# INTERACTING ”THROUGH THE DISPLAY”

## A New Model for Interacting on and Across External Displays

# DISSERTATION

an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

vorgelegt von
Diplom-Medieninformatiker

**SEBASTIAN BORING**

München, den 11. Juni 2010

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

# INTERACTING ”THROUGH THE DISPLAY”

## A New Model for Interacting on and Across External Displays

**DISSERTATION**

an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

vorgelegt von
Diplom-Medieninformatiker
**SEBASTIAN BORING**

München, den 11. Juni 2010

| | |
|---|---|
| Erstgutachter: | Prof. Dr. Andreas Butz |
| Zweitgutachter: | Prof. Dr. Patrick Baudisch |
| Drittgutachter: | Dr. Rafael ’Tico’ Ballagas |

Tag der mündlichen Prüfung: 19. Juli 2010

# Abstract

The increasing availability of displays at lower costs has led to a proliferation of such in our everyday lives. Additionally, mobile devices are ready to hand and have been proposed as interaction devices for external screens. However, only their input mechanism was taken into account without considering three additional factors in environments hosting several displays: first, a connection needs to be established to the desired target display (*modality*). Second, screens in the environment may be re-arranged (*flexibility*). And third, displays may be out of the user's reach (*distance*). In our research we aim to overcome the problems resulting from these characteristics. The overall goal is a new interaction model that allows for (1) a *non-modal* connection mechanism for impromptu use on various displays in the environment, (2) interaction on and across displays in highly *flexible* environments, and (3) interacting at *variable* distances. In this work we propose a new interaction model called *through the display* interaction which enables users to interact with remote content on their personal device in an absolute and direct fashion.

To gain a better understanding of the effects of the additional characteristics, we implemented two prototypes each of which investigates a different *distance* to the target display: *LucidDisplay* allows users to place their mobile device directly on top of a larger external screen. *MobileVue* on the other hand enables users to interact with an external screen at a distance. In each of these prototypes we analyzed their effects on the remaining two criteria – namely the *modality* of the connection mechanism as well as the *flexibility* of the environment.

With the findings gained in this initial phase we designed *Shoot & Copy*, a system that allows the detection of screens purely based on their visual content. Users aim their personal device's camera at the target display which then appears in live video shown in the viewfinder. To select an item, users take a picture which is analyzed to determine the targeted region. We further extended this approach to multiple displays by using a centralized component serving as gateway to the display environment. In *Tap & Drop* we refined this prototype to support real-time feedback. Instead of taking pictures, users can now aim their mobile device at the display resulting and start interacting immediately. In doing so, we broke the rigid sequential interaction of content selection and content manipulation. Both prototypes allow for (1) connections in a *non-modal* way (i.e., aim at the display and start interacting with it) from the user's point of view and (2) fully *flexible* environments (i.e., the mobile device tracks itself with respect to displays in the environment). However, the wide-angle lenses and thus greater field of views of current mobile devices still do not allow for *variable* distances. In *Touch Projector*, we overcome this limitation by introducing zooming in combination with temporarily freezing the video image.

Based on our extensions to taxonomy of mobile device interaction on external displays, we created a refined model of interacting *through the display* for mobile use. It enables users to interact impromptu without explicitly establishing a connection to the target display (*non-modal*). As the mobile device tracks itself with respect to displays in the environment, the model further allows for full *flexibility* of the environment (i.e., displays can be re-arranged without affecting on the interaction). And above all, users can interact with external displays regardless of their actual size at *variable* distances without any loss of accuracy.

# ZUSAMMENFASSUNG

Die steigende Verfügbarkeit von Bildschirmen hat zu deren Verbreitung in unserem Alltag geführt. Ferner sind mobile Geräte immer griffbereit und wurden bereits als Interaktionsgeräte für zusätzliche Bildschirme vorgeschlagen. Es wurden jedoch nur Eingabemechanismen berücksichtigt ohne näher auf drei weitere Faktoren in Umgebungen mit mehreren Bildschirmen einzugehen: (1) Beide Geräte müssen verbunden werden (*Modalität*). (2) Bildschirme können in solchen Umgebungen umgeordnet werden (*Flexibilität*). (3) Monitore können außer Reichweite sein (*Distanz*). Wir streben an, die Probleme, die durch diese Eigenschaften auftreten, zu lösen. Das übergeordnete Ziel ist ein Interaktionsmodell, das einen *nicht-modalen* Verbindungsaufbau für spontane Verwendung von Bildschirmen in solchen Umgebungen, (2) Interaktion auf und zwischen Bildschirmen in *flexiblen* Umgebungen, und (3) Interaktionen in *variablen* Distanzen erlaubt. Wir stellen ein Modell (Interaktion *durch den Bildschirm*) vor, mit dem Benutzer mit entfernten Inhalten in direkter und absoluter Weise auf ihrem Mobilgerät interagieren können.

Um die Effekte der hinzugefügten Charakteristiken besser zu verstehen, haben wir zwei Prototypen für unterschiedliche *Distanzen* implementiert: *LucidDisplay* erlaubt Benutzern ihr mobiles Gerät auf einen größeren, sekundären Bildschirm zu legen. Gegensätzlich dazu ermöglicht *MobileVue* die Interaktion mit einem zusätzlichen Monitor in einer gewissen Entfernung. In beiden Prototypen haben wir dann die Effekte der verbleibenden zwei Kriterien (d.h. *Modalität* des Verbindungsaufbaus und *Flexibilität* der Umgebung) analysiert.

Mit den in dieser ersten Phase erhaltenen Ergebnissen haben wir *Shoot & Copy* entworfen. Dieser Prototyp erlaubt die Erkennung von Bildschirmen einzig über deren visuellen Inhalt. Benutzer zeigen mit der Kamera ihres Mobilgeräts auf einen Bildschirm dessen Inhalt dann in Form von Video im Sucher dargestellt wird. Durch die Aufnahme eines Bildes (und der darauf folgenden Analyse) wird Inhalt ausgewählt. Wir haben dieses Konzept zudem auf mehrere Bildschirme erweitert, indem wir eine zentrale Instanz verwendet haben, die als Schnittstelle zur Umgebung agiert. Mit *Tap & Drop* haben wir den Prototyp verfeinert, um Echtzeit-Feedback zu ermöglichen. Anstelle der Bildaufnahme können Benutzer nun ihr mobiles Gerät auf den Bildschirm richten und sofort interagieren. Dadurch haben wir die strikt sequentielle Interaktion (Inhalt auswählen und Inhalt manipulieren) aufgebrochen. Beide Prototypen erlauben bereits *nicht-modale* Verbindungsmechanismen in *flexiblen* Umgebungen. Die in heutigen Mobilgeräten verwendeten Weitwinkel-Objektive erlauben jedoch nach wie vor keine *variablen* Distanzen. Mit *Touch Projector* beseitigen wir diese Einschränkung, indem wir Zoomen in Kombination mit einer vorübergehenden Pausierung des Videos im Sucher einfügen.

Basierend auf den Erweiterungen der Klassifizierung von Interaktionen mit zusätzlichen Bildschirmen durch mobile Geräte haben wir ein verbessertes Modell (Interaktion *durch den Bildschirm*) erstellt. Es erlaubt Benutzern spontan zu interagieren, ohne explizit eine Verbindung zum zweiten Bildschirm herstellen zu müssen (*nicht-modal*). Da das mobile Gerät seinen räumlichen Bezug zu allen Bildschirmen selbst bestimmt, erlaubt unser Modell zusätzlich volle *Flexibilität* in solchen Umgebungen. Darüber hinaus können Benutzer mit zusätzlichen Bildschirmen (unabhängig von deren Größe) in *variablen* Entfernungen interagieren.

# Preface

This dissertation has been written in partial fulfillment of the requirements for the Degree of Doctor of Natural Sciences in the Department of Mathematics, Informatics and Statistics at the University of Munich (*Ludwig-Maximilians-Universität München*). I was a graduate student at the chair for Media Informatics (*LFE Medieninformatik*) from April 2006 until July 2010 where I conducted most of the research presented in this dissertation.

During my time as a graduate student I also had the opportunity to collaborate with researchers from the University of Oulu, Finland and in particular with Marko Jurmu. We spent 13 months (five in Oulu and eight in Munich) investigating possibilities mobile devices as interaction devices for external displays. That time was a priceless experience for me, broadening my understanding mostly from the technical point of view. However, the prototypes developed in collaboration are part of this thesis with a strong focus on the interaction. Furthermore, I had the chance – for a short but invaluable time – to work together with Patrick Baudisch and his group at the Hasso Plattner Institute in Potsdam, Germany. During that time I gained a different understanding of the aspects regarding this dissertation. Together with the team, I further developed one of the central aspects of this thesis. The different locations as well as the different people helped in deepening my understanding of the research presented in this thesis. This led to changing my research topic significantly over time – shifting from interaction techniques in multi-display environments to interacting with external displays through mobile devices. It is inevitable that – although presented subsequently – several projects are not direct successors of a previous one. Nevertheless, each project led into the design and implementation of the next one to some extent.

Since I was working in different locations and labs with numerous collaborators, it is almost impossible to separate and associate aspects to individual persons. Thus, I chose to write this thesis using the more inclusive scientific plural. At the beginning of each chapter (chapter 4 to chapter 7), I stated the main contributors (i.e., co-authors of papers) of the respective prototype. However, I would like to mention that - as inevitable with larger research projects - more people are related to each of the prototypes and contributed in different ways.

# Acknowledgments

There are many people whom I would like to thank their support as well as their encouragement. Without them, a work like this could have never happened. In the following I mention numerous persons that had a huge influence on (1) the time during this thesis as well as (2) the thesis itself. Nevertheless, I already apologize to anyone whom I forgot to mention. This simply owed by the large amount of persons that had their part in this work.

First and foremost, I deeply thank my supervisor **Andreas Butz** who allowed me to start this whole journey. Besides providing lab space, equipment and a decent salary, he always had a sympathetic ear for all kinds of things. He further encouraged me, watched my moves closely

and, most importantly, helped me to define my research topic as it is presented in this thesis. He also tolerated – although not understood – my wish to work together with the University of Oulu in Northern Finland during the ”cold and dark” winter. Another important role during this dissertation was played by **Patrick Baudisch**. Although we only had a short period time for collaboration, I gained a different view on research and - most importantly - my dissertation topic. He further supported me in any meaning, took his valuable time to discuss things about prototypes as well as my thesis. I can truly say that without his influence I would be a different researcher today. Since collaborating with him and his team was absolutely fruitful I hope to have the chance to work with them again. My third advisor **Rafael ’Tico’ Ballagas** is actually the person I knew the longest. Whenever I had the chance to meet with him (at conferences or at the university), we discussed all kinds of things. When I asked him to be on my committee, he immediately agreed without knowing completely what I was working on. However, his feedback regarding my dissertation was invaluable and certainly helped to improve it. I am already looking forward to meet him (and his wife) again at a conference to share some thoughts or a cold drink. To all the three of you I would like to say a big ”Thank You” for being on my committee, supervising and advising me during this endeavor.

I had the chance to work in a perfect and fun environment in Munich. I would like to thank **Professor Heinrich Hußmann** who always supported my in terms of research and (if necessary) funding. Furthermore, I would like to thank **Otmar Hilliges** who (1) played an important role in my academic life and (2) was the perfect address to discuss (my probably biggest addiction) soccer-related things with. However (as a counterattack to his statement), it is not me going for the wrong team, Otmar. I also would like thank **Dominikus Baur** for sharing an office with me while I was writing this thesis. Writing papers with him was a great addition to my understanding of my own research as he sometimes had a different point of view. In conjunction with Dominikus, I would like to mention **Michael Sedlmair** as a perfect source for discussing research with. Additionally, I enjoyed the time with you at conferences and hope we meet at such again. **Fabian Hennecke** made my life easier with his different sense of humor. Luckily he also shared the same addiction of soccer and was even going for the same team which appears to be a rare thing in Munich. In **Lucia Terrenghi** and **Alexander Wiethoff** I had two persons with a completely different background. I thank both of you for broadening my horizon. I further want to thank **Gregor Broll** for enduring my nagging about NFC especially when having a long-distance flight next to me. In **Heiko Drewes** I found a person that always made me think twice about my projects which in the end lead to a more critical thinking.

I knew several people already during my studies at the University of Munich: **Sara Streng** and **Raphael Wimmer** originally tutored me in the very beginning and later became invaluable persons to discuss research as well as group-related things. I also enjoyed working with **Richard Atterer**, **Paul Holleis**, **Matthias Kranz**, and **Albrecht Schmidt** although they were going into a completely different direction. Although several people left the team and new persons joined, the overall atmosphere was always good. I want to thank all the following persons for being such a great community: **Andreas Pleuß** (for also having an interest in soccer), **Arndt Vitzthum** (for co-supervising the augmented reality class), **Hendrik Richter** (for always making me laugh with his jokes), **Alexander de Luca** (for filling me with all kinds of semi-facts), **Bettina Conradi** and

**Ronald Ecker** (for having discussions in some breaks), **Doris Hausen** (for taking lots of pictures for this thesis), **Yaxi Chen** (for collaborating with me in several projects), **Florence Balagtas-Fernandez** (for sharing thoughts in the coffee kitchen), **Alina Hang** (for always being kind even though I was stressed), and **Max-Emanuel Maurer** (for turning the kitchen into a high-tech market place). With the same importance I would like to thank **Rainer Fink** for being patient with all my wild requests regarding our IT infrastructure and **Anita Szász** for helping me out whenever I had problems related to the university's bureaucracy.

