detail, which is desirable for getting an *overview* [7]. Accordingly, a *zoom in* increases the scale resulting in a content expansion. Less information is visible but at greater detail, which is preferable if the user wants to target or *focus*[7] on certain objects within the magnified section.

The scale is defined by factors s_x and s_y and for uniform scaling both are equal and can be referred to as s_{xy}. Basic scaling happens around the coordinate origin, which is undesired in most cases. The matrix shows a conjugation of translation and scaling which implements scaling around a specific *zoom centre* $c = (c_x, c_y)$, also called expansion/contraction focus, anchor point or zoom pivot. This point remains stationary

while content around it is expanded or contracted.

$$T_{s_x,s_y,c} = \begin{pmatrix} s_x & 0 & c_x - s_x c_x \\ 0 & s_y & c_y - s_y c_y \\ 0 & 0 & 1 \end{pmatrix}$$

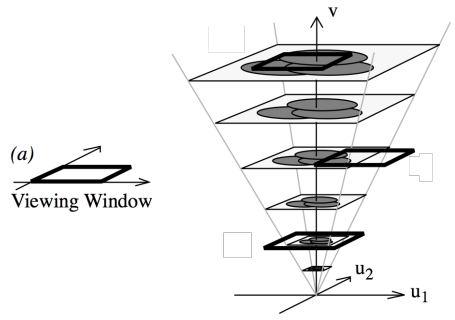
1.1.3 Rotation

In the literature rotation is usually not considered when talking about 2D navigation. However just like scaling and translation it can be described as a similarity transformation that allows the user to reveal content that was previously not visible. Some applications such as document viewers and maps have started to use orientation in a meaningful way. Similarly to scaling, a content rotation by an angle θ around a specific (anchor) point c can be achieved using a conjunction of translation and rotation:

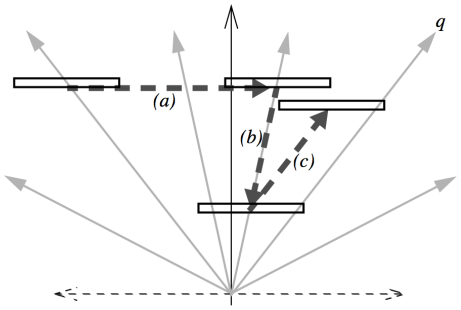
$$T_{\theta,c} = \begin{pmatrix} \cos\theta & -\sin\theta & -\cos\theta c_x + c_x + c_y \sin\theta \\ \sin\theta & \cos\theta & -\cos\theta c_y + c_y - c_x \sin\theta \\ 0 & 0 & 1 \end{pmatrix}$$

1.1.4 Space scale diagram

A visual and formal concept that explains the relationship and possible combination of translation and scaling is the **space scale diagram** [11], hereafter *SSD*. The concept is visualised in Figure 1.2.



(a) In the SSD content is replicated in each u_1u_2-plane according to the scale value of the v axis whereas the size of the users viewing window *(a)* remains constant in all v-levels.



(b) Examples of movements of the viewing window within the SSD: *(a)* translation, *(b)* scaling and *(c)* simultaneous translation and scaling.

Figure 1.2: The Space Scale Diagram [11].

In the SSD content shown on the users display is described by the position of the viewing window $(u_1, u_2) = (s_{xy} * x, s_{xy} * y)$ and scale $v = f_z(s_{xy})$, where (x, y) is the centre of the visible section (on the original unscaled picture) and f_z is a zooming function, e.g., $f_z(s_{xy}) = s_{xy}$ (linear zoom) or $f_z(s_{xy}) = log(s_{xy})$ (logarithmic zoom). Figure 1.2(b) shows that translation and scaling can be described as a 3D movement $(\Delta u_1, \Delta u_2, \Delta s_{xy})$ of the viewing window (u_1, u_2, s_{xy}) inside of the SSD. When describing such movements, it is important to point out that a zoom Δs_{xy} is mapped as a scalar to a vector that transfixes the u_1u_2-plane at each level v at the zoom centre $c_v = (c_x * v, c_y * v)$. This means the content shown after scaling does not only depend on Δs_{xy} but also on c_v, which is in line with the definition of scaling in Section 1.1.2. An example for this can be seen in Figure 1.2(b): The result of movement *(c)*, which is a simultaneous scaling and translation, could have also been achieved by a pure zoom with zoom centre q. The SSD could easily be extended to not only take scaling and translation but also rotation into account. The location of the viewing window would only need to be extended with information about its rotation (u_1, u_2, v, θ). Similarly to the zooming movement, rotation would anchor itself "around" c_v.

Another contribution of the SSD is that it helps to understand that translation and scaling influence one another [11, 24, 7]. For example: The shortest route between two points in multiscale navigation often involves heavy scaling to avoid extensive translation. It is simply much easier to cover larger distance when content is scaled to be much smaller. The SSD also helps to understand one of the many reasons why the display size has a noticeable influence on the number of zooms and translations [24] as well as an impact on user performance in search tasks [46, 54, 40]: With a larger display points of interests become visible after a shorter movement along the v-axis, resulting in a shorter overall path length within the SSD. Furthermore, it can been seen that the shortest direct path between any two points on different v-levels can only be achieved when allowing the user to freely choose a zoom centre or to scale and translate simultaneously (Fig. 1.2(b) *(c)* and q).

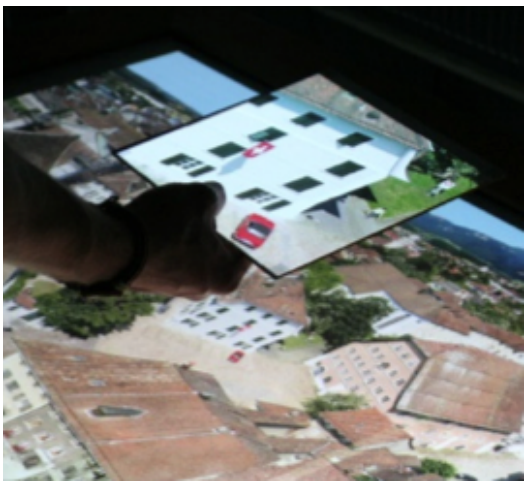
Whenever investigating the effectiveness of navigation techniques, the SSD can be used to plot the movements, providing a visual understanding of the navigation phenomena at question. The movement vectors of the SSD also come in very handy when mapping spatial movement of a physical display to virtual movement of the viewing window. This will be shown in more detail in Section 4.5.

1.1.5 Multiscale interfaces

The combination of translation and scaling (*panning* and *zooming*) is implemented by a large number of **multiscale interfaces** [3]. Several of them are reviewed by Cockburn et al. [7], who categorised them into four approaches. **Zooming systems** utilise a temporal segregation of focused and contextual views. That means, rather than seeing both views simultaneously, users zoom in to focus and zoom out for context. **Overview+detail systems** use concurrent focus and context views that are spatially segregated. **Focus+context Systems** provide focus and context within a single display using a variety of techniques, including selectively removing, diminishing, or magnifying items, and various displays supporting a wide field of view. **Cue-based systems** modify the display of items to highlight or suppress them, dependent on context or search criteria. Examples of all four categories are displayed in Figure 1.3.

In reality many systems use a combination of these techniques. In this thesis the scope is mainly limited to zooming systems. However, this does not necessarily mean that findings and conclusions of this work do not apply to other multiscale interfaces.

(a) A **zooming system**: This spatially aware display [53] can be moved closer or further away from the observer to zoom out or zoom in.



(b) An **overview+detail system**: This Magic Lens [4] called Paper Lens[51] acts similar to the spatially aware display but the entire information space (overview) is displayed behind it on the tabletop.

(c) A **focus+context System**: This spatial display uses user-centric spherical mapping to change the highlighted section of its bifocal visualisation [43].

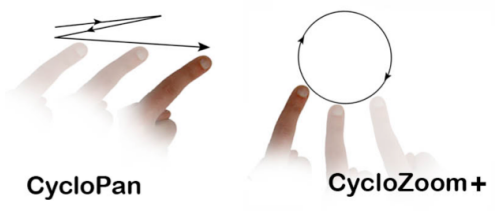
(d) A **cue-based systems**: The position and radius of the circle at displays boundary indicate the direction and distance of point of interests [2].

Figure 1.3: Examples of multiscale interfaces [7].

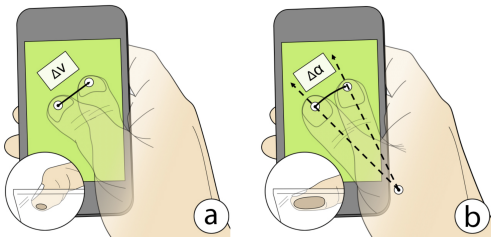
1.2 Multi-touch input for navigation on mobile displays

On mobile devices with their limited screen size multi-touch is currently the most widely used input modality. In direct touch the **drag gesture** is often used for translation by continuously mapping the change of the fingertip position (in pixels) to t_{xy}. The **flick gesture** extends the drag gesture with inertial scrolling. This means that Δt_{xy} is not immediately set to 0 when the finger has been lifted. Instead it starts declining over time and the content keeps scrolling for a while. The velocity of the previous flicking movement determines the rate of deceleration. Additionally, the movement can be stopped immediately with a tap. This extension allows users to rapidly cover long distances.

For zooming the **pinch gesture** is often utilised. Here the mid-point between two fingers defines c and the change in distance between them is mapped to s_{xy}. This is done in such a way that the single pixels below the fingertips remain under the tip, creating the impression that content is manipulated directly in place and *stretched* like a rubber band, which serves well as a metaphor [34, pp. 106-108]. While intuitive this mapping requires users to use both hands. Furthermore, as virtual travel distances per gesture are short [35], users may have to repeat gestures several times [54].



(a) Cyclo Pan and CycloZoom+[35] use cyclic gestures to extend travel distances.



(b) FatThumb [5] uses the contact size of a drag gesture to discriminate between scaling and translation.

Figure 1.4: Single handed muli-touch gestures for navigation.

Extensions to these mappings, which recognise cyclic gestures, are depicted in Figure 1.4(a): **Cyclopan** [35] acts similar to the standard gesture but continues to pan in the initial direction when the stroke direction of the finger is reversed. The result is a cyclic rubbing movement, which allows the user to cover larger distances with one continuous gesture. Here t_{xy} is defined by the distance a between the start point T_1 and point T_2 where movement is reversed as well as a gain factor g which is based on the frequency of this cyclic movement. **CycloZoom+** [35] allows users to zoom

by drawing an approximately circular ellipse. The angular velocity is mapped to s_{xy} in such a way that a clockwise circulation corresponds to a zoom-in and a counter clockwise to a zoom-out. Users can control the zooming speed by varying the velocity or the radius of the ellipse. They can also control the zoom centre c as it is defined by the centre of the ellipse. Unlike the pinch gesture the movement can be executed using only the thumb of the holding hand.

Another technique that allows for singled-handed zooming is **FatThumb** [5] (Fig. 1.4(b)), which uses the contact size of the thumb to discriminate between two modes: When detecting a small contact size, the movement of the thumb is mapped to t_{xy}, while when detecting a bigger contact size, the angular movement around the CMC joint[1] is mapped to s_{xy} in such a way that a clockwise movement is zooming in. Additionally, the contact size linearly determines the gain factor. The centre of zoom c is defined by the point where the first *"fat"* touch of the current gesture was detected.

All of the just described techniques allow for one-handed interaction. But on the downside they rely very heavily on fine motor skills [54]. Furthermore they may suffer from occlusion [58].

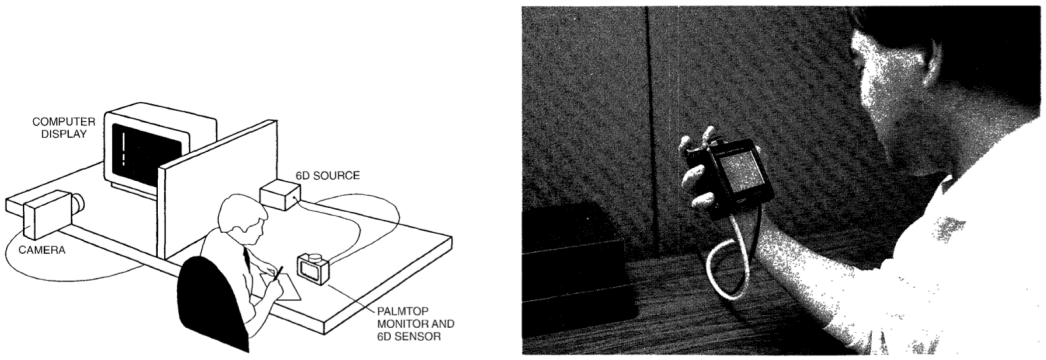
1.3 Spatially navigating displays: Peephole displays and Magic Lenses

An alternative for touch input is spatial input, which benefits from a **larger interactive area and motor space**. Hinckley et al. [19] presented a survey about "interfaces based upon free space 3D input technology" that provides many insights about spatial input. For example it is argued that this input modality can take advantage of the **proprioceptive senses**. Moreover, it is pointed out that the commonly used metaphor of moving a camera or peephole is based on real life experience. Also the importance of a well-designed clutching mechanism is stressed. Furthermore, it is suggested to use a mapping to relative motion rather than an absolute coordinate frame so that cognitive load and fatigue are reduced.