A thesis and its related prototypes are usually not done only by one person. For this reason, I would like to thank the students I worked with during my time here: **Manuela Altendorfer** who built substantial parts of the *Shoot & Copy* prototype, and **Daniel Wolf** for implementing *LucidDisplay*. Without these people, this thesis would have either taken a whole lot longer and would certainly not have been as it is now. Thank you to all of you!

During my stay in Oulu, Finland I had the chance to work with another but equally nice group. Most importantly, I would like to thank **Marko Jurmu** who shares a lot of similar interests and was always the right address for any topic to discuss with. In **Hannu Kukka** I found a person there that further works in a similar direction. I had great discussions with you. And last but not least, **Jukka Riekki** and **Timo Ojala** who allowed me to spend time there as well. The discussions with them certainly helped during developing our prototypes. While I stayed at the Hasso Plattner Institute in Potsdam, Germany, I had the opportunity to work with Patrick Baudisch's students. I would like to thank **Christian Holz** for discussing my research and **Sean Gustafson** for co-authoring one of my papers. I had a great time with you guys.

Naturally, none of this would have happened without the endless support and encouragement from my family - namely my grandmother **Rita**, my uncles **Oliver** and **Torsten**, my (step-)father **Rolf**, and most importantly my mother **Jeanett**. Although sometimes surprised that I want to become a researcher, they never tried to bring me off track. I certainly owe you a lot for keeping up with my plans and ideas until now as well as in the future.

And above all, I want to thank my wonderful girlfriend **Emma**, who joined my life in the middle of this journey and decided to stay although it certainly was not easy for her sometimes. She supported me when I needed it (and that happened often), distracted me from work with her appreciation for the small things in life, and understood when I sometimes did not have time due to deadlines or other work-related things – all that being in a different country. Without her support and understanding (especially when I was stressed out), I would have never been able to finish this thesis. I hope that she continues to be my friend and partner wherever this journey will take us. Emma, you truly deserve the biggest ”Kiitos” that has ever been said.

# TABLE OF CONTENTS

# List of Figures

# I

# Introduction, Motivation and Related Work

Chapter 1

# Introduction

*The scientist is not a person who gives the right answers, he's one who asks the right questions.*

**– Claude Lévi-Strauss –**

Today we interact with both external (i.e., screens from third-party providers) and personal (e.g., mobile phones or PDAs) displays. Their variety ranges from large public displays (e.g., shopping malls or airports), screens built into appliances for special purposes (e.g., cash or ticket machines) or ultra-personal displays we carry with us during the entire day. However, their usage models, their input capabilities, and hence the ways we interact with them, greatly differ based on their size, their placement or their purpose. In this scenario, people are forced to learn and understand numerous input schemes to reach their goal (e.g., memorizing information of interest from a public display, withdrawing cash, or buying a ticket). We aim to overcome this heterogeneity by allowing the same input on various visual displays regardless of the displays' actual input hardware and capabilities. The idea is to use personal, mobile devices and transfer local input onto external displays. In this work, we review limitations of existing approaches and introduce a new model called *through the display* interaction which allows people to interact on various (both reachable and unreachable) displays through their own personal device in the same way as if they would interact on the respective remote screen. At the same time, we address both technical and interaction issues and pay special attention to the mobile nature of the interaction devices.

This chapter reviews different types of displays and describes their interactivity based on both their physical dimensions and their purposes. We further show the potential benefits of decoupling input and output on an external display by using the input capabilities of personal devices instead. Subsequently, we briefly describe the approach taken in this work and give a detailed outlook on the thesis' structure. We conclude this chapter by giving an in-depth statement of the contributions by this thesis.

## 1.1 Categorizing Displays

Displays are described by a variety of characteristics. One obvious way to categorize them is by using their physical properties (see figure 1.1). These include the *physical size* (often specified with the diagonal and aspect ratio), the *pixel density* (measured in dots per inch), and the *virtual resolution* (stated in pixels). It is assumed that increasing the physical size of a display decreases its mobility. Hence, large displays remain stationary while small screens may move throughout the interaction process. Further physical properties are given by their reachability which can have three distinctive states: *fully reachable* (i.e., users can reach each and every single position on the screen), *partly reachable* (i.e., only a sub-region of the display can be reached), and *unreachable* (i.e., the screen is not in the user's immediate vicinity). Again, this property can be linked to the physical size of such displays. For example, a large display (e.g., aspect ratio of 4:3 and a diagonal of 10 feet) cannot be reached fully without any further tools. The last physical property is given by the input capabilities. These include *non-interactive* (i.e., the display is a passive information broadcaster), *semi-interactive* (i.e., the display features a set of specialized input capabilities), and *fully interactive* (i.e., the display can be used directly).

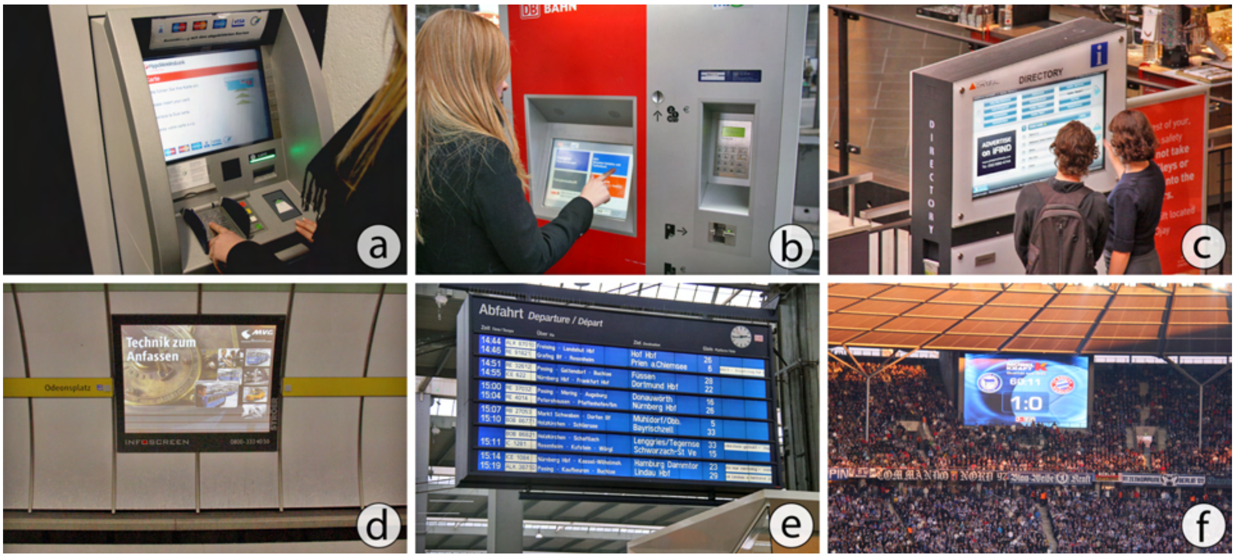


**Figure 1.1:** Displays in our everyday life environment (increasing in size and distance): reachable and interactive displays occur on ATMs with a separate keypad (a), on ticket machines for public transportation featuring a touchscreen as well as an additional keypad and display for payment (b), or on interactive building directories (c). Unreachable displays can be found in subway stations (d), train stations or airports (e), or in stadiums (f).

Besides the physical properties of such displays, they can be categorized by social features or their interaction style. Huang et al. [HM03] introduce the dimensions *type of space* and *number of simultaneous users*. The first dimension describes the environment they are used in. The authors define *public space* (i.e., displays are accessible and visible to everyone), *semi-private shared space* (i.e., displays are only visible by a small group of users which can be controlled), and *private* (i.e., only one or two users have access to the display). The dimension describing the number of users simply states how many users are able to simultaneously interact with the display which we believe is again highly influenced by physical size. For example, a large group of users will not be able to interact on a small personal display simultaneously. Vogel et al. [VB04] looked at the distances between users and large public displays. The authors thereby defined four different phases of interaction: ambient (i.e., information shown with slow updates), implicit (i.e., user is passing by but looking at the display), subtle (i.e., user pauses for a moment while walking by), and personal (i.e., direct interaction with the display). They further show a transitioning model between these phases.

A third level of categorization of displays uses the type of information exchanged between users and the display. The *type of information* is characterized in the same way as the type of spaces: *public information* (i.e., everyone is allowed to see it), *semi-private shared information* (i.e., a controlled group of users could see it), and *private* (i.e., no other than the owner of the information can have access to it). This is strongly influenced and sometimes undercut by the space the displays are located in. For example, entering private information such as a personal identification number (PIN) on a large public display is unwanted due to security reasons. Hence, while being placed in public spaces, automated teller machines (ATMs) use small displays whose input and output capabilities are accessible by only one person at a time.

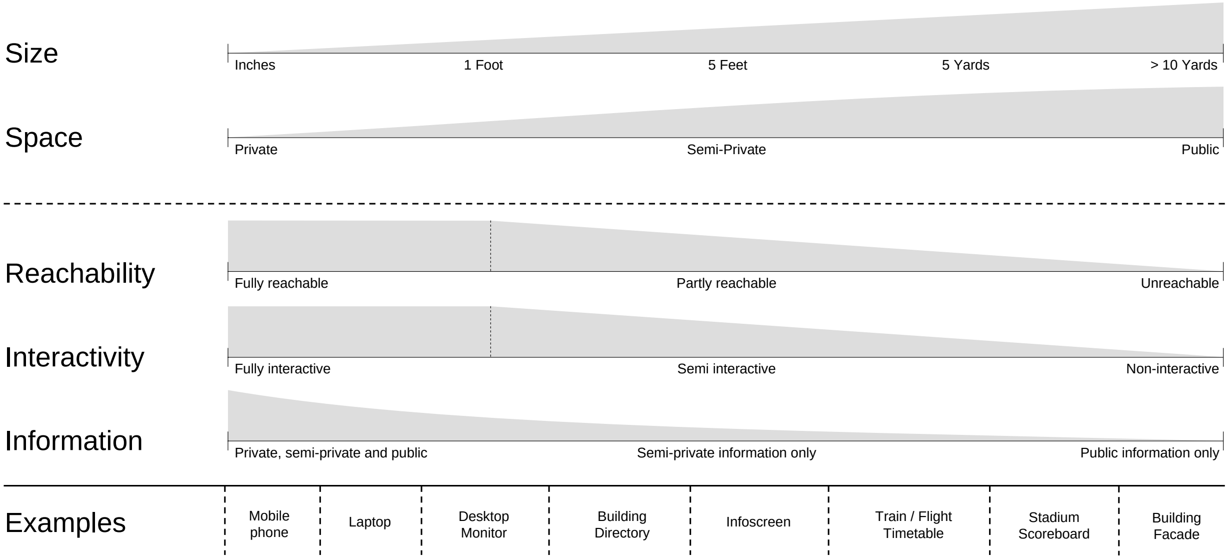


**Figure 1.2:** Dependency of display properties: if the displays' sizes increase, they are placed in public spaces. However, reachability, interactivity, and the types of information that can be used are limited more and more when displays get larger.

Besides the mentioned stationary displays in our environment, these categorizations (as summarized in figure 1.2) can be used to describe personal, mobile devices such as mobile phones, personal digital assistants (PDAs), tablet PCs and laptops. Naturally, their screen real-estates are rather small due to their mobility. Furthermore, they are fully reachable and interactive. Mobile devices are designed for one person at a time and can be used in any kind of space (i.e., *anywhere* and *anytime* [LC08]). Compared to stationary and external displays, users fully trust their own mobile device due to the personal nature of them. They do not fear security threats known from external devices (e.g., a replaced keypad which might capture entered PINs). There is also no hygienic issue (e.g., a greasy touch pad) as mostly no other users than the owner interact with the device. Furthermore, users are highly familiar with the input capabilities as they use their device frequently. The properties of mobile devices suggest that such are one solution to overcome the limitations and drawbacks given by the enormous heterogeneity of external displays.

## 1.2 Combining Personal and Public Displays

Due to their connectivity (i.e., Bluetooth, wireless LAN, and telecommunication networks such as GSM and UMTS), mobile devices easily communicate with external devices and displays [HJK$^+$10]. Similarly, mobile devices come with a large set of interaction capabilities such as numbered keypads, joysticks, accelerometers, and touch screens. The combination of connectivity everywhere and rich input capabilities hence could be employed for interaction on external displays. One big advantage in using input from a personal device is the opportunity to differentiate users. This seems to be a problem domain especially for direct touch-based interfaces (with one exception being the DiamondTouch [DL01]).

The combination of personal, mobile devices and external displays leads to a separation of the user interface: the input is given by the personal device while the output remains on the external display. However, there may be cases in which personal feedback on public displays is unwanted. For example, entering sensitive information such as a password through a pointer and a virtual keyboard on the external display presents a huge security problem. Strictly decoupling input (i.e., on the mobile device) and output (i.e., on the external display) can be enriched with personal feedback on the user's mobile device which is only visible to the respective owner. Hence, the output is split based on its sensitivity: personal feedback is given on the personal device whereas visual feedback that can stay public is shown on the external screen. The overall idea of having the input on the personal device then remains the same.

While the combination of a personal, mobile device and an external display seems to be a promising approach, there are several drawbacks. To interact with the desired display, users need to initiate a connection to it. This usually requires the selection of such a display in a list interface presented on the mobile device. The more screens are available in the environment, the harder it will be for the user to find the correct one. Another approach is to let the application decide which external screen is the desired one. For example, people can initiate a connection to a ticket machine by launching the corresponding application on their mobile device. The application is then responsible for finding the correct (nearby) display. However, this approach requires installing a specific application for each task, which may be unwanted for two reasons: first, users need to trust third-party software. And second, the mobile device is flooded with applications dependent on the amount of tasks. This then converts the problem of discovering a display to the issue of finding the correct application.

Interacting with external displays using personal, mobile devices is slightly different compared to interaction on a regular computer. In general, the properties known from interacting with graphical user interfaces are still applicable. The taxonomy of input devices classifies such as either *relative* or *absolute* and either *indirect* or *direct* [CMR90]. Relative devices use the previously detected position to get the motion delta for calculating the next cursor position. Absolute devices on the other hand use the position at which the input occurred. Direct devices require a tight physical coupling between input and output. Indirect devices are using the input at the location where the pointer is rather than at the position of the device itself. Another distinguishing factor of input is whether it is *discrete* or *continuous*. A discrete device senses input only when requested (e.g., pressing and releasing a button) and further gives only one signal. Continuous devices on the other hand deliver permanent position changes without requiring the user to request it.

Besides these known properties, interacting on external displays introduces new ones. One obvious factor is the distance between screen and user (and the mobile device respectively). The higher the *distance* is the lower is the screen's visibility depending on the screen's size. Furthermore, users need to be able to connect to a display to gain access to its content. These considerations become even more important factors when users want to perform cross-display operations between multiple displays in the environment. In this case, the distances between users and each of the displays most likely varies. Such cross-display operations introduce two additional restrictions: first, both the originating and the destination screen are required to understand the input in the same way. Thus, an interaction technique should not rely on specialized input devices such as digital pens [Rek97, HRG$^+$04]. And second, the correct transition of objects from one display to another (i.e., leaving the first display on the left side and entering the second display on the right) needs to be given to allow for visual continuity. There are other challenges related to large and multiple displays: first, large environments may allow multiple users to interact simultaneously. And second, the privacy of a single user is important when sensitive operations occur. However, these are beyond the scope of this thesis and are only mentioned here for completeness.

## 1.3 Problem Statement

Using the mobile device as input device to interact on external displays can have different characteristics. For example, the mobile device can be operated as a distant mouse allowing the same type of interaction known from traditional operating systems. In this case, the user controls a remote pointer using the sensing capabilities of the mobile device. In contrast, the personal device can also be used in a more direct fashion by pointing at the display (and its content respectively) in order to select items. Ballagas et al. already analyzed the design space of mobile device input [Bal07]. They define four dimensions: (1) absolute versus relative, (2) indirect versus direct, (3) discrete versus continuous, and (4) the dimensionality of the input. The first three dimensions of the design space result from the taxonomy of input devices in general presented by Card et al. [CMR90]. In the following we describe these properties in more detail:

**Absolute versus Relative Input:** Absolute interactions are present if each input point is dependent on the absolute physical location. Relative techniques in contrast are not dependent on the absolute physical location but rather use relative measurements. The computer mouse, for example, only measures the next position based on the difference to the last one. This vector (in mouse coordinates) then gives the translation of the virtual cursor on the display. Touch screens on the other hand have absolute positioning: the input is produced exactly at the position where the finger touched the surface.

**Direct versus Indirect Input:** Direct interactions are defined by showing the feedback exactly where the physical action took place. For example, digital tabletops allow such interactions as the physical action is tapping on an item shown on the screen, which then causes the display to give immediate feedback [Shn83]. In contrast to direct interaction, indirect approaches usually use pointers either virtually (e.g., mouse cursors) or physically (e.g., laser pointers [MBN$^+$02]). Both direct and indirect techniques are visible to the public (if the screen in general can be seen by many users).

**Discrete versus Continuous Input:** Discrete input only allows users to select between certain states of the input device. The most common devices are buttons since they only allow two states: pressed or not pressed. Such buttons can be found in a standard keyboard or the computer mouse. At the same time, discrete input only allows selecting content. In contrast, continuous operations are given when users can freely interact between origin and destination of the interaction. Moving a virtual pointer with a mouse is considered continuous although each position is discrete (i.e., the pixel position). Continuous operations hence allow for moving content from one position to another. Nevertheless, continuity can be achieved by several, sequential discrete operations.

Several solutions have been proposed each of which uses a combination of these properties for interaction on external displays using mobile devices. However, the aforementioned problems resulting from different display arrangements in the environment have only been taken into account partially. For example, some techniques may be restricted by the input capabilities of the involved display (e.g., both displays need to feature the same input modality) while others require users to find the correct display when they use a more generic pointing mechanism (i.e., controlling a pointer on the external display). A third issue arises when users should be able to interact independently of the distance between them and the target display. These drawbacks raise the need for a new generic model of interaction between personal devices and external screens. Such a model should allow users to connect their own mobile device to the remote display in a *non-modal* way. The interaction should hereby also define the target display automatically and hence allow users to interact spontaneously [SIDT02]. As public environments further greatly vary in terms of the displays' arrangement, the model needs to handle different distances (and possibly angles) between the mobile and the external display. And finally, cross-display operations (e.g., transferring content from one remote screen to another) should be possible regardless of the input capabilities of involved displays. It is apparent that the choice of the interaction technique may influence properties of the environment which as follows:

**Limited versus Full Flexibility:** Limited flexibility of the environment either occurs if (1) both involved devices need to understand the same input or (2) the displays cannot be rearranged. In fully flexible environments, on the other hand, cross-display operations, for example, work independently of the involved screens' input capabilities. Furthermore, rearranging displays in this environment would not require a recalibration of such. Ultimate flexibility is achieved if users are able to interact with external mobile displays (e.g., tablet PCs or projections from a steerable projector unit).

**Fixed versus Variable Distance:** Some interaction techniques require users (and their mobile devices respectively) to only have a certain distance to the external screen. If displays in the environment get larger, the viewing distance increases likewise. In such cases, the interaction technique needs to allow for larger distances between user and external display. As several users may have different distances to the display, the ultimate goal is to allow for varying distances, meaning that the distance does not have an influence on the interaction itself.

**Non-Modal versus Model Connection:** Modal connection mechanisms require users to tell the environment with which display they want to interact with. This is usually done by selecting this screen out of a list presented on the display on their mobile device. In contrast, non-modal connection procedures are usually embedded into the interaction itself. For example, the proximity of a mobile device to an external display may already initiate an interaction [SIDT02]. In this case, the users do not have to explicitly connect to a display upfront. Nevertheless, the technical realization of such prototypes may still require a connection a priori (without the user being aware).

Combining the properties of the environment with the previously mentioned characteristics of input devices leads to an extended design space (see figure 1.3). The ultimate goal of this thesis is to allow for an interaction technique that allows for *full flexibility* of the environment as well as a *variable distance* between the user's mobile device and the target display. Furthermore, we aim to enable a *non-modal connection* mechanism (at least from the user's point of view) to overcome the need for actively selecting a target screen. We additionally add two criteria that are of importance in public spaces: first, public displays are relatively large compared to regular computer screens and may allow multiple users to interact simultaneously (i.e., *number of users*). And second, users want to remain private in certain interactions such as entering a PIN (i.e., *privacy of the interaction*). In the scope of this thesis, we consider a technique as private if other users (e.g., bystanders) are not aware of someone's actions. However, the first three characteristics are of greater importance in the scope of this work.

Ideally, one suitable combination of input characteristics can be found that allows for interacting on external displays in the aforementioned style. We first analyze existing techniques regarding the previously introduced criteria based on literature. Subsequently, we examine the environment's properties separately in order to gain a better understanding of how they are influenced by the choice of input mechanisms. Based on the findings, we propose a new theoretical model of interacting with external displays. At the same time, we verify that each of the requested properties (i.e., *full flexibility*, *variable distances*, and *connecting non-modally*) is supported by the model conceptually. This is done by building instances of this model in various stages which are then experimentally evaluated. The expected outcome is the new interaction model as well as a final prototype which support the aforementioned scenario and allows users to interact with external displays in a highly *flexible* and *non-modal* way.

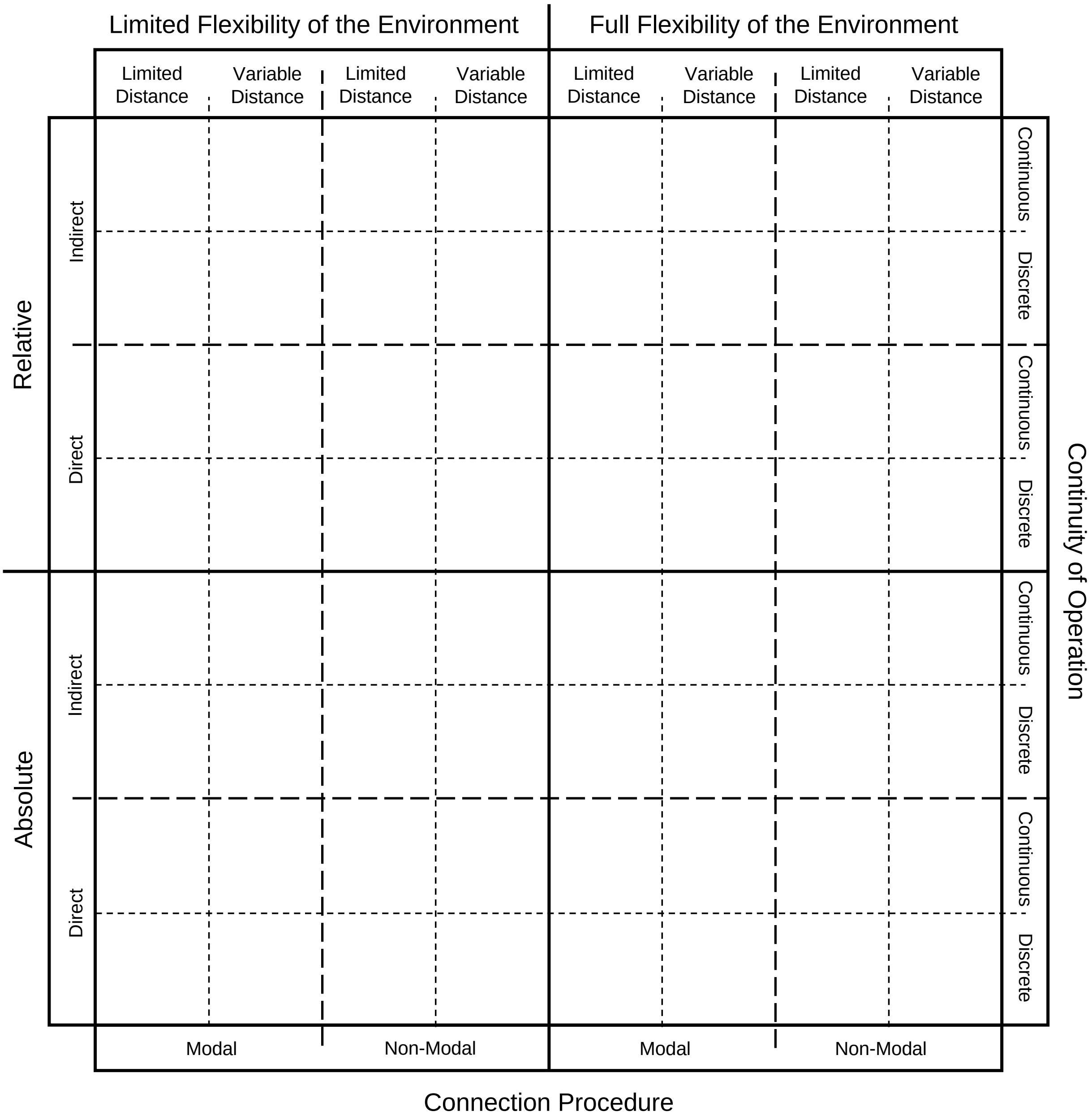


**Figure 1.3:** Spatial layout of the taxonomy for mobile device input on external displays including our proposed extensions (inspired by [Bal07]).