However, this thesis is not concerned with spatial input in general but rather with *spatially navigating dynamic peephole displays*. As this long name implies, there are many terms that are used in the relevant related work that do not necessarily have the same meaning among all publications and can be easily confused. For this reason, some terms will be defined in the scope of this work and identified with an abbreviation. As for spatial display:

[1] The third joint, counting from the tip of the thumb

A **spatial display** (or spatially aware display), ***SD***, is a display that uses its spatial movement as an input for interaction. Its movement is defined by its change of position and/or rotation. That is a position change in centimetres, millimetres or even sub-millimetres rather than meters and kilometres and a rotation change in degrees.



(a) The 6 DoF position of the tangible display is transmitted to a workstation that is capable of processing the movement in real time.

(b) The workstation updates its display according to the movement of the handheld. This updated section of the information space is transmitted to the handheld using a video camera.

Figure 1.5: Chameleon prototyp [10].

The first published SD is the **Chameleon** prototype presented by Fitzmaurice [10] in 1993. It is a handheld device that uses its orientation and position as input upon pressing a small button (see Fig. 1.5). In one case it was used to navigate a 3D workspace. This leads to the next definition:

A dynamic **peephole display**, ***PD*** is an SD using the dynamic peephole metaphor (Sec. 1.1.1) and its movement detection capabilities for virtual navigation.

Every SD setup introduced in this section will be accompanied by a notation that provides information about what properties of the physical world are mapped to the virtual world. For the Chameleon the notation is PD($m^*_{xyz}, r^*_{xyz} : m_{xyz}, r_{xyz}$) as physical movement of the PD along each axis (m_{xyz}) and its rotation around each axis (r_{xyz}) are mapped to (:) the movement (m_{xyz}) and orientation (r_{xyz}) in the virtual space. Furthermore, the $*$-notation indicates that these properties are only processed upon the users wish, in this case explicitly by pressing a button. The Chameleon was later

extended to support a variety of inputs and mounted to a mechanical boom that is tracked in 3D space[56].

Another important concept are **Magic Lenses** [4], which are introduced the same year as the original Chameleon. Among other features these lenses are displaying specific information depending on their position towards a reference plane or object (absolute mapping). That means they exist within an environment which provides visual context. Therefore Magic Lenses can be categorised as *Overview+Detail* system (Sec 1.1.5). The first physical implementation of such a system is metaDesk presented by Ullmer and Ishii [57] in 1997. Here a tabletop was used as visual context. If a Magic Lens can be used for multiscale navigation, it is also sometimes referred to as Zooming Lens.

> A physical *Zooming Lens*, ***ZL*** is an SD that exist within an environment that provides visual context used for virtual navigation.

The Magic Lens used in Ullmer's setup was called Active Lens. It is a ZL ($m_{xyz}, r_{xyz} : m_{xyz}, r_{xyz}$) . A similar but much more advanced system is the PaperLens system presented by Spindler et al. [51] in 2009 (Fig. 1.3(b) on page 7). There is quite a number of systems that can be categorised as ZLs. Some of them use projections (e.g., [30]), whereas others use real world objects as visual context (e.g., [48]). The suitable item density of the visual context has been studied by Rohs et al. [49].

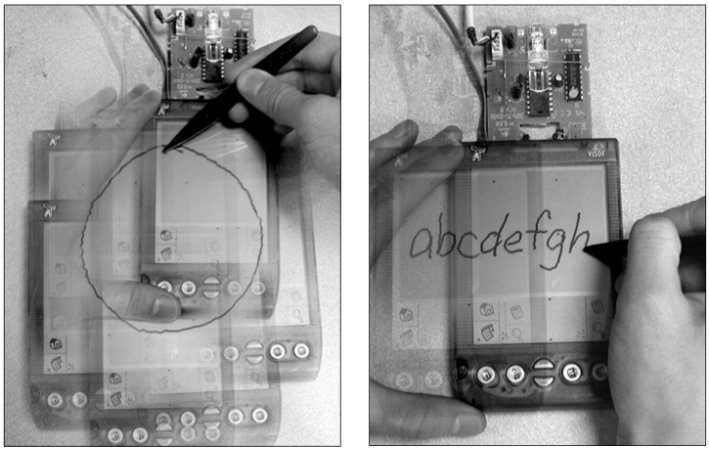


Figure 1.6: Peephole displays can support simultaneous annotation and 2D navigation. Here the user moving display to be able to draw/write things that are too large to fit the screen.

The use of an SD for 2D navigation was demonstrated by Yee et al. [60]. With their first prototype they mapped its movement in the plane to translation in the SSD,

making it a PD ($m_{xy} : m_{u_1u_2}$). A later prototype even mapped the z-Axis of the device to the v-Axis of the SSD, allowing to zoom content by moving the handheld inward or outward, making this one a PD($m_{xyz} : m_{u_1u_2v}$) and the first to be used for mutliscale navigation. Furthermore, Yee et al. also demonstrated the possibility of using the dominant hand for annotation while navigating (Fig. 1.6)

1.4 User navigation performance with spatial displays

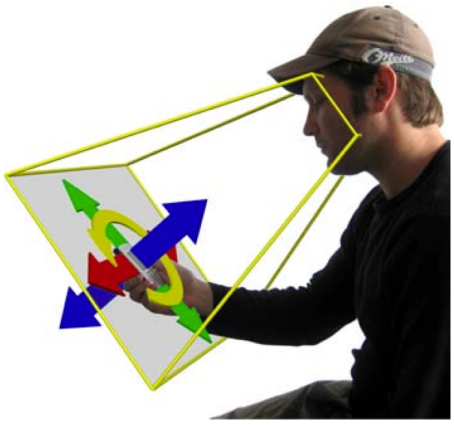
Several user studies have been conducted investigating navigation on PDs and ZLs. However, one should exercise caution when drawing conclusions from them, as they vary in study design (task etc.) and technical implementation (type of device, tracking system etc.) Additionally, gender [8] and display size [47] have an influence on navigation performance [54]. One study [40] even investigated possible sizes and aspect ratios of ZLs [40]. Furthermore, it has to be pointed out again that a ZL is an *Overview+detail system* whereas a PD is usually a *Zooming System* [19]. This means findings for a ZL cannot necessarily be transferred to a PD and vice versa. As shown in Section 1.1.4 navigation is much richer but also more complex when scaling is allowed. For this reason, the author argues that results from studies limiting navigation to a single scale by allowing only translation cannot be compared with those allowing multiscale navigation. This is why they are reviewed separately in the following sections.

1.4.1 Singlescale navigation studies

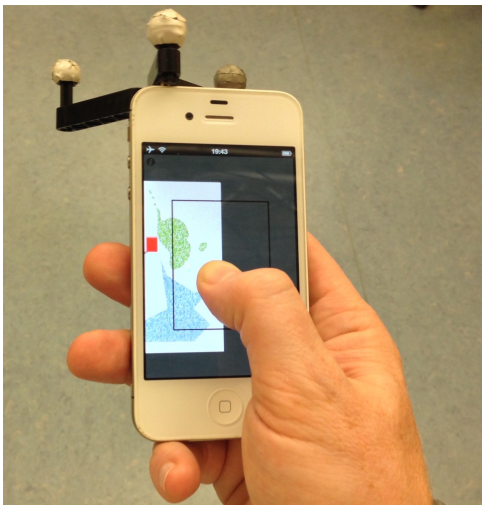
In one study Yee [60] tested his PD ($m_{xy} : m_{u_1u_2}$) against a stylus. Participants (N=24) performed a map viewing task as well a drawing task with both interaction techniques. Yee found that users finished the drawing task 32% faster with the PD. For the map task no difference in performance was found. In a study of Kerber et al. [32] participants conducted a navigation task using a wrist-worn smart watch that either worked with spatial input (PD ($m_{xy} : m_{u_1u_2}$)) vs touch input. They found the users to be significantly slower with the PD(12%), while nevertheless 8 out of 12 participants favoured spatial input. In yet another study of Hasan et al. [17] a PD ($m_{xy} : m_{u_1u_2}$) outperformed the touch based approach in a combined pointing and navigation task. Here subjects were 21,69% faster in comparison to touch. Pelurson and Nigay [43] conducted an experiment with 18 participants evaluating combined modalities for selection in bifocal views on mobile devices. They found performance with a PD ($r_y : u_1$) as depicted in Figure 1.3(c) to be slower than direct touch in focus-targeting phase, which resembles a navigation task.

1.4.2 Multiscale navigation studies

One study investigating how a PD performs in a multiscale navigation task was completed by Hansen et al. [15]. In Figure 1.7(a) it is depicted how they mapped spatial input to navigation (such a mapping is commonly used among PDs). Hansen et al. found no difference in performance between their PD ($m_{xyz} : m_{u_1u_2v}$) and a baseline joystick condition. However, most participants (12 out of 15) preferred the PD over the joystick. In another study with 18 participants of Rohs et al. [50] a PD ($m_{xyz} : m_{u_1u_2v}$) as well as a ZL ($m_{xyz} : m_{u_1u_2v}$) outperformed the same baseline (joystick). In a study of Kaufmann and Ahlström [31] pico projectors were used to display the context of a ZL ($m_{xyz} : m_{u_1u_2v}$). In their study the ZL showed equal navigation performance as Pinch-Drag-Flick (but excelled at spatial memory and location recall). Furthermore, Raedle et al. [44] combined a tablet with a wall-sized display and found that participants were 34% faster and showed 47% shorter path length with a ZL ($m_{xyz} : m_{u_1u_2v}$) versus touch.



(a) Illustration from Hansen et al. [14] showing multiscale navigation with PDs: Translation is achieved by moving the device along its xy-axis (red, green), while moving device along its z-axis (blue) initiates scaling.



(b) An optically tracked iPhone 4S running the application used in the navigation study [54] co-authored by the author. Users had to move 120 red rectangles inside the black frame. Half of the target were not initially visible (search tasks).

Figure 1.7: Studies evaluating multiscale PD navigation performance.

Two recent studies **compared PDs with standard Pinch-Drag-Flick**. One of them is a study of Spindler et al. [54]. The author of this thesis was part of the

research team and as such developed the PD prototype and conducted all experiments. In this study all 40 participants were significantly faster (on average more than 35%) using a PD ($m^*_{xyz}r^*_{xyz} : m_{u_1u_2}v$) in comparison to touch. The task of the study is briefly illustrated in Figure 1.7(b). In contrast to that, in a study of Pahud et al. [41] all participants (N=8) were significantly slower with a PD ($m_{xyz}r_{xyz} : m_{u_1u_2}v$). It is suggested that the contradicting results are due to differing design decisions and the use of different technologies, which will be covered in more detail in Chapter 3.

The study of Spindler et al. [54] was the first of its kind to provide hard statistical evidence that spatial input can outperform touch input in multiscale navigation. The results motivated the experiments presented in the next chapter.

2 Small pilot studies

This chapter presents three pilot studies that were conducted to improve an existing PD prototype and to identify topics for future large scale studies. While conducting the experiments of the before-mentioned navigation study (previous section), the author made several observations that led to the suggestion that the PD prototype could be improved even further. The prototype was also presented in an interactivity at CHI 2014 [53] (Fig. 4.1(a)) and show cased at ITS 2014, where it was tried and discussed by several researchers and expert users. The study observations as well as the discussions motivated the following studies. The first study explores the use of gain factors for PDs. The second study introduces constrains to spatial interaction in order to combat unintended translation and scaling. The last study explores whether PD can help users to point on initially unreachable items.

2.1 Even faster: CD gain for peephole displays

The direct manipulation metaphor of Pinch-Drag-Flick exploits the user's knowledge about the physical world by using a *1:1* control display gain [34, p. 106], meaning a 3 cm drag will translate content on the display by 3 cm. The same CD gain is used for ZLs. For this reason, it was also used for PD translation in the navigation study [54] (implementation details: Section 4.4.1). **The initial assumption was that a different CD gain would break the dynamic peephole metaphor**. Surprisingly, several participants demanded for a faster mapping when being questioned about possible improvements in a semi-structured interview [54]. This was true even though all participants finished the tasks considerably faster with the PD than with touch. The same demand for a gain was raised by some expert users that participated in the interactivity.

2.1.1 Evaluation of possible CD gains

A new version of the prototype was developed that allowed the user to change the CD gain for translation on the fly. Four members of the lab, all male, where invited

to try the new prototype independently from one another and in a standing position. Subjects were confronted with the same task used in the navigation study (Figure 1.7(b)), which was basically moving a red rectangle inside a black frame. In a training round they were asked to complete roughly 20 trails with the original *1:1* CD gain. Subsequently they were presented with different mappings and asked to complete at least 15 trails for each. The CD gains evaluated were: *2:3, 1:2, 1:3, 2:1* (in that order). In the last round they were allowed to change the gain at their own will to any value. **All participants choose gains of at least *1:2*, whereby two even chose *2:5* as their preference**.

2.1.2 Discussion of CD gain for peephole displays

Of course these results are not reliable as N=4 is simply too small and the study design did not account for all factors of influence. For example, results are likely to be different in a sitting position. But this observation provides hints that faster CD gains are indeed suitable for navigation tasks that require less precision.