## 1.4 Thesis Overview

This thesis is organized into four parts. In the first part, we motivate the work and give an extensive overview of existing literature in the field of interacting with external displays. At the same time, we investigate the different phases of interaction which each system supports. In the second part, we present two prototypes that are designed to investigate the distance between the mobile device and the external display: with *LucidDisplay* we examine the effects of directly superimposing a personal display on a larger screen to obtain higher accuracy in input and output. From the user's perspective, the connection process was non-modal as the user only had to place the personal device on top of the large (horizontal) screen. Subsequently, we present *MobileVue* to support the connection to a distant display as well as remote content manipulation in a relative and indirect fashion. In contrast to *LucidDisplay*, the connection to the remote screen had to be done explicitly and a priori to the interaction.

This motivated part III of this thesis, namely developing a new model that aims to embed the connection and pointing process on distant displays. This model – called *through the display* interaction – allows users to aim their personal device at the desired screen and interact with the remote content within the live video preview shown on the personal device. We first present *Shoot & Copy* to demonstrate the use of live video in a discrete pointing fashion. We then demonstrate two prototypes that extend our model: (1) *Tap & Drop* enables bimanual pointing with coarse and fine-grained accuracy. We also make use of the personal display either as temporary feedback screen in real-time or as individual content display. (2) *Touch Projector* allows continuous interactions and pays special attention to the mobility of the interaction device and the varying distances between the device and a target display. In the fourth part, we summarize this work, draw conclusions from previous chapters and give an outlook on future research. In the following we give a brief overview of each chapter, its content and its relation to other chapters:

**Chapter 2:** Gives a detailed overview of relevant literature related to our work. We first discuss the different phases of an interaction session - namely *connecting to a display* (on a conceptual level), *pointing at content*, and *manipulating content*. Based on these phases, we particularly focus on connecting to and interacting with external displays, camera-equipped mobile devices, and interaction through video. Despite interacting on one external display, we review cross-display object movements with a special interest in both fixed and dynamic display configurations. We conclude the chapter with a classification of existing approaches regarding our criteria stated before.

**Chapter 3 – *LucidDispaly*:** Demonstrates the effect of a mobile display being superimposed directly on a large external screen. With this prototype we examine the conceptually non-modal connection by simply placing the personal device on the remote display. This prototype does not allow for *variable* distances between the mobile and the external screen. However, both displays are accessible allowing for content transfer across the mobile device's bezels. We discuss different visualization approaches (i.e., the mobile display as independent screen, see-through device, or semi-transparent overlay) and conclude this chapter with an observational study of our initial prototype.

**Chapter 4 – *MobileVue*:** Explores the use of relative and indirect pointing on *distant* displays. As the non-modal connection presented in chapter 3 does not work, we implemented a prototype that (1) allows discovering and selecting a target display, (2) enables users to control a personal pointer on the external screen using the mobile device's sensing capabilities, and (3) allows users to select a tool for further content manipulation. We built each of these phases into a single prototype to observe additionally observe users during an entire interaction session. We conclude this chapter by identifying the weaknesses of this system in terms of our criteria.

**Chapter 5 – *Shoot & Copy*:** Demonstrates how the connection and pointing process can be merged in order to only have one phase. After discussing the theoretical benefits of this approach, the chapter describes how a display, its boundaries and the transformation between the target display and the mobile device are identified purely based on its visual content. Subsequently, we describe how this approach is extended to work across multiple displays. We therefore embed a centralized managing component that distinguishes displays based on their content and routes the input from the mobile device to the correctly identified display. The chapter also includes a proof-of-concept prototype that allows users to retrieve content from a nearby display by taking a picture of it.

**Chapter 6 – *Tap & Drop*:** Extends the prototype presented in chapter 5 to support real-time continuous feedback as well as bimanual input. At the same time, we show how the local display (i.e., the mobile device's screen) can be utilized to show personalized information such as private data or content specific controls. We further extend our model to allow for local input and output in terms of *content selection* and *content manipulation*. We then demonstrate the use of a single tap (as already presented in chapter 3) to allow transferring content between the local layer and the external display back and forth. We conclude this chapter with a proof-of-concept implementation that focuses on these independent content layers.

**Chapter 7 – *Touch Projector*:** Extends our model to support continuous interactions. We first review the necessary changes for continuous cross-display operations (i.e., dragging content from one external screen to another). Secondly, we address the mobile nature of the personal device and present extensions to the original model. These are designed to solve problems that occur from varying distances and unstable camera images which both harm continuous interactions. We present the design and implementation of these extensions. Our evaluation reveals that our extensions (i.e., zooming and temporarily freezing the video image) indeed allow faster and more accurate continuous interactions at variable distances between users and target screens.

**Chapter 8:** Concludes with a summary and discussion of the presented prototypes. We reflect on the properties of the original taxonomy as well as our added dimensions. We arrange our prototypes according to these dimensions to identify which combination of the traditional dimensions influences on the new properties. We then give a brief outlook on future research opportunities that have been opened up by our model.

## 1.5 Thesis Contributions

This thesis contributes to the field of interacting with external displays in several ways. First, we extend the design space of ubiquitous mobile phone input with the aforementioned [BRSB08]. We then develop a new model for interacting with external screens by allowing users to interact with remote content through the live video image shown on their personal, mobile display. Iteratively, we extend this model to allow (1) *non-modal* connections, and (2) a *fully flexible* environment. To meet the criteria of *variable distances*, we further extend the model to allow continuous operations with real-time input and feedback. Subsequently, we use the personal, mobile display not only as input surface but also use its visual output capabilities to give personal feedback. Additionally, we present an alternative that still allows all the presented interactions even though live video is not available. In the following we give a more detailed overview of the main contributions of this work:

> **Extension of the Design Space:** Several approaches exist for interacting with external displays using a mobile device. We give a detailed overview of the distinct phases within an interaction session as well as existing interaction techniques within each phase of a session. This analysis lead to an extension of the design space of ubiquitous mobile device input [BRSB08]. While the original taxonomy deals with the input domain, we extend it by means of the environment the mobile device is used in. The added factors are the *flexibility* of the environment, the *variability* of the distance between mobile device and target display, and the *modality* of the connection process. This design space is then used to create a new interaction model in fully flexible environments at varying distances without the need of explicitly connecting to a target display.
>
> **Non-Modal Connection Mechanisms:** The increasing number of displays in a given environment today requires users have to determine and select the display they want to interact with. When they interact with the display directly (e.g., by using touch input), this connection is established on a conceptual level in a *non-modal* way. However, performing such actions at a distance usually requires them to actively select a display on their mobile phone. Assuming autarkic displays in the environment, cross-display interactions can only be performed by establishing another connection to a second display. By embedding the connection process into the interaction itself, we overcome this limitation from the user's point of view. We propose a model that enables users to aim their mobile device (ands its built-in camera respectively) at the desired target display. They can then interact *impromptu* with the content shown in the viewfinder. At the same time, pointing at the display implicitly establishes a connection to the screen.
>
> **Flexible Environments:** The more displays are present in an environment, the more changes in the displays' arrangement may occur. Especially when mobile displays are part of the environment (i.e., act as external display for a user), the environment is changing frequently and requires an interaction technique that allows for more flexibility. Thus, the *non-modal* connection procedure has to be enabled on all displays in the environment in the same way. We propose an infrastructure that realizes such

non-modal connection procedures on all displays. We further show how this can be used to allow for cross-display operations (such as dragging content from one display to another) without the need for disconnecting from one display and reconnecting to another. This is achieved by employing the mobile device's camera (and its video respectively). Based on this video, the mobile device tracks itself with respect to displays in the environment. The presented infrastructure does not rely on pre-arranged displays or pre-modeled environments and allows for *full flexibility* of the environment.

**Interacting at Variable Distances:** When displays in an environment get larger, the distance between them and users usually increases. Furthermore, multiple users may have different distances to the display. However, mobile devices usually feature wide-angle lenses which only allow for precise interaction at a maximum distance from the external screen. It is apparent that an interaction technique needs to support variable distances to allow for more flexibility of the user. We first allow users to zoom into the video image. With this, they can interact at higher distances with the same precision. The image further needs to be stabilized to counteract slight hand movements caused by, for example, the natural hand tremor. Despite modifying the existing infrastructure to allow real-time interaction for continuous operations, we address the mobility of the interaction device. We present three improvements to allow for a wide range of (meaningful) distances while maintaining operation accuracy and image stability.

# Chapter 2

# Related Work

*We can't solve problems by using the same kind of thinking we used when we created them.*

**– Albert Einstein –**

The work described in this thesis aims to develop a novel model for interacting with external displays using a personal, mobile device. To inform the design space of such a model, this chapter gives a detailed overview of existing techniques. At the same time, we first focus on the three different phases of an interaction session - namely *connecting* to a display, *pointing* at its content, and *manipulating* its content (see section 2.1). We identify the similarities and differences compared to the ones found in traditional GUIs. According to these phases, we review existing research prototypes. We start with the logical *connection* that needs to be established between the mobile device and the external display before any interaction can take place (see section 2.2). Subsequently we review interaction techniques at-a-distance and pay close attention to both bringing content to the user as well as redirecting the user's input (see section 2.3). To gain a deeper understanding of our design space, we then focus more detailed on *pointing* and interaction (i.e., *manipulation*) techniques for external displays using mobile devices (section 2.4). At the same time, we investigate physical pointing (e.g., touching), relative pointing and pointing through the mobile device's camera as interaction capabilities. In the following section, we review the interaction through video for both static (i.e., fixed mounted) and dynamic (i.e., mobile) cameras (see section 2.5). As our model aims to also allow for cross-display interactions, we also review existing approaches. At the same time, we focus on multi-display environments and describe existing techniques for exchanging content between the personal and an external screen or across two external displays (see section 2.6). We then summarize and discuss the presented approaches and arrange them in the extended taxonomy to identify unexplored regions (see section 2.7). The discussion and analysis then leads to our own exploration regarding different *distances* between both devices which will be presented in part II of this work.

## 2.1 The Phases of Interaction Sessions

Interacting with external displays usually requires users to establish a connection to the target screen: first, users need to identify the display they want to interact with (*selection*). Second, they need to establish a wireless connection to it. Subsequently, users can then control the external display and manipulate content. These three steps are similar to traditional graphical user interfaces (GUIs) as described by Fitzmaurice: (1) *acquire physical device*, (2) *acquire logical device*, and (3) *manipulate logical device* [Fit96]. In our context, the acquisition of a logical device maps to the selection and connection to an external display using the personal, mobile display. The second step is to control a remote pointer which in turn allows the acquisition of content on the remote screen. The pointer control can be done using the rich sensing capabilities of today's mobile devices. The subsequent manipulation can then be done by using the personal, mobile device (and its display in particular) as well.

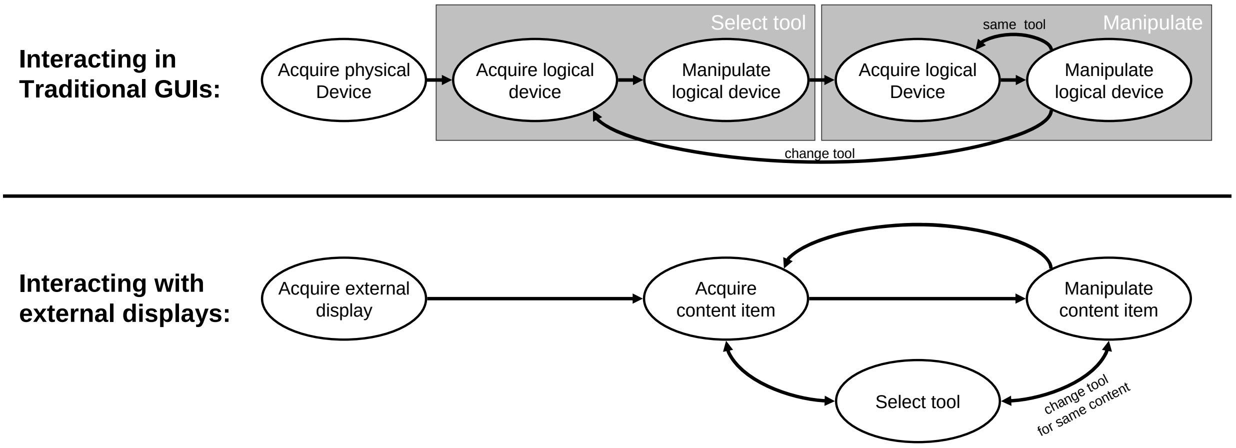


**Figure 2.1:** Interaction phases for both traditional GUIs and interacting with external displays: top shows the interaction phases for traditional GUIs. Users have to select the tool first and subsequently can position the pointer on the content in order to manipulate it. Bottom depicts the phases in our prototype. Users can select their tool (possibly eyes-free) while moving towards the content. This saves one phase and interaction time respectively.

However, before acquiring and manipulating content, users usually need to select a tool first (depending on the task). For example, in Photoshop, users need to select their brush before they can paint with the selected one in the image. Selecting the tool can also be expressed through these phases: acquire the tool and subsequently select (i.e., manipulate) it. Manipulating content required users to perform the second (i.e., acquire content) and third (i.e., manipulate content) twice for the overall goal of manipulating the content itself. The top of figure 2.1 shows the model for traditional GUIs with the extension of selecting a tool.

In contrast to traditional desktop computers, the input device in our prototype (i.e., the mobile device) features a display on its own. This display can be used to select the tool that is needed to perform the manipulation task. By doing so, users acquire the tool by selecting it on screen (i.e., highlighting it. The tool then is selected automatically which means that no further manipulation is needed. We can reduce the overall task of manipulating content by one stage – namely the manipulation of a menu item (e.g., clicking on it). However, users need to move their eyes to their personal display to select the tool. We argue that this movement would occur as well when they need to locate the tool on the external screen. The only difference is that users do not need to refocus as the distance to the remote displays remains constant. Figure 2.1 shows the reduced phases when interacting with an external screen using a mobile device.

On multi-touch devices such as Microsoft's Surface[1], manipulating objects can further be done using two fingers. This avoids the need for explicitly selecting a tool beforehand. For example, if users want to scale an item, they simply use two fingers and perform a pinch-like gesture. In our prototype, this is only realizable by giving two pointers to each user. Using the device's movement as input requires users to switch between these pointers when they want to control both of them. As recent mobile devices feature multi-touch displays (e.g., Apple's iPhone[2]), one could think of a mobile touchpad allowing more than one finger. In this scenario, users can control the first pointer with their first finger and the second with another one. Besides the ambiguous mapping of fingers to pointers, these additional pointers lead to further problems (e.g., screen occlusion, or distinguishing different pointers) as described in the next section.

In the remainder of this chapter, we review existing solutions in more detail for each of these phases separately. We first discuss prototypes that facilitate the connection to an external display (*display acquisition*). Second, we investigate the interaction with content and its manipulation on a distant screen in general. Subsequently, we focus on pointing and interacting on external displays through mobile devices in more detail (*content acquisition* and *manipulation*). Sections 2.5 and 2.6 are detached from this structure as they present special cases related to this work, namely interaction through video and cross-display operations.

## 2.2 Connecting to External Displays

Before any interaction can take place on an external display using a mobile device, users needs to connect their device to the desired target display first. The connection task can be further broken down into two levels: first, on the technical level, both devices (i.e., the mobile device and the target display) need to have a physical connection which can be established via cable or – as a more appropriate approach in this case – via wireless networks such as Bluetooth, wireless LAN or cellular networks (e.g., GSM, UMTS). The second level, on the other hand, is the logical link to the external screen from the user's point of view. This means, users have to tell the system (and their mobile device respectively) to which display they want to connect to. Although both levels are important, we only focus on the second level in the scope of this thesis and rely on existing technologies and solutions from the field of network communications.

[1] *Microsoft Surface*: http://www.microsoft.com/surface/

[2] *Apple iPhone*: http://www.apple.com/iphone/

In this section, we review several existing methods that deal with the explicit connection to an external display. We first focus on side-by-side connections which require users to physically connect (and reach respectively) both devices (see section 2.2.1). Subsequently we give an overview of selecting and connecting to displays that are (temporarily) unreachable. At the same time, we focus on mobile devices to facilitate this process for the user (see section 2.2.2).

### 2.2.1 Connecting at Short Distances

From the user's point of view, placing one device close to another may mean a requested connection between the two. Such pairing procedures can be described as physical coupling. Depending on the hardware used, these can happen in two different ways: first, users have to give input on both devices to allow the establishment of a connection for the underlying infrastructure (*modal* connection). Second, the user places the personal device in a certain way which then initiates the connection implicitly (*non-modal* connection). For both scenarios, numerous prototypes have been proposed, which we will introduce and discuss in this section.

#### *Physically Connecting both Devices:*

A very explicit way of connecting two devices has been proposed by Rekimoto et al. [RAK03]. In *SyncTap*, users have to synchronously press a button on both the devices they wish to connect to each other. This system is also capable of handling simultaneous connection requests of two different users (see figure 2.2a). An even more physical approach is proposed by Hinckley [Hin03]. This system allows users to bump devices together which then causes a virtual desktop spanning across both of the displays. This action then forms one single logical desktop (see figure 2.2b). Similarly to bumping two devices together, Holmquist et al. propose holding together both devices and subsequently shaking them [HMS$^+$01]. If two devices in the environment then recognize the same data from their accelerometers, they can connect to the other device. An interesting extension to such physical approaches is presented in *Stitching* by Hinckley et al. [HRG$^+$04]. Users can span a desktop across displays by drawing a line from one screen to another. This system also considers the spatial relationship between both devices. However, with *Stitching* being the exception, each of these techniques relies on special hardware built into each of the devices or both devices being movable as in [HMS$^+$01]. *Stitching* on the other hand requires the involved devices being capable of recognizing the input in the same way which in turn limits its *flexibility* in terms of device interoperability.

More implicit approaches (i.e., without switching a mode) can be found when devices touch each other (or are very close to) each other. Such systems rely on short-range or direct-touch sensing capabilities given by the target device. Their detection can be grouped into two categories: first, systems use radio frequency to detect nearby devices. And second, optical tracking can be utilized to identify the other device. In the first category, mostly near field communication technologies (NFC) or infrared is used. Rekimoto et al. present a system which connects devices that are in the proximities of others [RAKO03]. The authors use NFC technologies for detecting the proximity between well as the identification (i.e., the network address) of both devices. A standard wireless network is then used for communication between the devices once a connection is established. The approach of initiating the connection by touching another device is common in research [SRHH09]. Tandler et al. present a system called *ConnecTables* which works similarly [TPMT+01]. Both devices act as stand-alone displays but form a single workspace as soon as they are very close to (i.e., touch) each other (see figure 2.2d). These technologies, however, do not allow for higher distances as both displays need to be reachable.

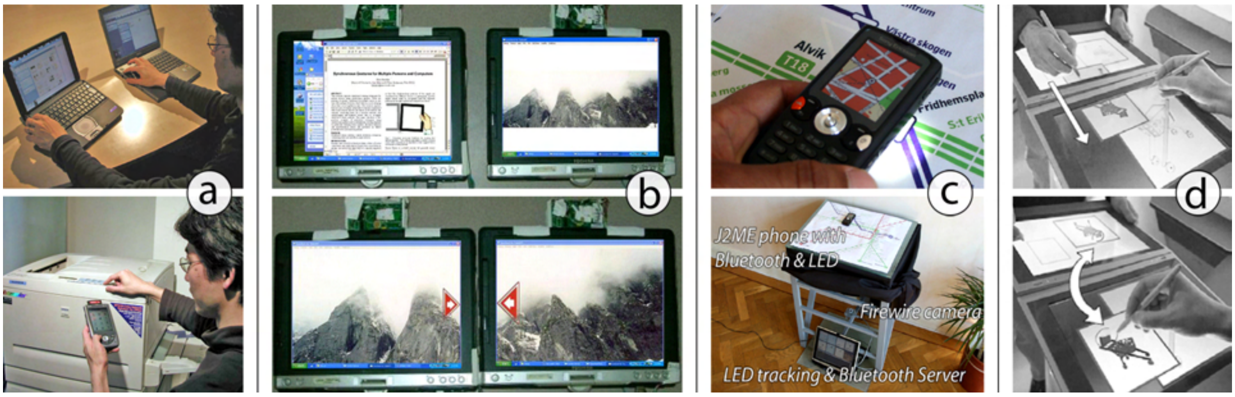

a
b
c
d
J2ME phone with Bluetooth & LED
Firewire camera
LED tracking & Bluetooth Server

**Figure 2.2:** Different systems for connecting displays side-by-side from left to right: a) *SyncTap* [RAK03], b) *Synchronous Gestures* [Hin03], c) *LighSense* [Olw06], and d) *ConnecTables* [TPMT+01].

### *Superimposing Devices on External Displays:*

In case of static, paper-based displays, NFC is widely used today [RRA+06]. In contrast to the aforementioned approaches, such systems do not directly connect to the display but rather to a service via internet [RSA06]. The connection is still initiated through radio frequency identification (RFID) tags placed behind a poster. In this case, the tag contains the address to connect to. Broll et al. present a framework to interact with such paper-based displays [BRP+09]. In on of their case studies, users can buy a ticket for the local public transportation system by touching the corresponding tags for each action. Whenever the user touched a tag, the system communicated to the service associated to the poster (and its tags respectively). Rukzio et al. further employ NFC to connect to external appliances in order to control them through a mobile device [RLC+06]. Sanchéz et al. show how NFC can be used to connect a mobile device to a public display by touching an RFID tag nearby the desired display [SCR07]. In their scenario, users can connect to an external, large display in order to control a slideshow being shown on the screen. The RFID tag does not necessarily be next to the display but has to give a clue to users in way that they know which display they are about to connect to.

If the device's orientation is necessary besides its position, researchers mostly rely on optical sensing methods. Rekimoto et al. demonstrate how laptops can be connected to large, stationary displays (i.e., display walls and tabletops) in the environment [RS99]. Each laptop can be identified uniquely as each of them carries a visual marker. Olwal presents another approach that avoids modifying the mobile device [Olw06]. Their system *LightSense* detects the flashlight LED of cell phones placed on a tabletop (see figure 2.2c). While the system per se does not scale, their tracking solution could be extended to allow the detection of a device-unique blinking patterns of the mobile device's flashlight. *BlueTable* implements such handshakes when users place their mobile device on a tabletop [WS07]. Their system requests a blink on the infrared port of each device in the environment to identify the one that has been placed on a tabletop.

## 2.2.2 Connecting at Variable Distances

Physically reaching the secondary display may be impossible due to its placement, its size, or other limitations. Especially displays meant for a large audience at the same time suffer from these problems. In such cases, the user needs to be able to connect from a distant position. As discussed earlier, NFC may overcome this limitation but a tag has to give clear hint to the user to which display it is linked. Most likely, such tags should be installed close to the display they are linked to. Another option is to visually design RFID tags in a way that it is obvious to the user. However, statically organized RFID tags may fail if the display environment is changing.

### *List and Map Representations:*

If directly reaching the display is not an option, users might select the target device through a *modal* list-based interface. Rukzio et al. show how the list can be populated with displays nearby by using the Bluetooth discovery mechanism [RLC$^+$06]. Similarly, a wireless network could answer with all devices being in the same subnet or connected to the same wireless access point. Krumm et al. proposed a system that allows users to search for nearby persons based on the signal strengths of their respective mobile devices [KH04]. This already narrows down the results that are being presented to the user in order to allow a faster selection of the target device. The latter system is also used for the aforementioned *Stitching* technique as it senses cross-display pen strokes only on nearby devices [HRG$^+$04]. However, such techniques do not scale for environments with a large number of displays or devices.

Another approach is to use a map representation of the display environment as two-dimensional *modal* interface. Such visualizations need to have the correct spatial relationship between all external displays. The interactive workspaces project used maps on PDAs to allow users to gain control over a distant display [JFW02]. Biehl et al. show a similar approach in which users can select the destination screen by dragging content on an iconic representation [BB04]. These systems rely on predefined spatial layouts and would need to be reconfigured whenever a display is moved [WJF$^+$09]. Kortuem et al. provide a system that allows dynamic positioning of potential target devices relative to the user's display to allow spontaneous interaction [KKG05]. In their system, users can see the distance to other devices as well as the orientation between their own device and others. Gellersen et al. then show how such a system can be used to spontaneously interact with other devices in the environment through a personal display [GFG$^+$08].

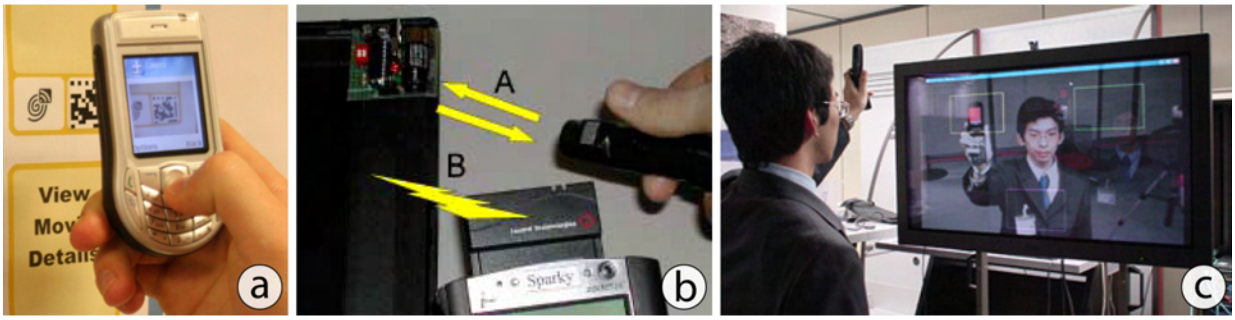


**Figure 2.3:** Different systems for connecting at a distance from left to right: a) Broll et al. use markers to connect to services [BSR$^+$07]. b) Swindells et al. connect two devices with an infrared-emitting pen [SIDT02]. c) Miyaoku et al. use the mobile device's display to connect to and interact with an external display [MHT04].