Humans are very accustomed to manipulate the position of physical objects in 3D space and their fine motor skills allow them to do so with remarkable accuracy [19, 55]. But even though Spindler et al. [54] found the PD to be less fatiguing than the Pinch-Drag-Flick condition, extensive spatial movement over a prolonged time may very likely cause exhaustion, which is also referred to as the *gorilla arm effect*. Additionally, another concern raised is that humans in general rather avoid extensive physical movements and *"tend to move their hands in a surprisingly small working volume"* [19]. This is especially true when interacting in public due to reasons of **social acceptance**. A lower CD gain could provide a solution for these issues as smaller physical movements would result into considerably larger movements within the SSD. Consequently, the average physical path length could be shorter causing **less exhaustion**. If users are willing to use their entire arm length, they are able to navigate further than with a higher (slower) CD gain factor. If clutching[1] is supported its use may be be necessary less often, as **longer distances can be covered with single movement**. However, choosing a CD gain leads to some trade-offs as accuracy is reduced and users may also suffer from negative effects such as overshooting [34, pp. 80-81]. In this study, Participants mentioned that very fast CD gains were hard to

[1] Clutching in this context means that users can move the device without causing any effect. Similar to a clutch of a car that decouples the running engine from the transmission, it allows users to *decouple* spatial movement from the spatial interaction processing unit (by, e.g., pressing a button). Hence, they can reposition their hands and arms, avoiding uncomfortable positions.

control and were therefore perceived as stressful. Furthermore, with lower CD gains tracking issues are of greater severity and optical flow is increased, which is linked to experiencing higher mental load. Subjects were also asked whether with their favourite mapping intuitiveness and ease of use where negatively influenced. This was denied in all cases.

As a conclusion of this study it can be said that **CD gains other than *1:1* can be chosen without breaking the metaphor**. For this reason, the author suggests to allow users to choose their own CD gain, as it is usually done with mouse speed and mouse acceleration. Whether a faster mapping can be applied to ZLs, remains highly questionable as they still have to be aligned with their visual context, which usually remains stationary. **Future work** could provide empirical evidence which CD gains for translation and scaling are suitable for different types of navigation and pointing tasks requiring different levels of accuracy. An even more interesting topic are non-linear CD gains [34, pp. 80-81] and wheter they can be employed without breaking the metaphor. Especially mappings that employ velocity concepts similar to the flicking movement used in multi-touch could be promising, as also suggested by Pahud et al.[41].

2.2 Constrains for peephole displays

As pointed out in Section 1.1.4 the navigation mechanisms scaling and translation are strongly related. This is why for PDs they are usually mapped to the location changes of the device, which is perceived as very natural and intuitive mapping. But spatial displays use up to six degrees of freedom for navigation. This introduces a disadvantage as for spatial input users are hardly able to manipulate a single degree of freedom independently. [19, 55, 41]. As a consequence, while scaling with a PD an unintended translation is introduced and vice versa. Such phenomena have been observed for other input devices as well. For example, users find it very difficult to draw a straight line using a mouse. They also may accidentally rotate content when executing a pinch gesture on a multi-touch device. A common practise to solve this problem is the use of constrains. For example, in the real world a line is drawn with the help of a ruler that constrains the movement of the pen. Similar constrains are put in place in the virtual world. Some interfaces constrain the movements of objects to a single axis as long as the user is pressing a certain key. Another example is the sole rotation of objects around a specific axis using dedicated handles displayed next to them [55].

2.2.1 Double clutch prototype

Motivated by these approaches a new version of the prototype was developed that allowed the user to explicitly activate and deactivate movement filters. These filters could be set so that relative motion along a particular axis of the device is either passed through or filtered out (technical details can be found in Section 4.5). Hence, it became possible to execute scaling without translation and vice versa. The original prototype had a single touch-sensitive clutch that had to be explicitly activated by touching the surface of the display anywhere. This means once the user had placed his/her finger (usually a thumb) on the display, the movement and change in rotation of the display were interpreted as input. As soon as the finger was lifted, the motion processing was halted. This design was reused and slightly adapted. For the new prototype the touch sensitive area of an iPad Air 2 was vertically split in two halves. This was done because users usually hold the tablet with both hands and can tap either half using one of their thumbs. Now if the area on the left half of the display was touched, spatial processing began but with a scaling-only filter in place. If the area on the right half was touched, processing began with a translation-only filter. If both sides were touched simultaneously, no movements were filtered out allowing simultaneous scaling and translation as usual. If no area was touched, processing was halted.

2.2.2 Testing the double clutch prototype

Once again four members of the lab, all male, tested the new prototype independently from one another. They were introduced to the new clutch design and asked to complete several trials of the same task used in the previous study [54]. All participants were not in favour of the new design and found themselves pressing both clutches simultaneously to return to the old functionality. Furthermore, they confused the two different clutches and found it hard to remember what side is assigned to which mechanism. So, as a result, separating scaling and translation wasn't of use for handling the task.
Moreover, the participants reported that the metaphor of the peephole seemed to have broken, especially for scaling.

2.2.3 Discussion

This approach to constrains in peephole navigation wasn't successful. Not only was the clutch design found to be confusing but simply isolating movements seemed to have broken the *integral mapping* and the metaphor. These results are similar to findings by Jones et al. [23], where spatial movement of a finger around the display was used for

navigation. One prototype (*"1B Sim"*) was equipped with a single clutch and allowed simultaneous translation and scaling. It showed comparable performance to the touch base line and was significantly faster than another prototype (*"2B"*), which had two separate clutches for translation and scaling.

An alternative approach to constrains in spatial input could be the use of physical constrains (e.g., force feedback) instead of virtual ones, as they preserve the metaphor and are known to impose less cognitive load [19]. However, this leaves the question how to physically constrain the movement of the display. Another option might be to visually communicate the virtual constrains that are in place for example by straight grid lines for translation and expanding or contracting rectangles for zoom. This could help the user to understand which mode they are in. Yet another approach to constrains could be to detect the intention of the user. For example, if the movement pattern of the display suggests that the user is trying to translate rather than scale, movement along the z-axis could be ignored.

2.3 Pointing on phablets with dynamic peephole displays

The mobile context introduces many use case where single handed interaction is desirable or even required [28]. So it is no surprise that in a field observation made by Hoober [20] in 2013 49% of users were holding and touching their device with one hand only. However, recently larger devices, sometimes also referred to as phablets[2], have become popular. They face issues caused by the users limited thumb length, which makes only a sub region of the display available for single handed direct touch pointing as illustrated in Figure 2.1(a) on page 22. For the remaining regions two handed interaction or awkward hand positions are required. For this reason, Apple introduced a feature called *Reachability*: After a double tap on the home button, the entire content of the screen slides half way down. However, this technique does not satisfactory solve the problem as demonstrated in Figure 2.1(b)).

If in contrast a phablet device would be a PD, users could just move their device towards unreachable content, adjusting the viewport in such a way that the target is close to the thumb. So content could be tapped without stretching the thumb or adjusting the grip on the device.

[2] A mobile device larger than a typical phone but smaller than a typical tablet.

2.3.1 Adapted PD prototype for pointing

An iPhone 6 plus, a typical phablet device, with a screen size of 68 x 122 mm was used for implementation. The spatial processing was set to not require any explicit activation. Hence, it was always on but only translation was processed in order to disable scaling. The result was a PD ($m_{xyz}r_{xyz} : m_{u_1u_2}$), which is depicted in Figure 2.1(c) (however, the prototype used in the evaluation did not use the thumb tracker shown in the figure). The device displayed a red rectangle which was the pointing target. A trial was considered finished when the user successfully pointed (tapped) on it. A green screen followed on which the number of remaining tasks was displayed. It had to be tapped to proceed with the next trial. Depending on the configuration rectangles where either 5, 7 or 10 mm in size, which was inspired by the study design of Wolf [59, p. 70], who based her choice of target sizes on the studies of Parhi et al. [42] and Hasan et al. [18]. Positions of targets were randomly distributed and saved in a trial list that was the same for all participants and for all techniques (within subject design).

2.3.2 Comparison to *"Reachability"* technique

While testing the prototype with two visitors of the lab (performing 100 trials with target size 10mm) it became evident that the PD is superior to Apples Reachability technique. The two users where extremely slow with Apples technique. One participant was not even able to finish some tasks because targets where still out of thumbs reach after the content had slid down (see Fig. 2.1(b)).

2.3.3 Touch based implementations of the prototype

In order to judge performance of the PD it had to be compared to state of the art techniques that are capable of solving the same problem. Wolf [59, p. 13-16] assembled a comprehensive overview of pointing techniques with some of them also addressing the problem of screen accessibility. One approach is the remote control of a cursor using a drag motion. This approach motivated two different interaction techniques: the **extended cursor** and the **inverse cursor**. The extended cursor, depicted in Figure 2.1(d), is inspired by the cursor used by touch pads for desktops. Once a dragging motion is detected, the cursor emerges at the users finger tip and travels in the direction of the drag with a 1:2 CD gain. The inverse cursor, shown in Figure 2.1(e), is only different from the extended cursor in so far that it travels in the opposite direction of the drag motion. Another approach is followed with the **drag content** technique

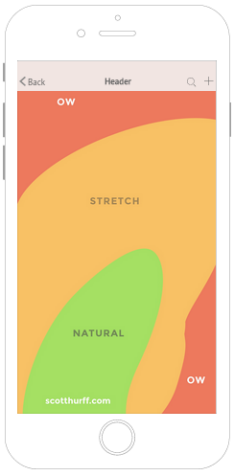
depicted in Figure 2.1(f). Here the user moves content towards him/her using a drag gesture, which makes it the touch based inverse of the PD.

All techniques where implemented in such a way that they still allowed direct touch pointing. So if a target was already within thumb reach, users could directly touch it without being forced to use the technique. This was done because *"In pointing research with mobile phones, direct touch was in comparative studies always found to be the fastest technique"* [59, p. 15]. It should be mentioned that there are even more touch based approaches that address the issue of screen accessibility, which, however, have not been implemented in the prototype. An example are miniature interaction areas [29] that are placed within the users thumb reach. Such an area is a representation of the entire screen and therefore allows the user to point anywhere on the display. However, such approaches suffer from the Fat Finger problem [58] and require very exact pointing of the user. For this reason, they were not considered for testing.

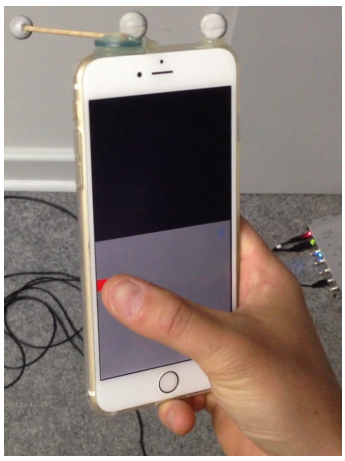
2.3.4 Comparison to touch based techniques

In a small pilot study two participants worked with all four techniques. They used each one to point on 100 targets with 10mm in size. Average task completion times for the extended cursor ($\bar{t} = 0.965s, \sigma = 0.434s$) and the inverse cursor ($\bar{t} = 1.013s, \sigma = 0.631s$) were lower than for the PD ($\bar{t} = 1.031s, \sigma = 0.300s$). Performance with the drag content technique ($\bar{t} = 0.999s, \sigma = 0.310$) was diverse among the participants. One performed best/ fastest with it whereas for the other it was the technique with the highest completion time. In a second session the experiment was repeated with 5mm targets. Here again completion times for extended cursor ($\bar{t} = 1.309s, \sigma = 0.847$) and inverse cursor ($\bar{t} = 1.446s, \sigma = 1.332$) were lower than for the PD ($\bar{t} = 1.530s, \sigma = 0.669$). And similar to the first session, the drag content technique was best/fastest for one subject and slowest for the other (overall $\bar{t} = 1.230s, \sigma = 0.404s$).

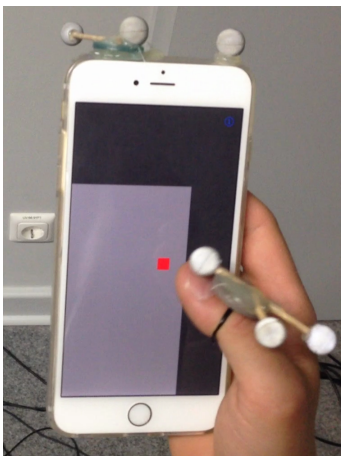
However, it has to be emphasised that surely no final conclusions should be drawn from this analysis as this study did not have enough participants (N=2) to produce reliable results. Nevertheless, observations made during the study led the author to the conclusion that with the touch based techniques some good alternatives to the slide down approach have already been found. This is why the conduction of a large scale study, to show that the PD is yet another feasible solution to solve the accessibility problem, was considered not fruitful. Instead a new design was developed in order to improve the spatial technique.



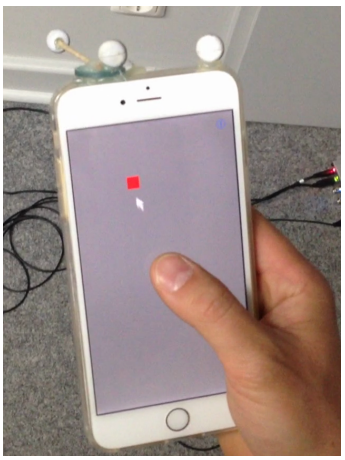
(a) Illustration by Hurf [21] showing areas on an iPhone 6+ display that are potentially hard to reach.