*Pointing at the Intended Display:*

The more displays are present, the more complex such list- or map-based representations get. Instead of selecting the display in a list, users can also point at it. Rukzio et al. present a system that allows pointing at an external device using a laser pointer [RLC$^+$06]. The external device reacts on the laser and sends control information to the mobile display. Although not being implemented, the identification of the requesting mobile device could be handled by using a pulsed laser with a unique pattern. Swindells et al. demonstrate how users can point on two devices to initiate a connection between them using infrared tags and an infrared-emitting pen [SIDT02] (see figure 2.3b). Miyaoku et al. show *C-Blink* a system that tracks a mobile device's display for both connecting and interacting with the target display [MHT04]. The connection process here is to link the display's movement to a cursor on the screen (see figure 2.3c). However, both systems require the target display to be augmented with detection devices (e.g., cameras, or photoreceptors) which in turn limits the *flexibility* of the environment.

Instead tracking the personal device externally, techniques exist that allow the mobile device to track itself with respect to the environment. Fiducial markers (detectable by the mobile device's camera) can be used to augment the environment [KB99]. Besides detecting the device's orientation and position, markers can convey additional information (e.g., the address of the corresponding display) [BRSB08]. Similarly to NFC-based systems, such visual codes are usually being used to connect to services via internet. Rohs presents visual codes that can be evaluated directly on camera-equipped, mobile devices [Roh04]. These codes can carry information similarly to the industrially developed QR Codes [Den00]. Broll et al. use markers to connect to services in the same way as NFC [BSR$^+$07] (see figure 2.3c). In contrast to Broll and Rohs, Claycomb et al. refined these markers to establish secure connections to external devices [CD06]. While markers seem to be a promising approach for connection purposes, they have several limitations: first, they carry abstract (for humans unreadable) information which raises the same issues as NFC-based techniques (i.e., an explanation which action is performed by a visual marker). Second, they are visually unappealing and may be less accepted by users [JOMS06]. Overall, such markers can be seen as mid-range communication (MRC) in addition to NFC.

## 2.3 Interacting at a Distance

Once users have selected a display, they should be able to *point* at content and *manipulate* it. If displays are placed in a way that they are unreachable, solutions have to be found that allow interaction at a distance. Wallace et al. identified two ways when interacting with remote displays using a local device [WMI08]: first, the users' local input can be redirected (i.e., allowing the local input focus to be transferred to distant displays). And second, remote content can be brought to the user's device to manipulate it locally. The authors state that redirecting input will only work sufficiently when the target display is not occluded during the interaction. They further claim that content redirection may be less susceptible than redirecting input. Another problem is that input redirection requires a line of sight to the display which may not always be given. For example, people sitting around a tabletop may face a wall display with their back. For such cases, content redirection will outperform input redirection. Reaching content at a distance is often realized by using special input devices or mechanisms such as laser pointers [MBN$^+$02]. We categorize such techniques as input redirection since they transfer the users' input to distant screens. In this section we first focus on content redirection (see section 2.3.1). Subsequently, we describe solutions that allow redirecting the input (see section 2.3.2).

### 2.3.1 Redirecting Content to the User

Redirecting content from a (possibly distant) external display to a user's mobile device can have several characteristics: first, the entire display's content can be shown on the mobile device. Second, users can select a certain region of interest within which they wish to interact in. These regions can either refer to a single information item or a sub-region of the entire display. And third, the information can be brought to users in a time-multiplexed fashion, potentially causing delays. In this section we give a brief overview of existing systems in each of these categories.

*The World-in-Miniature Representation:*

The simplest form of bringing content to the user is to show a miniature representation of the target display. This technique is often referred to as world in miniature [SCP95]. Within this view, users can interact with remote content on their local display. From the user's point of view the content is being redirected. However, from the technical perspective, each input occurring in the local miniaturized view is redirected to the target display, causing it to change its visual content. These changes are then also updated on the local representation. One major drawback is the degree of miniaturization needed to allow both local and remote content being shown. Hence, the control-display (CD) ratio (i.e., the ratio between input movement and target movement) is decreasing when the local display requires higher size reductions. While an interactive tabletop in combination with a wall display is rather uncomplicated, the usage of a mobile device together with a large display will harm the interaction efficiency.

Especially if both devices differ in size (i.e., the local display is much smaller than the target screen), additional techniques have to be found to allow high precision input. One common issue is the *fat finger problem* when touch devices are used. If the screen is highly miniaturized already, the user's finger occludes significant portions of the remote display's content. Vogel et al. showed how shifting the content can help to increase the targeting accuracy of finger input for small targets [VB07]. While the input is made more accurate, the high visual capacities of large displays cannot be reproduced on small screens. To address this, such systems use local magnifications of the content region of interest. These techniques partially distort or remove content surrounding the area of interest [LA94]. The distortion is further dependent on the factor of magnification needed which in turn correlates with the difference in screen sizes of both displays. Hence, such approaches only partly reduce the problem.

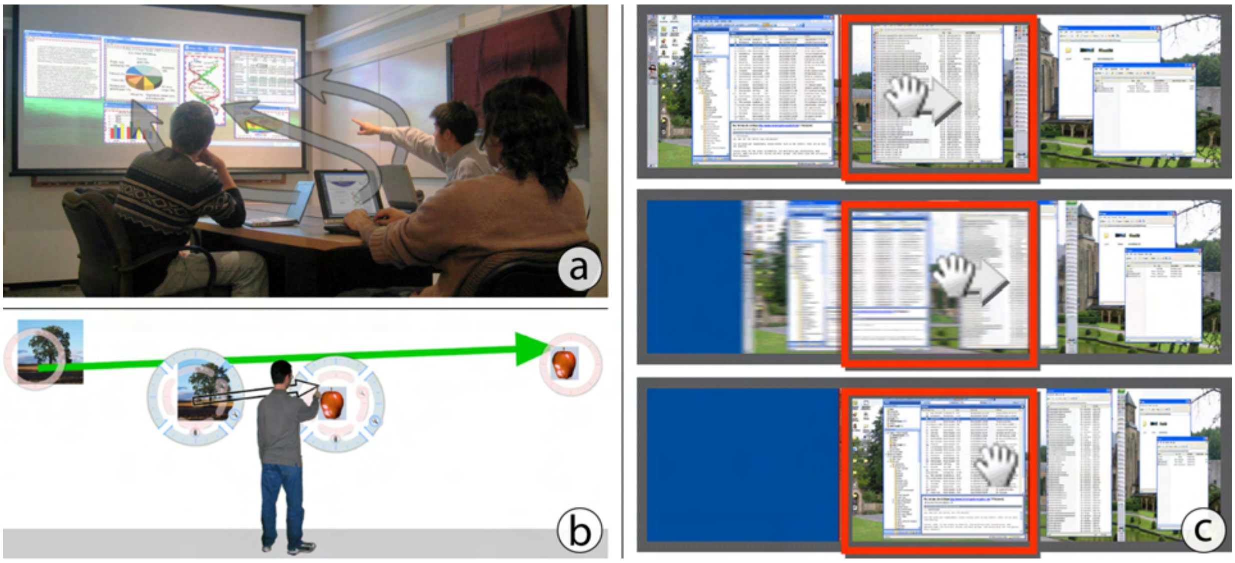


**Figure 2.4:** Bringing content to the user: a) Select sub-regions of desired content with *WinCuts* [TMC04]. b) Bridge large distances using *Frisbee* [KFA+04]. c) Dragging the entire content toward the user similar to a *Tablecloth* [RCB+05].

### *Selecting Regions of Interest:*

Instead of showing the entire remote screen on the local display, users can select regions of interest (i.e., parts of the remote screen's content) a priori. The simplest form is a shared space holding documents and applications as shown in *IMPROMPTU* [BBB+08]. Tan et al. present *WinCuts*, a system that allows users to cut out content of a remote screen for subsequent local manipulation [TMC04]. If the selected content area is smaller than the local device's screen, no miniaturization occurs (see figure 2.4a). The CD ratio remains constant allowing for high-precision manipulation. Repositioning the current cut can then be done using the *Tablecloth* technique [RCB+05]. Users simply drag the entire content until the new region of interest is shown (see figure 2.4c). Consequently, the cut moves in the opposite direction of the drag operation. *Radar Views* allow similar kinds of interaction [NASG05]. They present a sub-region of the target display in which users can interact. The concept of *Frisbee* works likewise [KFA$^+$04]: in this system, users interact with a local representation of distant content (see figure 2.4b). The view of distant content can be moved both locally (i.e., the lens is repositioned, the content remains) or remotely (i.e., the position of the lens remains, but the shown distant content is moved). The authors also propose a linked version in which remote content movement corresponds to the local motion of the lens. In both systems the context of the selected region is not shown during the interaction. This potentially may harm the interaction or lead to zooming operations if users want to get an overview of the current region.

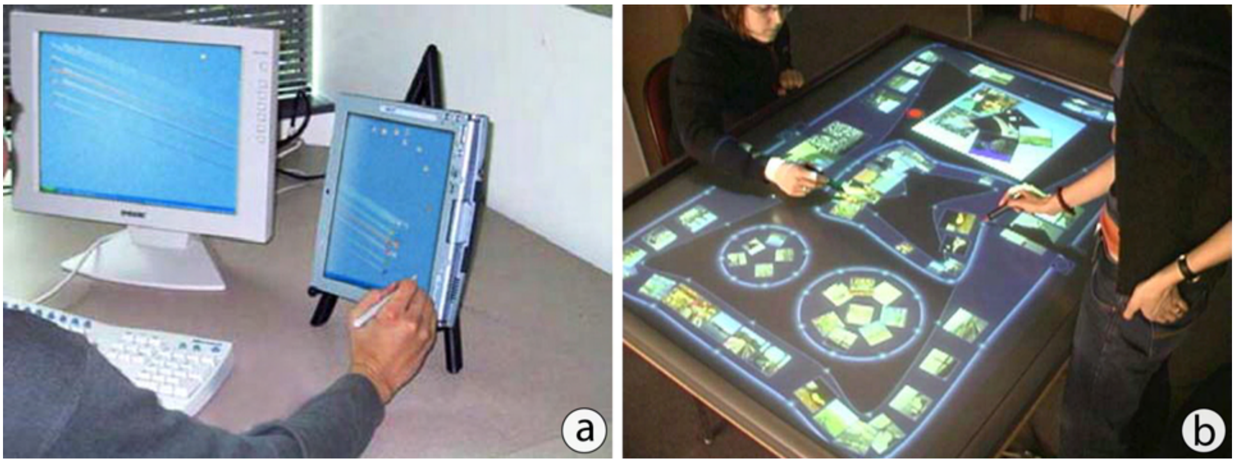


**Figure 2.5:** Spatial versus temporal content movement: a) Accessing content by dragging towards it with *Drag-and-Pick* [BCC$^+$03]. b) Time-multiplexed content movement on a large surface in predefined *Interface Currents* [HCS06].

Indicating a direction can bring content of interest to the user. Baudisch et al. present *Drag-and-Pop* which allows users to drag an item toward possible target icons (i.e., icons that can handle the dragged item) [BCC$^+$03]. Similarly, users can select items by moving the cursor in the direction of target icons using *Drag-and-Pick.* The icons are warped to their temporary location to give a hint of their origin (see figure 2.5a). When used from a personal display the spatial arrangement of displays in the environment (and especially the relationship between mobile and distant screen) needs to be known during the interaction. To avoid the need of positioning methods, Collomb et al. present *Push-and-Pop* [CHBL05]. Their system combines the aforementioned *Drag-and-Pop* with the *Push-and-Throw* method [Has03]. When users start to drag an item, the system responds with potential target icons regardless of their direction. Hence, this approach works when invoked on a mobile device without knowing where it actually resides. However, as discussed in section 2.6, users need to be aware of the icon's origin.

### *Time-multiplexed Content Redirection:*

In contrast to the discussed approaches, content can also be brought to users in time-multiplexed ways (see figure 2.5b). In such systems, content items are constantly floating on given paths. At the same time, the content passes by each user. Hinrichs et al. presents an implementation on a tabletop computer [HCS06]. While their system only uses one display as workspace, the proposed *Interface Currents* can also float across multiple displays. Users are able to reach content that would otherwise be distant when it is passing them. Spreading such interfaces across multiple screens requires the spatial arrangements of displays in the environment to be known a priori. Only this ensures the logical routing of information across displays.

Content shown on a distant display can also be brought to users if it is of interest for them. Rukzio et al. present the *Rotating Compass*, a system that presents navigational information to users [RSK05]. A large (projected) display shows a simplified version of a compass with the four cardinal points while a compass needle is rotating. If the compass is pointing into the correct direction for a certain user, the mobile device of this person vibrates. This notifies the user that the currently shown information (i.e., direction) is the correct one. While this system does not transport information to the user's mobile device from a technical point of view, it still conveys information to the user by linking both the external display and the mobile device.

## 2.3.2 Redirecting Input to Reach Distant Content

Instead of bringing content to the user, local input can be redirected to distant displays. In *Courtyard*, Tani et al. combine shared large screens with individual personal displays [THY$^+$94]. They especially focus on transferring mouse and keyboard input between these two devices. Users can move the mouse pointer between both screens as if they were one canvas. When the pointer leaves the individual display, it continues to travel proportionally correct (i.e., if it leaves the personal screen in the middle of the top border, it appears in the middle of the bottom border on the large display). With this, the correct spatial layout of the displays does not need to be known. This might cause the pointer to jump requiring users to find their pointer on the large screen again. Baudisch et al. address this with *Mouse Ether* by applying appropriate transformations to mouse move events when crossing display boundaries [BCHG04]. Besides the displays' spatial arrangement, the physical dimensions (e.g., the displays' bezels) need to be known upfront.

### *Extending Input Devices Beyond the Local Display:*

Multiple displays can be joined to form a single logical one. In such scenarios, the mouse pointer is associated with the entire display continuum instead of being bound to one screen. Conceptually, this scenario is comparable to multiple monitors attached to a single computer. Johanson et al. present *PointRight*, which allows users to move the mouse pointer across multiple displays (and in their scenario multiple computers) [JHWS02]. In their scenario, the displays' arrangement is preconfigured to allow this kind of interaction. To allow a more flexible setup of displays, Rekimoto et al. use visual markers attached to mobile devices [RS99]. In their scenario, users can extend their personal cursor to adjacent displays. The *Anchored Cursor* technique shows a line from the current cursor position back to the originating device. In this way, others can observe to whom the pointer belongs (see figure 2.6a). In both systems, however, the user's perspective is not taken into account. Nacenta et al. solve this by obtaining the users' head position in three dimensions [NSC$^+$06]. In their system, the display configuration is known and maintained as a 3D model as well. Their approach stitches multiple displays based on the user's perspective allowing for a correct cursor transition between these screens (see figure 2.6b). Brumitt et al. present a different approach to track persons in intelligent environments [BMK$^+$00]. Similarly to *Perspective Cursor*, their approach allows perspective-based interaction techniques.

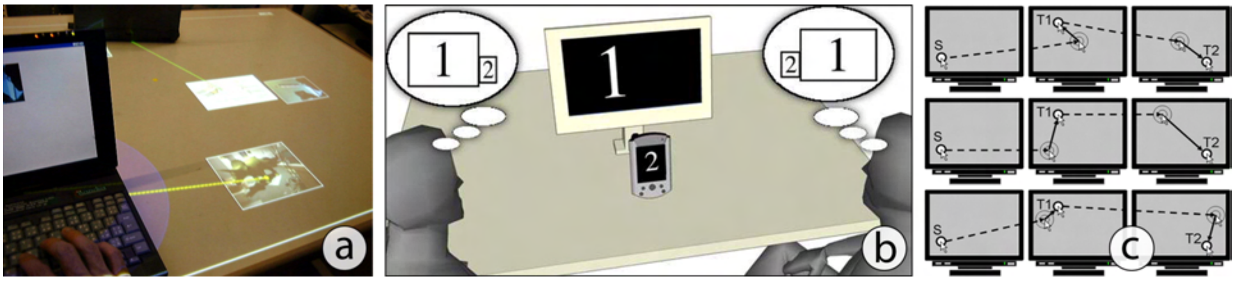


**Figure 2.6:** Transferring controls to distant displays: a) Extending the reach of a mouse pointer beyond the display boundaries [RS99]. b) The perspective problem for different users based on their position [NSC$^+$06]. c) A mouse pointer jumps from one screen to the next one by pressing a certain button [BF05].

Moving the cursor at long distances (and across displays) causes both reduced visibility at high speed as well as longer traversal times. To address the first case, Baudisch et al. present *High-density Cursor* which adds cursor images between the actual cursor positions [BCR03]. The problem of high traversal times can be address by letting the cursor jump from one display to another. Benko et al. present *Multi-Monitor Mouse*, a system that allows users to invoke a jump of the cursor from one screen to another [BF05]. The purpose of their system is to decrease the traversal time of a cursor across multiple displays (see figure 2.6c). Users are able to let the cursor jump to the next screen by either keyboard switches or pressing a button on the mouse. To identify the cursor at its new location, "sonar" circles are displayed at the destination. Warping the pointer further avoids the need for knowing the display arrangement [BF07].

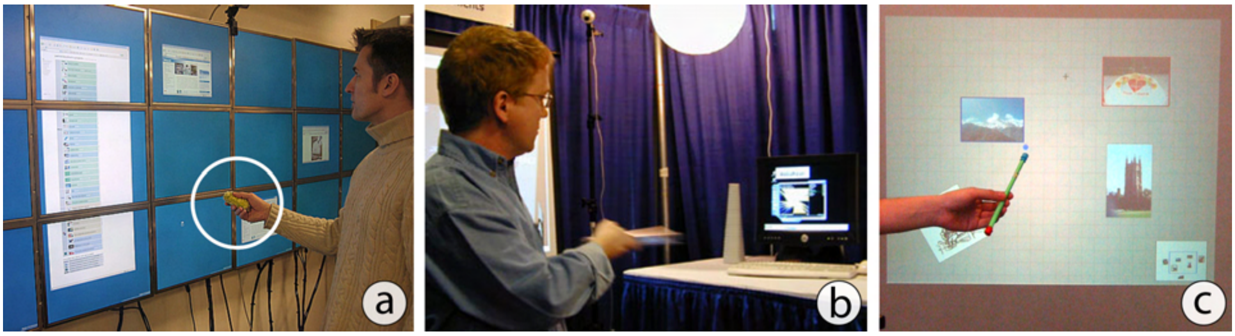


**Figure 2.7:** Controlling a pointer on distant displays: a) *Soap* controls a mouse pointer in mid-air [BSW06]. b) The *XWand* can be used to point at distant screens further allowing gestures to control the content [WS03]. c) *VisionWand* uses two cameras to detect both ends of the wand to obtain its position and orientation [CB03].

### *Specialized Input Devices:*

Specialized devices can be used to control pointers at a distance. Baudisch et al. present Soap which resembles relative and indirect pointing [BSW06]. Instead of pointing relatively, several systems exist that allow absolute pointing at a distance. In traditional presentation scenarios, laser pointers are used to highlight important information on a screen out of the presenter's reach [KM98]. Similarly, Olsen et al. describe the camera tracking of laser points and give an explanation of possible interaction techniques [ON01]. To allow three-dimensional gestures, *XWand* is equipped with magnetometers and accelerometers [WS03]. The wand's position is tracked using computer vision by detecting a flashing infrared LED with two different cameras (see figure 2.7b). The *VisionWand* system avoids the need for complex sensory by only using two cameras [CB03] (see figure 2.7c). Parker et al. show how wand-like techniques can be used to reach distant objects on a tabletop system [PMI05]. All these systems display a pointer (e.g., the laser's spot) on the remote screen and further require users to identify their own one. The difficulty of locating ones own pointer increases with the number of simultaneous users [DC02]. Furthermore, fatigue levels increase when holding a device in mid-air.

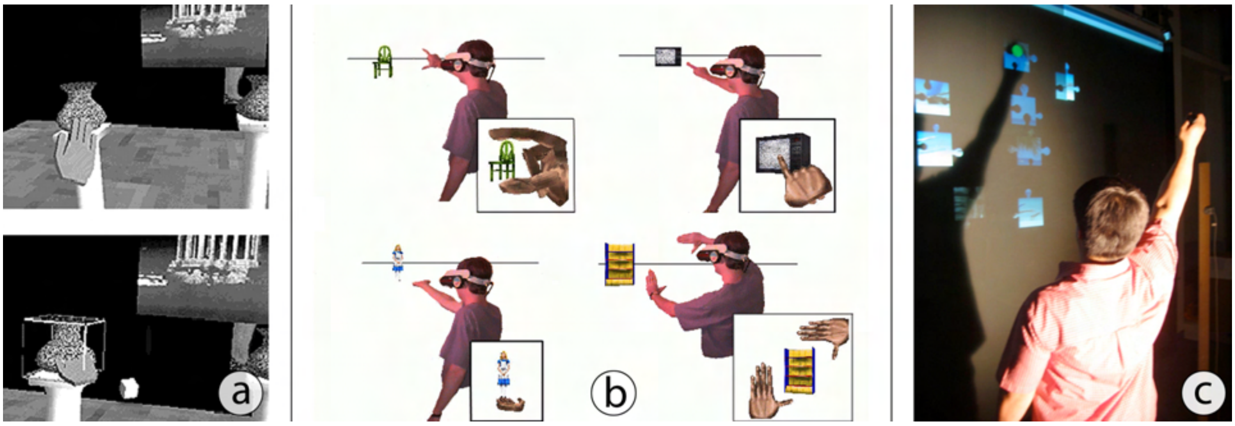


**Figure 2.8:** Techniques for reaching distant content in augmented and virtual reality: a) The *Go-Go* technique extends the user's arm in virtual reality applications [PBWI96]. b) *Headcrusher* allows users to manipulate content at a distance in their own perspective [PFC+97]. c) With *Shadow Reaching* users can reach distant content by casting a shadow [STB07].

### *Selecting Content in Augmented and Virtual Reality:*

In augmented and virtual reality, the problem of selecting and manipulating content exists as well [HPGK94]. Early systems use a technique called *Ray-Casting* which extends the user's finger by a ray of light to reach distant objects [BH97]. Poupyrev et al. allows users to grow their arms interactively within the virtual world [PBWI96]. Their technique uses a non-linear mapping between the real hand's position and the virtual one (see figure 2.8a). As the tracking is based on a user-centered coordinate system, the maximum distance to reach an object is limited due to physical constraints. Pierce et al. present *Headcrusher* to allow a more direct interaction with distant objects [PFC+97]. In their system users can place distant objects between their thumb and index finger (see figure 2.8b). The system further allows several gestures such as lifting an object (i.e., placing the whole hand underneath the object and subsequently move upwards), or framing (i.e., use both hands to frame an object). *Shadow Reaching* allows users to access distant content by casting a shadow onto the external display [STB07]. The shadow is cast by a light source placed behind users (see figure 2.8c). However, both systems require users to hold their arms in mid-air which may increase fatigue.

## 2.4 Mobile Devices as Interaction Devices

Recently, mobile devices have been used to point at distant screens and objects. In contrast to previously described approaches, mobile devices feature their own screen allowing personalized output. The personal display can be used to augment physical objects or static displays such as paper-based maps or posters. Additionally, they have several sensing capabilities enabling a rich set of interactions. These may include a keypad, a touch screen, accelerometers, electronic compasses, a GPS receiver, a camera, or an NFC reader. Combined with their connectivity (i.e., wireless LAN, GSM, UMTS), local input can be transferred to distant screens. Similarly, the personalized output can be communicated back to the mobile device. In this section, we review existing techniques of pointing on remote screens using a mobile device. We first describe how mobile devices can be used to physically touch an external display (see section 2.4.1). Subsequently, we review the use of a mobile device as relative pointing device to control a pointer (see section 2.4.2). As this work focuses on mobile devices being aware of their position to allow for more *flexible* environments, we further present existing systems that make use of spatially aware handheld devices to allow handheld augmented reality (see section 2.4.3).

### 2.4.1 Touching Displays Physically

The most direct approach to interact with an external display is to touch it directly. Doing so with bare fingers is a common way of interacting digital information on large surfaces such as DiamondTouch [DL01], or SmartSkin [Rek02]. Furthermore, there is also a broad set of techniques that work with static displays such as paper-based maps or large posters. These techniques usually rely on NFC which was first used in combination with real-world objects by Want et al. [WFGH99]. They augmented objects with RFID tags in order to interact with associated services. For example, they placed RFID tags in books to link the physical document with related electronic documents that can be found online (and be read on the mobile display).

#### *Accessing Information through NFC:*

Paper-based maps are static displays without the opportunity to change their visual output. Reilly et al. proposed to augment such maps with RFID tags to allow users to obtain additional information for specific locations of interest (e.g., sights in a city) [RWDBI05]. They were able to show that participants completed tasks faster when using a marked-up map compared to maps without markup. The attention shifts between the personal device and the map increased when markup was used [RRA+06]. A further identified problem was that the personal device occludes a quite large area of the map (see figure 2.9a). Users thus need to move the personal device if they want to select something close to their current location. In field studies, Geven et al. found that novice users have problems in initiating the interaction [GSF+07]. The authors further identify challenges for designers of future NFC-based interactions in the real-world.