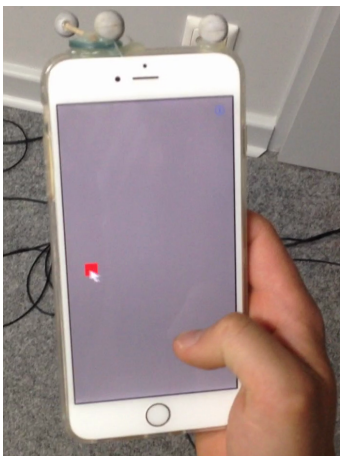
(b) The *"Reachability"* mechanism implemented by Apple on an iPhone 6+: a double tap on the home button causes the content on the screen to slide half way down . This still leaves some targets hard to reach.



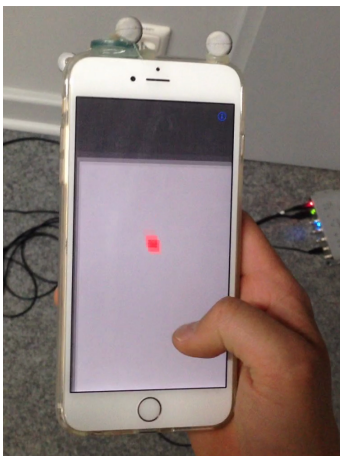
(c) The PD allows the user to move the target within thumbs reach. In this later version of the prototype the thumb is tracked as well. Once close to the screen content remains stationary making it easier to hit.



(d) The **extended cursor** moves ahead of the users drag movement. As a result users "push" the cursor towards the target. Upon releasing the finger the cursor "clicks" at its current position.



(e) The **inverse cursor** moves in the inverse direction of the drag movement. A pulling finger motion moves the cursor away from the finger and towards the target.



(f) With this technique the user can **drag content**, resulting in a similar finger motion as with inverse cursor. Once within the thumbs reach the target can be tapped effortlessly.

Figure 2.1: Implemented pointing techniques of the prototype.

2.3.5 Iteration of PD pointing technique

During the pilot study it was observed that the PD suffered from a problem. Once users had moved the target and started to close in with their thumb to tap it, they involuntarily moved the display slightly, which caused the target to slip away under their thumb. This is a problem other researcher have experienced as well:

> "Furthermore, during implementation of the spatially-aware techniques we noticed that accurately homing in on and tapping small map markers was cumbersome as any jitter from device–or the user's in-air pointing finger–causes small erratic displacements of the displayed map area and its markers." [17]

For this reason the prototype was iterated. In the new design the user's thumb was tracked as well, using a custom build marker as depicted in Figure 2.1(c). Now if the finger was very close to the display spatial processing was halted (implementation details: Section 4.4.3). This avoided the slipping but introduced another problem. Users usually keep their thumb within millimetres of the display *"lurking"* for the target. But the small markers attached to the thumb did not allow for such an accurate tracking to detect whether the thumb was in its waiting position ($\approx 4mm$ away) or if it was just about to tap the target ($\leq 2mm$ away). This forced users to consciously lifting the thumb further away from the display as they would usually do, which made the interaction mentally more demanding and less natural. Another approach to solve the problem was much more straight-forward and successful. Here the translation filter was removed from the spatial technique, making it a PD ($m_{xyz}r_{xyz} : m_{u_1u_2v}$). Now users could scale target to such a size that they were hard to miss, even if the targets slipped.

Another evaluation was conducted with the latter setup. Two participants pointed on targets sized 10, 5 or 2 mm (with 100 trials for each size). Only the PD ($m_{xyz}r_{xyz} : m_{u_1u_2v}$) and the extended cursor were tested. For 10 mm targets completion times were lower for the extended cursor ($\bar{t} = 0.901s, \sigma = 0.318089s$) than for the PD ($\bar{t} = 1.117s, \sigma = 0.277s$), whereas for 5 mm targets the PD was the faster technique (for PD $\bar{t} = 1.365s, \sigma = 0.439s$ vs. for ext. cursor $\bar{t} = 1.541s, \sigma = 0.998s$). Participants were not able to finish the 2mm trials with the cursor but could finish them with the PD ($\bar{t} = 1.795, \sigma = 0.491$).

2.3.6 Discussion and limitations

While enabling scaling made the PD faster it resulted in an unfair comparison to the touch based techniques as they were incapable of it. The drag content technique could be easily modified to allow scaling using the two-handed pinch gesture. The resulting technique would be equivalent to the Pinch-Drag-Flick technique, which has already been compared to the PD in the navigation study [54]. This is why results of the navigation study already provide a strong indication of how such a technique would compare to the PD for pointing tasks. The study had three types of tasks: scaling tasks, translation tasks and search tasks. When working on a translation tasks, users where presented with a red rectangle initially visible on screen (no search). They had to adjust the visible section (using the navigation technique) in such a way that the red rectangle fitted inside a stationary black frame on the display. For the just presented pointing task the visible section had to be adapted so that the thumb was located just above the red rectangle, which makes the two task types very similar. The main difference is that for the pointing study the target had to be tapped in order to complete the task.

But the tapping of the target is very similar for the PD and drag content (or Pinch-Drag-Flick) technique. Consequently, if one technique was to be faster than the other, it had to be faster in the navigation phase. This conclusion led the author to reexamine the data of the the navigation study, finding that the PD was only superior in scaling and search tasks whereas there was no difference to pinch-drag-flick in pure translation tasks. In the authors opinion this suggest that **the PD and drag gesture based techniques perform comparably well in the case of short translations**. This would explain why for pointing the PD did not outperform the touch based techniques.

Future work could further evaluate single handed multiscale navigation and pointing performance of PDs against touch based approaches such as Fat Thumb [5].

While this pilot study could not introduce a better performing alternative for single handed pointing on inaccessible items, it created a better understanding of PDs and their strengths and weaknesses. The insides gathered from all three lab studies motivated the author to derive some general best practises about when to use PDs and how to implement them. They are collectively presented in the next chapter.

3 Designing for navigation with dynamic peephole displays

The conduction of the presented studies, a review of a large corpus of related work and the discussion with experts allowed the author to develop a deeper understanding of navigation with PDs. A selection of the most relevant insights will be presented in this chapter. While some of these insights may also apply for ZLs, they differ significantly in some cases, which will be discusses as well.

3.1 Part one: strength of dynamic peephole displays

PDs can be considered an alternative for joystick and multi-touch navigation (compare Section 1.4 on page 12). Moreover, in several studies user preferred the PD over other techniques [60, 14, 54]. However, it is important to understand in which navigation use cases they constitute an improvement and are worth the effort of implementation and, on the other hand, in which cases they are just a *"fancy gimmick"* lacking any actual benefit. Generally the use of PDs only makes sense when explicitly exploiting some of their **strengths**, which are:

- *performance* in mutli-scale navigation (Section 3.1.1),
- *complimentary nature* to other inputs (Section 3.1.2),
- *single handed* operability (Section 3.1.3).

3.1.1 PDs allow fast multiscale navigation

The results of the navigation study [54] and the pointing study (Section 2.3) led the author to the following conclusion: PDs can **outperform any currently published touch-based translation and scaling technique** [1] in *multiscale* navigation. However, for *singlescale* navigation that only requires translation PDs have shown mixed performance (see Section 1.4.1 and 2.3.6). This means PDs excel in navigating **large**

[1] To achieve this, special care must be taken when implementing a PD, which will be addressed in part two of this chapter.

information spaces rather than small ones. A factor reducing navigation time of PDs in large spaces is the available **motor and interaction space** [23], which is considerably larger than that of touch[54]. Even though users may not utilise the available physical interaction space entirely [19], this still leads to **longer distances being covered with a single continuous movement** [35, 54]. As hinted in Section 2.1 this effect can be even increased with **faster mappings**.

At the same time PDs are just as easy or even easier to understand [54] and handle as direct touch gestures. This seems especially to be true in comparison to the pinch gesture, which seems to cause quite a few issues for users [54]. With PDs the integral properties of scale and translation are **intuitively mapped**, which allows users to benefit from their proprioception (sense of relative positions of neighbouring body parts)[54], as well as their 3D manipulation and coordination skills. Furthermore translation and scale are **simultaneously controlled**, which eliminates the need to switch between gestures. As a result users can take a **short diagonal path** through the SSD (see Figure 1.2(b) on page 4). While this is also possible with some touch gestures, that have been introduced in Section 1.2 on page 8, these gestures require users to plan their traversal ahead, as they need to explicitly define the zoom centre when initiating scaling, whereas with PDs they can adjust the zoom centre by adding in some translation while zooming. This also allows user to adapt their navigation strategy and correct mistakes on the fly, without the need for interruption.

For all of the above reasons the authors proposes the hypothesis that the difference in performance to touch increases proportionally to the minimally required path length within the SSD that is necessary to solve a navigation task. Or in easier words: the benefits of using PDs for navigation increases with the size of the information space. However it is important to acknowledge that translation and scaling may not always be the most feasible option in these large information spaces, as pointed out by Stürzlinger and Wingarve [55]:

> "Larger travel is usually handled by *'jumping'* to a new location, either via search or bookmarks. In other words, people prefer to "teleport" for larger distances rather than navigate."

Jumping techniques could also be used in spatial displays allowing users to skip navigation if they have a precise idea where they want to navigate to. This is a reasonable approach as a user who is currently viewing Berlin in Google Maps, is indeed more likely to simply enter *"New York"* into the search field instead of using translation and scaling in order to view the same result. But jumping mechanisms do

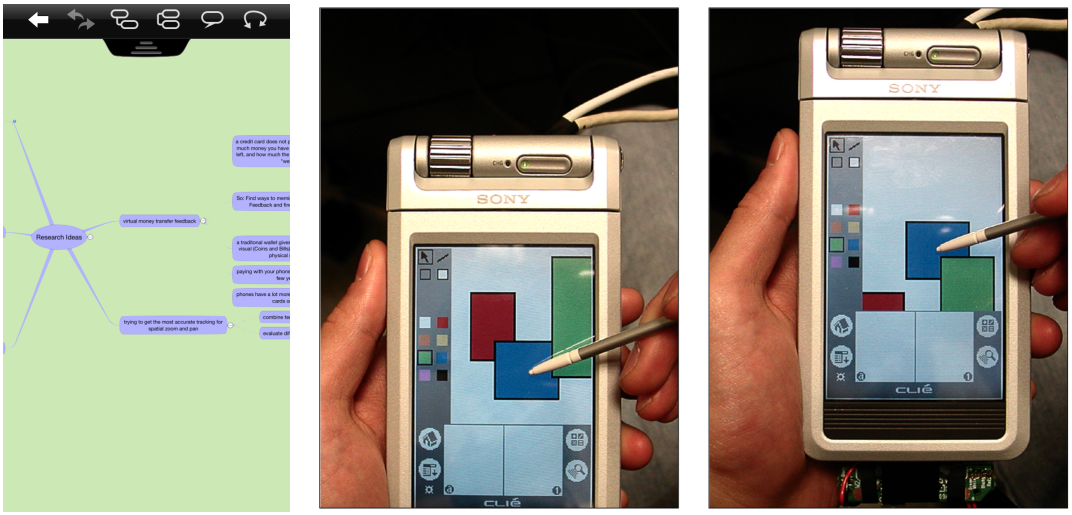
not cover all use cases, which is why navigation remains a relevant task. For example, a user who is searching for, e.g., camp grounds in the city of Aarhus will be confronted with more than a single result. In order to find the right result, the highlighted elements have to be inspected in detail, which yet again requires multiscale navigation. Furthermore, many things users are looking for are still beyond the capabilities of search engines, such as *"Find me a place to park under a bridge so my car is shielded from hailing"* or *"Find parts of wreckage of MH 370"*[2]. In cases like these users have to rely on visual search that still requires manual navigation in large information spaces. Moreover, explorative navigation with PDs and ZLs helps users to **memorise information** and facilitates long term spatial memory [44], which can be desirable for i.e. in zoomable user interfaces (ZUIs).

3.1.2 Multi-modal PDs: spatial movement as complimentary input

Unimodal devices rely on a single input modality to facilitate all interactions, like many handhelds that rely on touch gestures. But this introduces some problems as, for example, the drag gesture is very suitable for translation *and* pointing (Section **??**), which poses the question how it should be used for both. A solution employed on trackpads for desktop systems is using a two finger drag for translation and a single finger drag for pointing. However, this is not a feasible option for handhelds. So instead a single finger drag of the thumb on a mobile device may be interpreted as pointing *or* translation, depending on where (space multiplexing - e.g., dragging from an Edge [33]), how (variation - e.g., contact size of drag [5, 33]) or when (time multiplexing - "modes") it occurred. But these types of arrangements are known to **increase error rates and mental load** [7, 34, p. 96], especially if it is not apparent for the user how to trigger the desired interpretation of the gesture. Furthermore, simultaneous translation and pointing is not possible this way (sequential input). Alternatively, **task parallelism can be achieved by using separate input modalities**. Spatial movement as a **complimentary input modality in a multimodal system** could be used for exactly that matter. Users could move their handheld device for navigation while **simultaneously** performing modifications via touch, stylus or any other modality [41]. Such systems could **reduce the need for modes** and therefore prevent errors and increase user performance at the risk of increased complexity and mental load.

An illustrative example for this is the relocation of nodes in mind-maps or node link diagrams (Fig. 3.1(a)). First the user needs to find the object within the large

[2]Volunteers are searching high-res satellite images to determine the whereabouts of the missing plane `http://www.tomnod.com/campaign/malaysiaairsar2014`



(a) Mind-mapping on a multi-touch handheld using the app iThoughts.

(b) With this PD prototype presented by Yee et al. [60] users can manipulate objects with a stylus while navigating by moving the display. The stylus *"picks up"* and *"holds"* the object while navigating.

Figure 3.1: Spatial movement as complimentary input

structure (search navigation task), select it (pointing task) and move it to its new location (manipulation task). But if the new location is not immediately visible, this also demands for **navigation as a secondary task**. An unimodal solution for this that uses sequential input via modes is the *"cut and paste"* strategy. In manipulation mode the object is *"cut"* from its current position and saved to the clipboard. Next, entering navigation mode, the user travels to the new location. Finally, back in manipulation mode, the object is *"pasted"* from the clipboard. Another common unimodal solution is the use of velocity controlled translation in manipulation mode triggered by moving the object close to the edges of the display. This way the user controls the position of the object as well as the position of the viewport at the same time with a single input. But the use of velocity control for isotonic (displacement sensing) input devices is generally a suboptimal design choice and quite likely to cause a loos of control [34, p. 84-86, 113]. Yet another unimodal solution is the introduction of new gestures that allow simultaneous control of manipulation and navigation, as implemented in the iOS application iThoughts (Figure 3.1(a)). Here one finger can be used to move the node while another finger translates the background (the result looks similar to finger skateboarding). While this gesture is easier to handle than velocity control, it requires two hands, has to be learned and occludes large areas of the screen.

A solution using a touch-enabled-PD could allow the user to select and move an object using his/her thumb while simultaneously navigating the mind-map by moving the device. This can be done without the need to enter a mode. A solution similar to this, but using a stylus-enabled-PD ($m_{xyz} : m_{u_1u_2v}$), was presented by Yee et al. [60] and is depicted in Figure 3.1(b). When evaluating spatial as complimentary input in a drawing task (see Figure 1.6 on page 11), they found their participants to finish their task 32% faster with the PD than with unimodal stylus input. Hence, they demonstrated that mulit-modal PDs may not only **reduce the need for modes** and therefore prevent errors, but may also allow for faster task completion.

Generally it can be said that **more future work** is needed to identify real-world applications that benefit from spatial movement as a complimentary input. However, great care is necessary when implementing such a system for it might may be error-prone as the combination of modalities introduces more points of failure.

3.1.3 Single-handed operability

Lightweight PDs can be moved single-handed, which is very useful in situations where only one hand is available for interaction, whereas other common inputs such as multi-finger gestures (e.g., pinch) require two hands for interaction. With PDs even **multi-modal single-handed interaction** is possible. For example, a user could move the device with one hand for spatial navigation while at the same time using the thumb for touch input. Generally, it can be said that spatial input has potential to enrich single-handed interaction with mobile devices. This potential has not yet been sufficiently explored in current research. However, one limitation of the usefulness of spatial input is that single-handed use cases might be linked to walking (e.g., viewing a map while exploring a new city), which introduces difficulties for PDs [54], as they would have to distinguish between the intended movement contributed by the arm and the unintended movement contributed by walking. This could be solved by tracking the user's arms and hand movement relative to his/her torso rather than the movement of the display relative to itself. However, to the author's knowledge this has not yet been done in a mobile context (mobile tracking system).

3.2 Part two: Design recommendations

PDs have shown diverse performance in user studies (Section 1.4). The experiments differed in many factors such as task types, display sizes, gender balance etc. But they also differed in some design and implementation choices, which can be expected to

generally impact the performance of PDs. In the following section some important considerations for such choiches are presented:

- Section 3.2.1 looks at the diversity of tracking and processing technology and points out which properties might considerably influence user experience and performance
- Section 3.2.2 advocates the use of affordable activation mechanisms and clutching
- Section 3.2.3 suggests a dynamic orientation mapping that implements the dynamic peephole metaphor very well

3.2.1 Tracking technology

The tracking technology used in ZL and PD prototypes has seen large variety. Examples include cable bound tracking [10, 57] (Fig. 1.5 on page 10), booms [56], lasers [60], internal cameras [14, 16], motion capturing system that use a composition of external cameras [51, 44, 54, 23, 17](Fig. 4.1(a) on page 36), external depth cameras [52] and magnetic resonance tracking [41].

Nowadays spatial tracking on **consumer mobile devices** is possible to some extend but often limited in DoF. For example *Tilt-to-Zoom* [5] applications only utilize a single spatial degree and therefor are only a PD ($r_y : m_v$), which is incapable of spatial translation and does not full comply with the peephole metaphor. Some other tracking approaches [60, 15, 25] are limited to 3 DoF. However, **6 DoF tracking is necessary** to implement the dynamic mapping described in Section 3.2.3, which has been found to deliver the best performance [54, 41]. Yet again other applications [25] support spatial zooming and panning, but only as separate actions, eliminating simultaneous control as one of the greatest strengths of PDs (Section 3.1.1). Some recent approaches are using a gyroscope and are fusing the orientation and acceleration data with feature tracking of an internal camera [25], an external depth camera [45] or an internal depth camera [12, 39]. Especially the latter approach seems to be promising, as there have been many relevant advances in technology and computer vision research in recent years, such as the LSD-SLAM algorithm [9] and Google Tango [12]. Developments like these could pave the way for cheap and broadly available **device intrinsic 6 DoF spatial tracking** allowing consumer mobile devices to act as PDs. However, some of these system may lack the required **accuracy**. As pointed out earlier, humans possess remarkably accurate spatial manipulation skills, which is why a tracking system that does not match these capabilities is negatively impacting, if not jeopardising, the effectiveness of PDs. Another very important factor is **latency**, which causes a delay

in visual feedback and therefore in the feedback loop, which can severely impact the usefulness of the system [1, 34, pp. 81-84]. Even an extremely small latency (1 ms) is still noticeable by humans, as has recently been empirically shown for touch [38] and stylus [37] interaction. It can be expected that, even with the current developments in tracking technology, latency will remain a unneglectable factor of influence for the usefulness of spatial displays.

3.2.2 Activation and clutching

Being moved lies in the very nature of mobile devices. However, not all movements should be considered spatial input. For this reason PDs need an activation mechanism that defines the starting and end point of an intended spatial interaction. This is also referred to as clutching. Similar to a clutch of a car, that decouples the running engine from the transmission, it allows users to *decouple* spatial movement from the spatial interaction processing unit. As a result users can move the device without the movement being interpreted as input. Clutching is a very important topic for spatial navigation:

> "some of the most confounding (for the user) and hard-to-fix (for the implementor) **usability problems and ergonomic difficulties** can arise due to poor clutch design [...] a poor clutching interface can jeopardize the usefulness of spatial input." [19]

Many works [50, 41, 17] avoided possible pitfalls in clutch design by not implementing one. But this limits the **size of the information space that can be navigated** to the users arms reach[17]. In large information spaces, where PDs are expected to excel, as argued in Section 3.1.1, clutching is a necessity. Furthermore, it allows users to **avoid uncomfortable or awkward positions**, which makes PDs more socially acceptable, user friendly and less fatiguing. At the same time users may be inclined to clutch very frequently and utilise a very small motor space, which has been suspected to have a negative influence on performance [23]. It is still left to **future work to identify ergonomic [23] clutching mechanisms** that work best for PDs and do not conflict with other modalities [54]. It is important that such a mechanism provides feedback and can be operated without requiring visual attention.

The use of clutching is a different question for ZLs. As defined in Section 1.3 on page 9 they exist within a visual context, which means they demand for a mapping relative towards it. A clutching mechanism would decouple this spatial relationship which questions the usefulness of the context after all. Either clutching for ZLs is not

used, which imposes limits on their performance, or ways of preserving the relationship between the lens and its context have to be found. For example non-stationary context (e.g., projections) could adapt in such a way that the relationship is preserved.

3.2.3 Mappings for multiscale dynamic peephole navigation

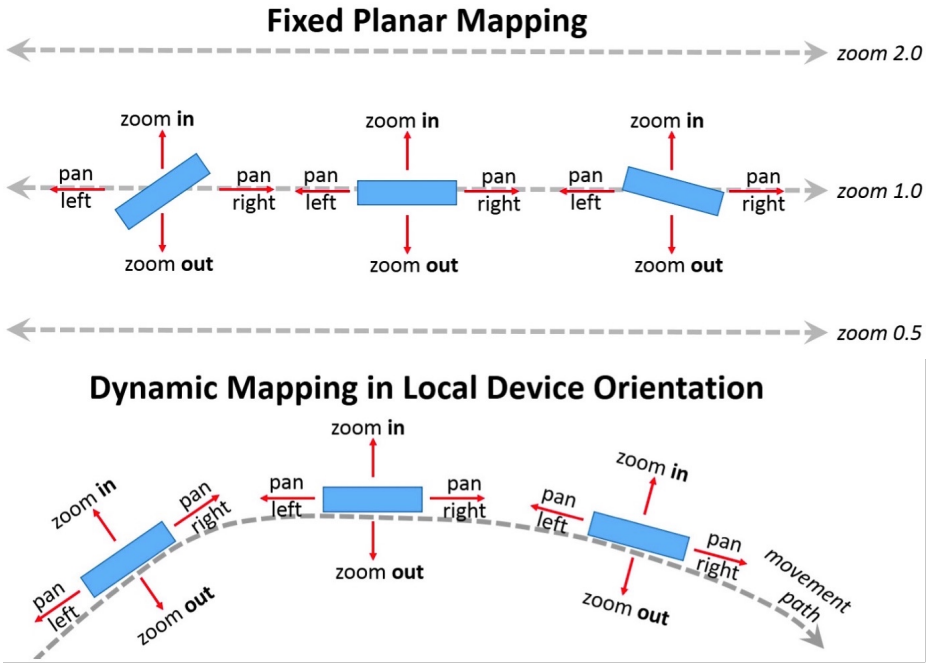


Figure 3.2: A relative planar and dynamic mapping as discussed by Pahud et al. [41].

The dynamic peephole metaphor is a core idea behind PDs. The difference between the dynamic and static peephole metaphor has been explained in Figure 1.1 on page 2. But there is more than one possible mapping for the dynamic peephole metaphor in 3D physical space, which makes the terminology a bit confusing. In an **absolute mapping** (of the dynamic peephole metaphor) the information space and the physical space are statically aligned creating the "illusion that the two worlds coexist"[1]. Here what is shown on the display depends on its position (and/or orientation) within an absolute coordinate system that is bound to the physical space. Absolute mappings are often used for ZLs [57, 51, 48, 30, 44] and augmented reality applications [1, 48].

But Hinckley et al. [19] note that "users may have trouble moving in a fixed absolute coordinate frame" and warn of "reduced task-performance due to cognitive load, fatigue, or both". As a solution they suggest to use relative gestures instead. Such gestures were used in the initial **relative mapping** used by Fitzmaurice et. al [10] and Yee et al.[60],

where the spatial reference was the device itself. An empowering factor of relative mappings are their detachment from the surrounding physical space. This opens up the possibility to use **custom mappings such as gain factors**, which can improve performance as demonstrated in the first small lab study (Section 2.1). But more importantly this makes them **suitable for a clutching mechanism**. In an absolute mapping such a mechanism is hardly useful as it is likely to cause disorientation due to disruption of visual flow while still leaving areas hard to reach.

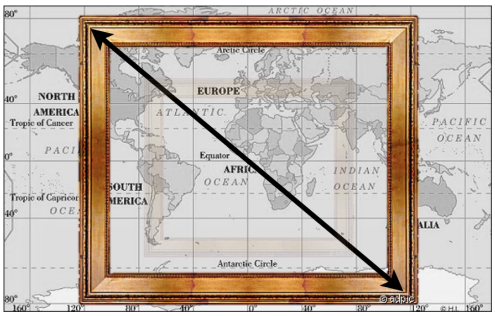
When mapping 3D movement and rotation (m_{xyz}, r_{xzy}) to mulitscale navigation ($m_{u_1u_2v}$) we are confronted with the question of how to meaningfully reduce dimensionality (six to three dimensions). The naive approach would be to simply ignore the devices orientation and just base navigation on its change in position. Such a *"fixed planar mapping"* [41] is depicted in Figure 3.2. But real time decisions are a main characteristic of navigation [26]. This means at any given point in time users may base their execution of movements on the current state of their spatial reference object. Hence, this object should reflect their options for navigations, meaning the rotation of the device should influence how its movement is processed. This can be done by rotating the (transformation) coordinate system to always match the current orientation of the device as introduced in 2013 by Pahud et al. [41] as well as Spindler and the author [54]. This is referred to as **dynamic orientation mapping** (not to confuse with dynamic peephole) and is depicted on the bottom of figure 3.2.

> "dynamic [orientation] mapping [...] uses the **current** orientation of the display as the new reference plane for future interpretations of motions. This means that zooming is mapped to movements along the normal of the display (local Z-axis), whereas motions within the display's XY-plane define panning." [54]

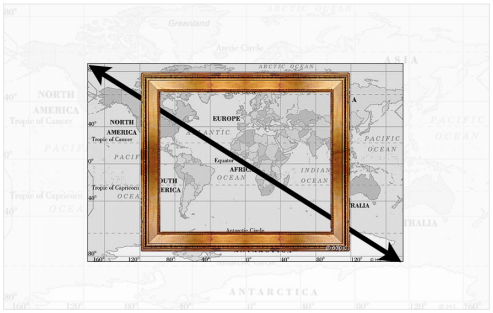
This mapping does not require users to move along a fixed spatial axis, which is desirable as they tend to move the device in body centric arcs. While Pahud and Spindler also considered a spherical body centric mapping, as suggested by Fitzmaurice et al. [10], they considered the dynamic orientation mapping superior, which is why the author recommends using it. However, this mapping could be adapted and improved. The rotation around the device's z-axis could be mapped to multiscale navigation rotation which would open up new ways of interaction but requires the user to control yet another parameter simultaneously. Furthermore, a problem of the current mapping is that it can not be used while walking, as it does not distinguish between intended and unintended movement. This problem could be addressed by tracking the unintended

movement and subtracting it from the movement of the display.

Zoom direction



(a) Dynamic peephole ($p = -1$): the user manipulates the position of the viewport (frame). Here it has been **moved towards** the user, appearing bigger as before. More content is visible but fewer detail (zoom out).



(b) Static peephole ($p = +1$): the user manipulates content (world map). If it is compressed via pinch gesture or spatially **moved away**, the same effect (zoom out) is achieved as in the picture on the left, but with the opposite movement.

Figure 3.3: Dynamic and static peephole control metaphors for scaling

The zoom direction has been a matter of discussion [41, 54, 60], which even demotivated researchers to implement zooming at all [17]. But the peephole metaphor does not only determine how translation has to be mapped (Figure 1.1 on page 2), it also determines the **zoom direction** as illustrated in Figure 3.3. The main argument presented against the *peephole zoom*, meaning zooming out when moving the device towards oneself (Figure 3.3(a)), is the belief that people tend to move objects they want to inspect in more detail closer to themselves instead of further away[54]. However, the peephole metaphor matches the users experience with viewfinders in camera or magnifying glasses and has been successfully used in several works [10, 44, 41, 60, 54]. In order to facilitate expert discussions about zooming directions at CHI 2014 [53] the prototype was adapted to offer both directions. After having presented both possibilities to numerous users as well as having been part of several discussions and having read extensive literature reviews the author concludes that dynamic peephole zoom ($p = -1$) has to be given clear preference and is therefore the recommended direction mapping.

4 Apparatus and implementations

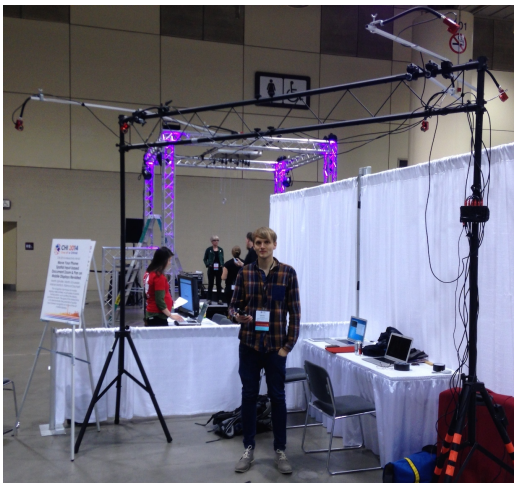
In this chapter some technical details about the prototype will be highlighted. The goal is to point out some solutions to problems one may face when trying to build a PD prototype using a handheld device. In the last section of this chapter the most advanced hybrid version of the prototype is presented. It includes many of the insights and design recommendations highlighted in this thesis.

The entire apparatus consists of a tracking and transmission system (Section 4.1) as well as various iOS devices running a custom build application (Section 4.2). The core modules of the application are the tracking module (Section 4.3), the exchange module (Section 4.4), the SSD module (Section 4.5) and the visualisation module (Section 4.6).

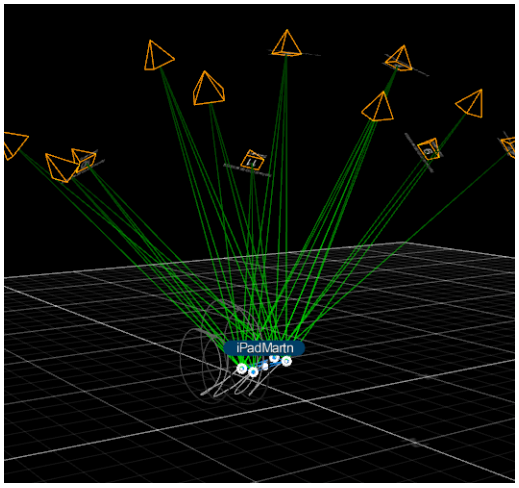
4.1 External optical tracking and wireless location data transmission

The initial setup of the external tracking and transmission system was based on the **Paper Lens System** developed and presented by Spindler et. al.[51, 54] (Figure 1.3(b)). It was replicated at the Telekom Innovation Laboratories but reduced to the essential components for spatially aware tangible displays. The setup consists of 16 **OptiTrack** FLEX V100:R2 IR cameras each delivering 640x480 px resolution at 100 Hz, targeting a tracking volume of roughly 2mx2mx2m. A portable version of the setup is depicted in Figure 4.1(a). After a careful calibration the system is able to track the four infrared reflective markers mounted to the headphone jack of an iPhone or iPad (Figure 1.7(b) on page 13) with sub-millimetre accuracy. Theses markers were tracked as 6 DoF rigid body which had its defined centre at the mid-point of the display. This way the position (x, y, z) and rotation (q_x, q_y, q_z, q_w) of the rigid body corresponded exactly to those of the tangible display. The tracking apparatus is connected to a *Display Server*, which is also a component taken from the Paper Lens setup. It broadcasts changes of the location or orientation of the rigid body encapsulated in yid-packages [1] via UDP through a 2,4 GHz **Wifi network**. The

[1] Yid packages are an extension of VRPN (Virtual-Reality Periphal Network) packages



(a) A smaller portable version of the complete setup as used by the author in [53]. The OptiTrack cameras are mounted on beams in such a way that they track objects in the volume below them.



(b) Visualisation of the tracking volume: An object (here an iPad) tracked in 6 DoF has to be marked by at least three markers, each in sight of at least 3 cameras.

Figure 4.1: Optical tracking system using external cameras to track handheld devices.

iPhone or iPad is also connected to Wifi and running a custom build application (described in the next sections).

4.2 iOS application - operation principle and core modules

The development and refinement of the iOS Application is one main contribution of this thesis and required a lot of work. When starting the development the prototype contained 4787 lines of code whereas today it contains 8983[2]. The source code is stored in a private git repository[3] and can be accessed upon request.

In the application the live updates of the tracking system are received and preprocessed by a **tracking module**. The data then is reencapsulated in an *MSDisplayData*-package and send through a chain of modules. These **chained core modules** are the backbone of the application and implement a complex and sophisticated processing procedure. To manage this complexity each module is required to conform to the *MSDisplayDataProtocol*, ensuring it can handle the data it is receiving. Each module can change values in the MSDisplayData package before delegating it to the next

[2]Mainly Objective-C code. This number is not a measure of functionality nor good programming style. It just servers the purpose of creating a vague impression of implementation complexity.
[3]https://github.com/mschuessler/spatialDisplays

module in the chain. This way any number of modules can be chained together, each focusing on a clearly defined subtask. Each subtask and its corresponding module are well defined, easy to understand and can be adapted with a reasonable amount of effort. At the same time modules can be swapped for other modules that implement different processing procedures, mappings or other experimental features.

In the following sections the chain of modules used in the current most advanced version of the prototype will be briefly presented along with some refinements that helped to improve the overall user experience.

4.3 Tracking module

The general purpose of the tracking module is to - firstly - receive tracking data from an external (tracking system) or internal source (sensor) and -secondly- to pre-process this data in a meaningful way so it can be handled by the subsequent chain of module. The architecture of the application was designed with flexibility in mind. A tracking module can be easily exchanged for another, so that it remains possible to use other technologies like Kinect, Fast Trak or externally mounted depth cameras like “Structured Sensor”.

The tracking module in the setup described here was implemented by the *YidClient*, which is a custom build receiver of yid packages send by the *Display Server*. An earlier version of the YidClient used in [54] was limited to some very basic validity checks. Only packages that were obviously damaged in the transmission process were detected. However even though the tracking and transmission system went through numerous iteration, malfunctions were a common issue. In such a case the visible section displayed on the tangible display started to jump, drift, flicker or remain completely stationary. Drawing conclusions about the cause of the problem by observing these mostly unpredictable events was often a matter of luck and required a lot of time. In some cases the tracking of the rigid body did not work as a result of changed lighting conditions. For example, bright sunlight or different ceiling lamps caused particularly strong reflection on the glass surface or led to the tracking of other infrared reflective objects like glasses or duct tape. In other cases, cameras had moved over night or broken entirely which led to falsification of the calibration. Moreover, several times the transmission of the data was interrupted by Wifi issues. Often enough the cause of appearing problems was simply a newly implemented feature.

To allow for faster debugging the **data analysis and data cleansing were significantly extended**. The refined YidClient calculates a number of statistics. One example is the *rate of incoming packages*. This can be an indicator for the optical

tracking quality as it tends to drop and fluctuate if the system is loosing track of the rigid body. If the rate is equal or close to zero, the transmission system should be checked for malfunctions. Furthermore, reflections and other causes can lead to wrongly reported positions leading to extreme jumps of the visible section and causing a loss of orientation for the user. Hence, an *outlier detection algorithm* is needed. In the developed algorithm data points are added to a sliding time interval. The average movement speed over this time interval is calculated. If a newly received data point would mean a sudden increase of movement speed by very high percentage (e.g. 300%), the data point is considered an outlier and dropped. The number of dropped package is reflected in the *error rate*. If the error rate exceeds a given threshold or the package rate falls below a threshold, the screen is coloured slightly blue to provide visual feedback to the user about the technical issues. Additionally, if the users is working on a task as part of a study, the trail in question is silently invalidated and scheduled for retrial. This way it is ensured that every task was performed under acceptable tracking conditions.

4.4 Exchange module

This module is the receiver of the cleaned position and rotation data of the tracking module and consolidates this data into a canonical position without rotation, which also means a dimensionality reduction from six to three parameters.

4.4.1 Canonical transformation

In order to achieve the dimensionality reduction a **metric interaction volume** is defined with a position (*transformationMetricReferencePoint*), an orientation (*transformRotation*) and a size (*metricBounds*). The position of the display inside of this volume can be described canonically and without using any orientation data. The canonical position is calculated[4] in the function *transformPosition* shown in Listing 1.

Choosing the right size for the volume is a bit tricky as it defines the mapping between the physical and virtual movement. The height of the box which is mapped to scaling was determined in several iterative user interviews and set to 450mm. The ground plane (x,y) of the volume is based on the size of the information space at scale one. But as the information space is defined in points (not in pixel)[5] the size needs

[4]The math used in this rotation involves quaternions which makes it quite complex. A formal expression for this has been generated but is too large to be printed.

[5]UIKit is using points instead of pixels. For retina devices the pixel to point ratio is 4:1 while for

```
//Calculating canonical position relative to reference point
-(GLKVector3)transformPosition:(GLKVector3)position{
    GLKVector3 p = GLKVector3Subtract(position,self.transformationMetricReferencePoint);
    p = GLKQuaternionRotateVector3(self.transformationRotation, p);
    p.x/=self.metricBounds.width;
    p.y/=self.metricBounds.height;
    p.z/=self.metricBounds.depth;
    return p;
}
```

Listing 1: Transforming metric positions into canonical positions inside the volume.

to be transformed into mm. To facilitate this a *points-to-millimetre* factor, hereafter *ppmm*, was determined manually for all devices in the lab. This was done by dividing the screen resolution in points by the physical screen size in mm which was manually measured. The prototype has deployed to an iPhone4, 4s, 5, and 6+ as well to an iPad 3, Air, Air2 and mini 3. So for these devices bounds will be calculated correctly. As conversion is done centrally in a dedicated static class, shown in Listing 2, new devices can be added easily. The bounds can also be used for **linear gain factors** (Section 2.1). For example a gain factor of *2:1* can be achieved by doubling the *ppmm*.

4.4.2 Clutching and dynamic mapping

The exchange module also acts as the *clutch* (Section 3.2.2) that determines whether spatial movement is processed or not. Any location data that is received while the clutch is *OFF*, meaning decoupled, is used to move the interaction volume, as in this state the display does not move within it. Instead the volume follows the display (like a bubble around it). Once the module receives the *ON* signal by the clutch, location data of the display is processed differently (Listing 3). First the position of the display within the volume is calculated using *transformPosition*. The new canonical position, which also determines the **canonical movement since the last data package** $(\Delta x, \Delta y, \Delta z)$, is passed on to the next module (*delegate*).

The only difference between **dynamic and relative mapping** (Section 3.2.3 on page 32) is that for dynamic mapping the just received change in orientation leads to an update of the origin and orientation of the interaction volume coordinate system, which is done by the function *newReferencesWithData* shown in Listing 3. Consequently, in relative mapping the last orientation that was received before interaction began determines the orientation of the interaction volume, whereas for dynamic mapping it is continuously updated.

non-retina devices it is 1:1

```
+(CGPoint)convertPxPointToMm:(CGPoint)p{
    return CGPointMake(p.x/[MSDefaults ppmmw], p.y/[MSDefaults ppmmh]);
}

+(CGFloat)ppmmh{
    NSString *device = [MSDefaults deviceType];
    if ([device isEqualToString:@"iPhone5,2"])
        return 6.29362880886427;
    if ([device isEqualToString:@"iPhone7,1"])
        return 6.0327868852459;
    if ([device isEqualToString:@"iPad4,1"])
        return 5.19796954314721;
    if ([device isEqualToString:@"iPad4,7"])
        return 6.4;
    return 6.26223091976517; //iphone4 value
}

+(CGFloat)ppmmw{
    NSString *device = [MSDefaults deviceType];
    if ([device isEqualToString:@"iPhone5,2"])
        return 6.2015503875969;
    if ([device isEqualToString:@"iPhone7,1"])
        return 6.08823529411765;
    if ([device isEqualToString:@"iPad4,1"])
        return 5.22448979591837;
    if ([device isEqualToString:@"iPad4,7"])
        return 6.4;
    return 6.2015503875969; //iphone4 value
}
```

Listing 2: Conversion between pixels and mm is done using manually calculated values. (MSDefaults.m)

4.4.3 Thumb tracker

The yid Client is capable of processing packages of several different objects, which are differentiated by IDs. This was taken advantage of in the pointing study (Section 2.3). Here the user's thumb was tracked and spatial processing was deactivated once it was close to the screen (Figure 2.1(c) on page 22). The exchange module was adapted in such a way that it processed location data of the thumb differently. It was transformed into the same coordinate system as the display data. In this coordinate system the altitude of the thumb above the display is described by its distance on the z-axis from the display position. If this altitude was larger than a predefined threshold the clutch was set to *ON* and vice versa.

```
-(void)handleDisplayData:(MSDisplayData *)displayData{
    if (!displayData.isPrimaryDisplay) //Thumb data is processed differently
        return [self processSecondaryDisplayData:displayData];

    //All data that is received while clutching moves the reference point
    if(self.firstYidDataPackage) //This Bool is true while clutching
        [self newReferencesWithData:displayData];

    if(self.processYidData){
        self.lastCanonicalPosition=[self transformPosition:displayData.position];
        displayData.canoncialPosition=self.lastCanonicalPosition;
        self.firstYidDataPackage=NO;
        [self.delegate handleDisplayData:displayData];

        if(self.orientationIsDynamic)
            [self newReferencesWithData:displayData];
    }
}

-(void)newReferencesWithData:(MSDisplayData*)yidData{
    GLKVector3 lastPosition = self.lastCanonicalPosition;
    lastPosition.x*=self.metricBounds.width;
    lastPosition.y*=self.metricBounds.height;
    lastPosition.z*=self.metricBounds.depth;
    GLKQuaternion rotateBack = GLKQuaternionInvert(yidData.orientation);
    lastPosition = GLKQuaternionRotateVector3(rotateBack, lastPosition);

    //Setting new origin & orientation for interaction volume
    self.transformationMetricReferencePoint=GLKVector3Subtract(yidData.position, lastPosition);
    self.transformationRotation=yidData.orientation;
}
```

Listing 3: The core function of the exchange module processing the received tracking data.

```
-(void)processSecondaryDisplayData:(MSDisplayData*)yidData{
    GLKVector3 newPos = [self transformPosition:yidData.position];
    GLKVector3 thumbRelativeToDisplay = GLKVector3Subtract(newPos,self.lastCanonicalPosition);
    CGFloat thumbAltitude = thumbRelativeToDisplay.z * self.metricBounds.depth;
    BOOL altitudeTrigger = thumbAltitude >= [MSDefaults thumbTriggerHeight];
    self.processYidData = heightTrigger;
}
```

Listing 4: Processing the position of the thumb

4.5 Space scale diagram module

The SSD module receives the updated data package from the exchange module. Movements that occurred along each axis of the device $(\Delta x, \Delta y, \Delta z)$ between the last $(t-1)$ and the current measurement (t) have been added to the package. The **movement filters** $I_x, I_y, I_z \in [0,1]$ determine which of them should be processed. Movement filters can be set by any object that has access to this module. The **centre of zoom** $(c_x(t), c_y(t))$ is defined as an offset from the centre of the display. It can be set by other modules, e.g., a tapping gesture recogniser, by sending a notification to the SSD module. In the default case the centre is (0,0), which defines it at the vertical and horizontal centre of the display. This default was used in all studies. The factor $p \in [-1,1]$ is the peephole zoom factor which is -1 for dynamic peephole zoom as was shown in figure 3.3 on page 34. **Smooth zooming** is implemented by $\Delta s_{xy} = v(t-1) * \Delta z$, which means the change of zoom is amplified by the current zoom scale, harmonising visual flow [54]. The new position in the SSD $(u_1(t), u_2(t), v(t))$ is obtained as described in Section 1.1.4 on page 4: "Δs_{xy} is mapped as a scalar to a vector that transfixes the u_1u_2-plane at each level v at the zoom centre $c_v = (c_x * v, c_y * v)$". The result of this mapping is a zooming movement vector that is added to the previous position $(u_1(t-1), u_2(t-1), v(t-1))$ along with a translation vector as shown in the following function:

$$f_{ssd} = \begin{pmatrix} u_1(t) \\ u_2(t) \\ v(t) \end{pmatrix} = f\left(\begin{pmatrix} I_x\Delta x \\ I_y\Delta y \\ I_z\Delta z \end{pmatrix}, \begin{pmatrix} c_x(t) \\ c_y(t) \end{pmatrix}, \begin{pmatrix} u_1(t-1) \\ u_2(t-1) \\ v(t-1) \end{pmatrix} \right)$$

$$f_{ssd} = (1 + \Delta z p) \begin{pmatrix} u_1(t-1) + c_x(t) \\ u_2(t-1) + c_y(t) \\ v(t-1) \end{pmatrix} + \begin{pmatrix} I_x\Delta x - c_x(t) \\ I_y\Delta y - c_y(t) \\ 0 \end{pmatrix}$$

The equation can be rewritten so that the two components of translation and scaling, which are added to the previous position, become visible:

$$f_{ssd} = \begin{pmatrix} u_1(t-1) \\ u_2(t-1) \\ v(t-1) \end{pmatrix} + \begin{pmatrix} I_x \Delta x \\ I_y \Delta y \\ 0 \end{pmatrix} + I_z \Delta zp \begin{pmatrix} u_1(t-1) + c_x(t) \\ u_2(t-1) + c_y(t) \\ v(t-1) \end{pmatrix}$$

It is possible to **integrate linear CD gain factors** (Section 2.1 on page 15) into this formula by simply defining I_x, I_y, I_z as canonical instead of binary. This means a mapping that doubles panning speed but halves zooming speed is implemented by $I_x, I_y = 2$ and $I_z = 0.5$. However until now a linear CD gain has only been implemented in the exchange module

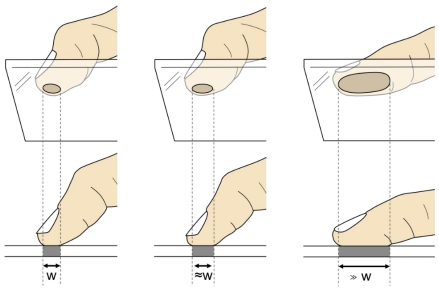
4.6 Visualisation module

The visualisation module receives the updated data package, which now contains information about what section of the information space should be displayed. Low latency in visual feedback is crucial, but uncompressed large information spaces can easily consume the entire RAM of a handheld device introducing significant lag. To address this issue two mechanisms tweaking performance were developed and integrated into this module. The first are **scale layers**. Instead of storing the entire information space in RAM, several versions of the information space at different scales are precomputed and saved to storage. They are loaded into RAM when the module is visualising content in low scale (e.g., an overview after a zoom out). As they only require a fraction of the RAM and processing power for rescaling, they speed up visualisation significantly. The number of layers, their compression and the scale when to stop using them can be configured in the application settings. This allows the operator to adapt configuration in regard to the capabilities of the device at hand. As an example the iPhone 4S has been found to run smoothly with 4 layers at 50% (0.5) quality, an iPhone 6 plus can already use 2 layers at full quality (1.0), whereas an iPad Air 2 does not require any layers at all.The pre-computation of layers is executed automatically if the settings have changed and can take a little while. The second performance optimisation is the **tiling of the original information space**. When no precomputed scale-layer is used, which happens when displaying high scale factors, the information space is subdivided into rectangular tiles. Only tiles that are currently displayed and neighbouring tiles, that are expected to be displayed soon, are loaded into RAM. The combination of scale layers and tiling allows to visualise very large information spaces while introducing minimal lag. The encapsulated layers are translated and scaled using affine transformation matrices (Section 1.1). This means

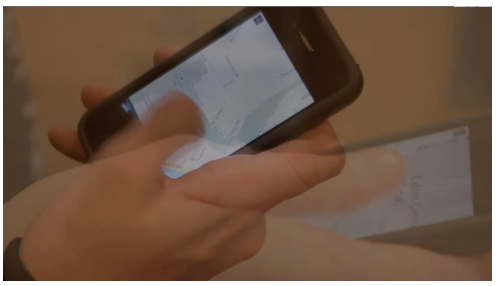
every position in the SSD has a directly corresponding transformation matrix and vice versa (bijective relationship).

4.7 Single handed hybrid peephole

After conducting the studies described in chapter 2 the prototype was iterated to incorporate many of the recommendations that have been pointed out (Section 3.2), while exploiting two of the PD specific strengths (Section 3.1). The result is a hybrid peephole, that can be viewed as the final conclusive contribution of this thesis. It combines touch and spatial input for navigation. Contents can be translated with one hand using the drag gesture, which in contrast to the spatial technique does not suffer from unintended scaling as it only allows two degrees of freedom, making it naturally **constrained**. As pointed out before this gesture is very suitable for brief single-scale navigation (pure translation). But if users want to zoom or cover larger distances, they can switch into **spatial multiscale navigation** by simply *pressing down* their finger a bit harder. The **contact size** of the finger on the display acts as the **clutch** for spatial processing, as illustrated in Figure 4.2.



(a) The contact shape provides additional information about a touch. It can be characterised by its contact size (see also Boring et al. [5]).



(b) Symbolic picture: In the hybrid prototype a small contact size is still used for translation (upper position) while a large contact size activates spatial mode, which allows simultaneous scaling and translation (lower position).

Figure 4.2: Contact size as mode trigger for spatial input.

As result users can switch seamlessly between a drag gesture and spatial zoom by fastening their grip. They are not forced to use the spatial technique when no benefit is expected, for example when browsing content that is optimised for mobile displays or well-suited to be displayed in a single scale on mobile devices (e.g., mobil-friendly websites, emails, calendar). But whenever they do encounter **large information**

spaces (photos, unoptimised websites, maps) or **items that are hard to reach**, the power of **fast single handed mutliscale navigation** is just at their fingertips. Another benefit of the combination is that it allows user to adapt quite **fast mappings** for spatial translation as they can easily combat overshooting with a corrective drag gesture (which is suspected to be more precise for translation [55]). The strength of both techniques are utilised while avoiding many of their weaknesses.

The author wants to point out that the power of this new prototype lies in its sophisticated single-handed-operability, leaving the other hand free for other tasks or additional interaction with the device at hand.

4.7.1 Details for hybrid implementation

The source code of FatThumb[5], which inspired the use of the contact size as a clutching mechanism, was kindly provided by Sebastian Boring. It was partly rewritten and made available as a lib file by the author so that it can be used by other applications. The spatial processing unit was not initially designed to be combined with another navigation techniques (only manipulation techniques were considered). Every module within the processing chain only updates its internal states upon receiving a new spatial data package. But the iOS implementation of touch based interaction updates the visual representation of content directly, leading to a **misalignment** of the spatial processing chain and the visualisation module. One option to combine both techniques is the use of a gesture recogniser that reports directly to the SSD. Though this would be an elegant solution it would require manual implementation of the gesture interpretations which introduces many points of failure. Instead the position within the SSD was derived using the **affine transformation matrices** of the visualisation module, which were updated by the SSD as well as the native iOS touch gesture recognisers. This made it possible to inversely calculate all internal states of the modules without rewriting the entire application.

Bibliography

[1] Ronald T Azuma and others. 1997. A survey of augmented reality. *Presence* 6, 4 (1997), 355–385.

[2] Patrick Baudisch and Ruth Rosenholtz. 2003. Halo: A Technique for Visualizing Off-screen Objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 481–488. DOI: http://dx.doi.org/10.1145/642611.642695

[3] Benjamin B. Bederson and James D. Hollan. 1994. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*. ACM, 17–26. DOI: http://dx.doi.org/10.1145/192426.192435

[4] Eric A. Bier, Maureen C. Stone, Ken Pier, Ken Fishkin, Thomas Baudel, Matt Conway, William Buxton, and Tony DeRose. 1994. Toolglass and Magic Lenses: The See-through Interface. In *Conference Companion on Human Factors in Computing Systems (CHI '94)*. ACM, New York, NY, USA, 445–446. DOI: http://dx.doi.org/10.1145/259963.260447

[5] Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. 2012. The fat thumb: using the thumbs contact size for single-handed mobile interaction. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services (MobileHCI '12)*. ACM, New York, NY, USA, 39–48. DOI:http://dx.doi.org/10.1145/2371574.2371582

[6] Kevin F. Bury, James M. Boyle, R. James Evey, and Alan S. Neal. 1982. Windowing vs Scrolling on a Visual Display Terminal. In *Proceedings of the 1982 Conference on Human Factors in Computing Systems (CHI '82)*. ACM, New York, NY, USA, 41–44. DOI:http://dx.doi.org/10.1145/800049.801752

[7] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. 2009. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Comput.*

Surv. 41, 1, Article 2 (Jan. 2009), 31 pages. DOI:http://dx.doi.org/10.1145/1456650.1456652

[8] T.R.H. Cutmore, T.J. Hine, K.J. Maberly, N.M. Langford, and G. Hawgood. 2000. Cognitive and gender factors influencing navigation in a virtual environment. *International Journal of Human Computer Studies* 53, 2 (2000), 223 – 249. DOI: http://dx.doi.org/10.1006/ijhc.2000.0389

[9] Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. *LSD-SLAM: Large-Scale Direct Monocular SLAM*. Springer International Publishing, Cham, 834–849. DOI: http://dx.doi.org/10.1007/978-3-319-10605-2_54 This is the full paper of the algorithm.

[10] George W. Fitzmaurice, Shumin Zhai, and Mark H. Chignell. 1993. Virtual reality for palmtop computers. *ACM Trans. Inf. Syst.* 11, 3 (July 1993), 197–218. DOI: http://dx.doi.org/10.1145/159161.159160

[11] George W. Furnas and Benjamin B. Bederson. 1995. Space-scale diagrams: understanding multiscale interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 234–241. DOI:http://dx.doi.org/10.1145/223904.223934

[12] Google. 2014. ATAP Project Tango – Google. http://www.google.com/atap/projecttango/. (2014). website recieved on 22.09.2016 16:52.

[13] Carl Gutwin and Chris Fedak. 2004. Interacting with big interfaces on small screens: a comparison of fisheye, zoom, and panning techniques. In *Proceedings of Graphics Interface 2004*. Canadian Human-Computer Communications Society, 145–152. http://dl.acm.org/citation.cfm?id=1006058.1006076

[14] T.R. Hansen, E. Eriksson, and A. Lykke-Olesen. 2005. Mixed Interaction Spaces–a new interaction technique for mobile devices. *Demonstration at UbiComp* (2005).

[15] Thomas Riisgaard Hansen, Eva Eriksson, and Andreas Lykke-Olesen. 2006. Mixed Interaction Space - Expanding the Interaction Space with Mobile Devices. In *People and Computers XIX The Bigger Picture*. Springer London, 365–380.

[16] Antonio Haro, Koichi Mori, Tolga Capin, and Stephen Wilkinson. 2005. Mobile Camera-Based User Interaction. In *Computer Vision in Human-Computer Interaction*. Springer Berlin / Heidelberg, 79–89. http://dx.doi.org/10.1007/11573425_8

[17] Khalad Hasan, David Ahlström, and Pourang P. Irani. 2015. Comparing Direct Off-Screen Pointing, Peephole, and Flick & Pinch Interaction for Map Navigation. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction (SUI '15)*. ACM, New York, NY, USA, 99–102. DOI:http://dx.doi.org/10.1145/2788940.2788957

[18] Khalad Hasan, Xing-Dong Yang, Hai-Ning Liang, and Pourang Irani. 2012. How to Position the Cursor?: An Exploration of Absolute and Relative Cursor Positioning for Back-of-device Input. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 103–112. DOI:http://dx.doi.org/10.1145/2371574.2371591

[19] Ken Hinckley, Randy Pausch, John C Goble, and Neal F Kassell. 1994. A survey of design issues in spatial input. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*. ACM, 213–222. DOI:http://dx.doi.org/10.1145/192426.192501

[20] Steven Hoober. 2013. How Do Users Really Hold Mobile Devices? http://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php. (2013). website recieved on 30.10.2016 04:13.

[21] Scott Hurf. 2015. How to design for thumbs in the Era of Huge Screens. http://scotthurff.com/posts/how-to-design-for-thumbs-in-the-era-of-huge-screens. (2015). website recieved on 30.10.2016 03:31.

[22] Jeff A. Johnson. 1995. A comparison of user interfaces for panning on a touch-controlled display. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 218–225. DOI:http://dx.doi.org/10.1145/223904.223932

[23] Brett Jones, Rajinder Sodhi, David Forsyth, Brian Bailey, and Giuliano Maciocci. 2012. Around device interaction for multiscale navigation. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*. ACM, 83–92. http://doi.acm.org/10.1145/2371574.2371589

[24] S. Jones, M. Jones, G. Marsden, D. Patel, and A. Cockburn. 2005. An evaluation of integrated zooming and scrolling on small screens. *International Journal of Human-Computer Studies* 63 (2005), 271–303.

[25] Neel Joshi, Abhishek Kar, and Michael Cohen. 2012. Looking at you: fused gyro and face tracking for viewing large imagery on mobile devices. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. ACM, 2211–2220. DOI:http://dx.doi.org/10.1145/2208276.2208375

[26] Susanne Jul and George W. Furnas. 1997. Navigation in Electronic Worlds: A CHI 97 Workshop. *SIGCHI Bull.* 29, 4 (Oct. 1997), 44–49. DOI:http://dx.doi.org/10.1145/270950.270979

[27] Susanne Jul and George W. Furnas. 1998. Critical zones in desert fog: aids to multiscale navigation. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*. ACM, 97–106. DOI:http://dx.doi.org/10.1145/288392.288578

[28] Amy K. Karlson and Benjamin B. Bederson. 2006. *Understanding Single-Handed Mobile Device Interaction*. Technical Report.

[29] Amy K. Karlson and Benjamin B. Bederson. 2007. ThumbSpace: Generalized One-handed Input for Touchscreen-based Mobile Devices. In *Proceedings of the 11th IFIP TC 13 International Conference on Human-computer Interaction (INTERACT'07)*. Springer-Verlag, Berlin, Heidelberg, 324–338. http://dl.acm.org/citation.cfm?id=1776994.1777034

[30] Bonifaz Kaufmann and David Ahlström. 2012. Revisiting Peephole Pointing: A Study of Target Acquisition with a Handheld Projector. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 211–220. DOI:http://dx.doi.org/10.1145/2371574.2371607

[31] Bonifaz Kaufmann and David Ahlström. 2013. Studying spatial memory and map navigation performance on projector phones with peephole interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3173–3176. DOI:http://dx.doi.org/10.1145/2470654.2466434

[32] Frederic Kerber, Antonio Krüger, and Markus Löchtefeld. 2014. Investigating the Effectiveness of Peephole Interaction for Smartwatches in a Map Navigation

Task. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI '14)*. ACM, New York, NY, USA, 291–294. DOI:http://dx.doi.org/10.1145/2628363.2628393

[33] Sunjun Kim, Jihyun Yu, and Geehyuk Lee. 2012. Interaction techniques for unreachable objects on the touchscreen. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*. ACM, 295–298.

[34] I. Scott MacKenzie. 2013. *Human-Computer Interaction: An Empirical Research Perspective* (1st ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[35] Sylvain Malacria, Eric Lecolinet, and Yves Guiard. 2010. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces: the cyclostar approach. In *Proceedings of the 28th international conference on Human factors in computing systems*. ACM, 2615–2624. DOI:http://dx.doi.org/10.1145/1753326.1753724

[36] Sumit Mehra, Peter Werkhoven, and Marcel Worring. 2006. Navigating on handheld displays: Dynamic versus static peephole navigation. *ACM Trans. Comput.-Hum. Interact.* 13 (2006), 448–457. DOI:http://dx.doi.org/10.1145/1188816.1188818

[37] Albert Ng, Michelle Annett, Paul Dietz, Anoop Gupta, and Walter F. Bischof. 2014. In the Blink of an Eye: Investigating Latency Perception During Stylus Interaction. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1103–1112. DOI:http://dx.doi.org/10.1145/2556288.2557037

[38] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for Low-latency Direct-touch Input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 453–464. DOI:http://dx.doi.org/10.1145/2380116.2380174

[39] OCCIPITAL. 2014. Structure Sensor – OCCIPITAL. http://structure.io/developers. (2014). website recieved on 22.09.2016 16:52.

[40] Ji-young Oh and Hong Hua. 2006. User evaluations on form factors of tangible magic lenses. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 23–32.

[41] Michel Pahud, Ken Hinckley, Shamsi Iqbal, Abigail Sellen, and Bill Buxton. 2013. Toward compound navigation tasks on mobiles via spatial manipulation. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*. ACM, 113–122. DOI:http://dx.doi.org/10.1145/2493190.2493210

[42] Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. 2006. Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. ACM, 203–210. DOI:http://dx.doi.org/10.1145/1152215.1152260

[43] Sebastien Pelurson and Laurence Nigay. 2015. Multimodal Interaction with a Bifocal View on Mobile Devices. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction (ICMI '15)*. ACM, New York, NY, USA, 191–198. DOI:http://dx.doi.org/10.1145/2818346.2820731

[44] Roman Rädle, Hans-Christian Jetter, Simon Butscher, and Harald Reiterer. 2013. The effect of egocentric body movements on users' navigation performance and spatial memory in zoomable user interfaces. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*. ACM, 23–32. DOI:http://dx.doi.org/10.1145/2512349.2512811

[45] Roman Rädle, Hans-Christian Jetter, Nicolai Marquardt, Harald Reiterer, and Yvonne Rogers. 2014. HuddleLamp: Spatially-Aware Mobile Displays for Ad-hoc Around-the-Table Collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (ITS '14)*. ACM, New York, NY, USA, 45–54. DOI:http://dx.doi.org/10.1145/2669485.2669500

[46] Roman Rädle, Hans-Christian Jetter, Jens Müller, and Harald Reiterer. 2014. Bigger is not always better: display size, performance, and task load during peephole map navigation. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 4127–4136. http://doi.acm.org/10.1145/2556288.2557071

[47] Byron Reeves, Annie Lang, Eun Young Kim, and Deborah Tatar. 1999. The Effects of Screen Size and Message Content on Attention and Arousal. *Media Psychology* 1, 1 (1999), 49–67. DOI:http://dx.doi.org/10.1207/s1532785xmep0101_4

[48] Michael Rohs, Antti Oulasvirta, and Tiia Suomalainen. 2011. Interaction with magic lenses: real-world validation of a Fitts' Law model. In *Proceedings of the*

SIGCHI Conference on Human Factors in Computing Systems. ACM, 2725–2728. DOI:http://dx.doi.org/10.1145/1978942.1979343

[49] Michael Rohs, Robert Schleicher, Johannes Schöning, Georg Essl, Anja Naumann, and Antonio Krüger. 2009. Impact of item density on the utility of visual context in magic lens interactions. *Personal and Ubiquitous Computing* 13, 8 (2009), 633–646. DOI:http://dx.doi.org/10.1007/s00779-009-0247-2

[50] Michael Rohs, Johannes Schöning, Martin Raubal, Georg Essl, and Antonio Krüger. 2007. Map navigation with mobile devices: virtual versus physical movement with and without visual context. In *Proceedings of the 9th international conference on Multimodal interfaces*. ACM, 146–153. DOI:http://dx.doi.org/10.1145/1322192.1322219

[51] Martin Spindler and Raimund Dachselt. 2009. PaperLens: advanced magic lens interaction above the tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ACM, Article 7, 1 pages. DOI:http://dx.doi.org/10.1145/1731903.1731948

[52] Martin Spindler, Marcel Martsch, and Raimund Dachselt. 2012. Going Beyond the Surface: Studying Multi-Layer Interaction Above the Tabletop. In *Proceedings of the Conference on Human Factors in Computing Systems*. ACM, 1277–1286. DOI:http://dx.doi.org/10.1145/2207676.2208583

[53] Martin Spindler, Martin Schuessler, Marcel Martsch, and Raimund Dachselt. 2014a. Move Your Phone: Spatial Input-based Document Zoom & Pan on Mobile Displays Revisited. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. ACM, New York, NY, USA, 515–518. DOI:http://dx.doi.org/10.1145/2559206.2574777

[54] Martin Spindler, Martin Schuessler, Marcel Martsch, and Raimund Dachselt. 2014b. Pinch-drag-flick vs. Spatial Input: Rethinking Zoom & Pan on Mobile Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1113–1122. DOI:http://dx.doi.org/10.1145/2556288.2557028

[55] Wolfgang Stuerzlinger and Chadwick A. Wingrave. 2011. *The Value of Constraints for 3D User Interfaces*. Springer Vienna, Vienna, 203–223. DOI:http://dx.doi.org/10.1007/978-3-211-99178-7_11

[56] Michael Tsang, George W. Fitzmaurice, Gordon Kurtenbach, Azam Khan, and Bill Buxton. 2002. Boom Chameleon: Simultaneous capture of 3D viewpoint, voice and gesture annotations on a spatially-aware display. In *ACM UIST Symposium on User Interface Software and Technology*. ACM Press/Addison-Wesley Publishing Co., 111–120. DOI:http://dx.doi.org/10.1145/1201775.882329

[57] Brygg Ullmer and Hiroshi Ishii. 1997. The metaDESK: Models and Prototypes for Tangible User Interfaces. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology (UIST '97)*. ACM, New York, NY, USA, 223–232. DOI:http://dx.doi.org/10.1145/263407.263551

[58] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid touch: a see-through mobile device. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 269–278. DOI: http://dx.doi.org/10.1145/1294211.1294259

[59] Katrin Wolf. 2015. *Grasp Interaction with Tablets*. Springer Berlin / Heidelberg.

[60] Ka-Ping Yee. 2003. Peephole displays: pen interaction on spatially aware handheld computers. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '03)*. ACM, New York, NY, USA, 1–8. DOI: http://dx.doi.org/10.1145/642611.642613

List of Figures