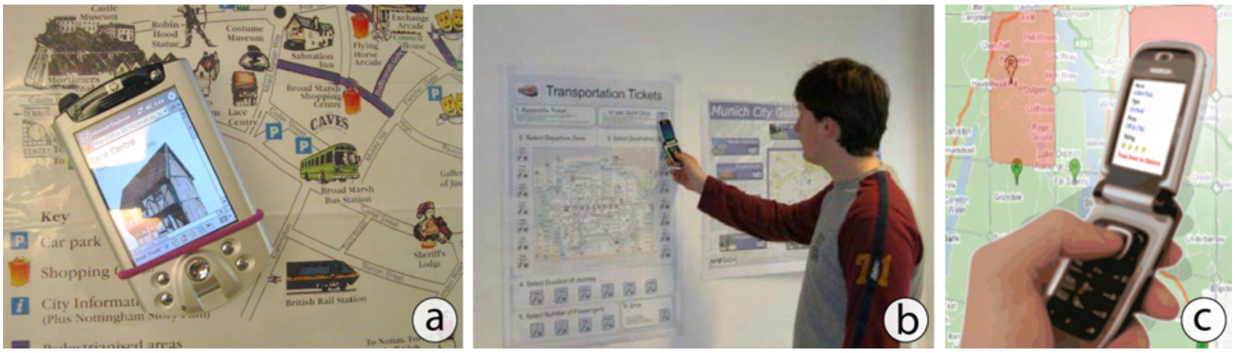


**Figure 2.9:** Touching displays physically for interaction: a) *Marked-up maps* allow mobile devices to display digital content associated to a position on the map [RRA+06]. b) A user *collects* data from a poster [BHP+08]. c) Multiple tags with a front projection to allow for two-dimensional input [HR08].

The usability of NFC-based interaction has been researched widely. Mäkelä et al. conducted a study on user perceptions of both RFID and visual tags [MBGH07]. Their results show that users understood such tags to carry direct information rather than acting as references to external sources. This potentially reduces usability due to misconceptions of such interaction styles. O'Neill et al. further found that novice users are faster in reading 2D visual barcodes than NFC tags [OTGW07]. The visual codes showed no improvement over time whereas the users' performance increased significantly for NFC. Häikiö et al. investigated the use of touch-based interfaces for elderly users [HWI+07]. Their results show that such interfaces are helpful for elderly users as they seem to overcome potential physical or cognitive weaknesses.

### *Discrete versus Continuous Interactions:*

Instead of selecting a single tag, NFC can also be used for multi-tag interactions. Such interactions are using several tags in combination to accomplish one task. Broll et al. present *Collect & Drop*, a system allowing users to buy tickets for a movie theater or the public transportation system [BHP+08]. Their system uses multiple tags for both *DataItems* (e.g., number of seats, movie title) and *ActionItems* (e.g., buy a ticket). The mobile device's display presents an overview to users during the interaction process (see figure 2.9b). They can follow which items have been selected already. In a subsequent study, the authors found that designing an explicit Start Tag helps (especially novice) users to initiate the interaction with these paper-based interfaces [BKHB09].

They further identified general issues when interacting with RFID tags, such as holding the mobile device too far away from the tag, or touching the tag with the wrong side of the device. Another issue was that participants touched tags not long enough to allow the system to read it.

Besides employing NFC tags for discrete interactions (cf. [SRHH09]), Hardy et al. used multiple tags arranged in a matrix to allow continuous interaction [HR08]. Their system uses a projector to display digital information on the grid of markers overlaid by a sheet of paper (see figure 2.9c). The authors compared NFC against direct finger interaction and remote interaction (i.e., using the phone's local input) for target acquisition. They found that their system *Touch & Interact* is comparable to a regular touch screen while being significantly faster than remote input techniques. The matrix also allows drawing a rubber band around items in a continuous fashion. The recognition speed of NFC is not yet fast enough to allow real-time interaction. Another limitation is the low resolution of such a matrix. Permanently tracking the device instead of using low resolution regions seems to be a solution to the low responsiveness [Olw06].

## 2.4.2 Pointing and Interacting at a Distance

If the target display cannot be equipped with NFC technology or is not fully reachable by users, other approaches have to be used to point and interact on them. One common way is to show a personal pointer on the external display which can then be controlled using a mobile device. The *Pebbles* project was one of the first systems allowing users to relatively point with mobile devices [MSG98]. The system allowed both text entry and relative pointing on (possibly distant) displays. McCallum et al. allow both absolute and relative positioning on a touch-enabled device [MI09]. A single tap results in absolutely positioning the cursor on the remote display's screenshot shown on the personal device. Sliding the finger on the touch screen causes the cursor to move relatively as known from regular touch pads (see figure 2.10a). When multiple users are interacting on the same external screen through their PDA, they have to take turns as the public display only features one mouse pointer. From the viewpoint of single display groupware, every user participating in the interaction should have a personal pointer [TG04].

### *Discrete Key-based Input:*

Several mobile devices have a keypad and mostly feature an additional joystick (or at least arrow keys). These discrete controls can be used to control a pointer on remote displays. Silfverberg et al. investigate the use of isometric joysticks as pointing devices [SMK01]. They found that such devices are comparable to traditional relative and indirect pointing techniques such as a laptop's touchpad or a computer mouse (see figure 2.10b). Selecting the acquired target by pressing the joystick directly introduced high error rates. The authors suggest that adding a secondary button for selection purposes might decrease the error rate significantly. The project *Blinkenlights* uses the arrow keys of mobile devices to allow users to play the classical Pong game on entire building facades [Met01]. Each window represented one pixel causing a low resolution (see figure 2.10c). Pressing and releasing the joystick in the desired direction (i.e., up or down) caused the Pong paddle to move accordingly by one pixel. Hence, this interaction resembles discrete input.

### *Input based on Acceleration Sensors:*

Modern mobile devices are equipped with sensors (e.g., magnetometers, accelerometers, electronic compasses) to measure orientation with respect to their local coordinate space. *Rock'n'Scroll* uses accelerometers to let users scroll through a photo library on the device by tilting the device in the desired direction [Bar00]. Similarly to Rekimoto's approach, the system allows to navigate a large two-dimensional space on the local display [Rek96]. Such tilting techniques, however, can also be employed to control remote pointers as shown by Dachselt et al. [DB09]. Most commonly they are used in gaming applications as known from the Nintendo's *Wii* controller [Nin06]. Vajk et al. use mobile devices to control cars on a remote display [VCBE08]. By tilting the device, the virtual car moves in the tilted directions (see figure 2.10d). Tilting a device can have two different meanings of control: first, the tilting angle is used to determine the pointer's position (cf. [BF98]). And second, the device's inclination determines the pointer's acceleration in the tilted direction.

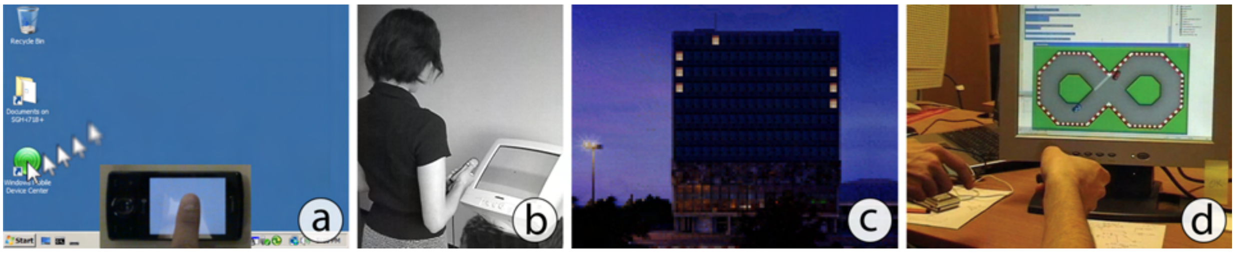


**Figure 2.10:** Pointing at a distance using mobile devices: a) *ARC-Pad* allows absolute and relative positioning on remote displays [MI09]. b) A user controlling a remote pointer with an isometric joystick [SMK01]. c) The project *Blinkenlights* allows users to play the classic pong on building facades using the keypad of a mobile device [Met01]. d) The device's accelerometers can be used to control virtual cars on a larger display [VCBE08].

Accelerometers have further been used to detect the mobile device's motion. Yatani et al. present *Toss-It* which allows users to *throw* images onto a distant screen [YTH$^+$05]. In their system both the strength of a toss as well as its direction determine the target display. *MobiToss* enriches the throwing model with interaction [SOC08]. After tossing a movie clip onto an external display, the sensors can be used to control the clip by tilting the phone (e.g., applying different video effects). Such sensors further allow the recognition of gestures for interacting on distant screens rather than controlling a pointer [KKM$^+$06]. Shirazi et al. combine multiple mobile phones with an interactive tabletop to simulate a poker game [SDP$^+$09]. The accelerometers are used to indicate typical gestures known from poker games such as folding the cards.

### *The Mobile Device's Camera as Sensor for Interaction:*

A different approach of detecting the mobile device's motion is optical tracking. Most of today's systems use the method of *inside-out-tracking* in which the device tracks itself (mostly with respect to its local coordinate system). Ballagas et al. present *Sweep* which turns the mobile camera-equipped device into an optical mouse-like interface [BRS05]. Their system uses *optical* *flow analysis* to determine the mobile device's motion based on the previous video frame [HS80]. The image processing is done entirely on the mobile device which is rather slow due to computational constraints (see figure 2.11a). However, future mobile devices will increase in their capabilities. Jiang et al. use an absolute positioning to control a remote pointer [JOMS06]. The device's camera captures a frame (including the display with a pointer), detects the offset between pointer and the center of the camera and calculates the new pointer position (see figure 2.11b). Hansen et al. use computer vision to determine the distance to a display [HELO05]. Dependent on the distance, the mobile display can show a zoomed portion of an image shown on the remote display (see figure 2.11c). In contrast to these approaches, Miyaoku et al. use *outside-in-tracking* to detect the mobile device's motion [MHT04]. A camera mounted on the target display detects the mobile display and is able to distinguish between different mobile devices. Pointing the device towards the screen most likely results in a rather unusual hand posture. Furthermore, the mobile display cannot be used during the interaction for personalized input.

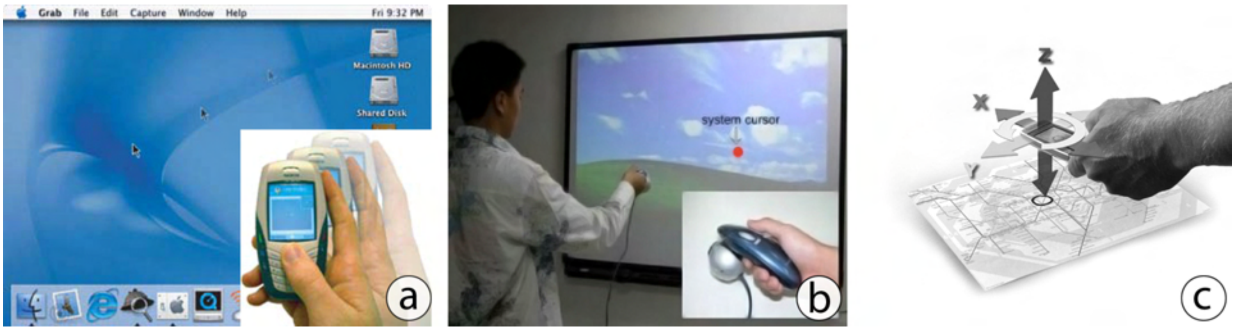


**Figure 2.11:** Relative and absolute pointing with the mobile device's camera: a) *Sweep* uses optical flow analysis turning the mobile device into a mouse-like interface [BRS05]. b) With *Direct Pointer*, users point at the position on the target display in an absolute fashion [JOMS06]. c) Optical tracking can be used to determine the mobile device's position and orientation with 6 degrees of freedom [HELO05].

## 2.4.3 Spatially Aware Handheld Devices

Recent advances in mobile device technology allow the recognition of objects to identify the spatial relationship of a mobile device with respect to the targeted objects [QBG08]. The *Chameleon* is a spatially aware handheld computer which allows users to browse information in three-dimensional situated information spaces [BF98]. In this system, both the physical and the virtual worlds are visible to the user. The *Boom Chameleon* is an extension to this system by Tsang et al. [TFK$^+$02]. The display is mounted on a boom around which users can circle to get perspective representations of three-dimensional models (see figure 2.12a). The implementation further decreases fatigue as users do not need to hold the device permanently in their hands. Nevertheless, the required boom restricts the user's movement and limits the *flexibility* of the environment at the same time. Furthermore, the device's mobility is rather low.

### *Spatially Navigating in Large Virtual Information Spaces:*

Fitzmaurice et al. bring augmented reality to handheld displays by adding a sensor allowing six degrees of freedom (DOF) [FZC93]. In their experiment, the authors show that users perceive depth on a small handheld two-dimensional display better by moving the device compared to a stationary 21 inch screen. The handheld display acts as a porthole into a larger three-dimensional virtual workspace. Within this space, users can select virtual objects and interact with them. In subsequent research, Fitzmaurice introduces possible scenarios such as a static map that actively emits different layers of information [Fit93]. If users are interested in further information regarding a certain region, they move the handheld device over it. The mobile device then displays the virtual information associated to this region (see figure 2.12b). Yee presents scenarios using a spatially aware device called *Peephole Display* [Yee03b]. This system simplifies the aforementioned ideas to a two-dimensional workspace also allowing for having two layers. While the virtual one remains fixed with respect to the world coordinate system, an overlaid layer moves with the device. In this scenario, users are able to copy and paste items [Yee03a].

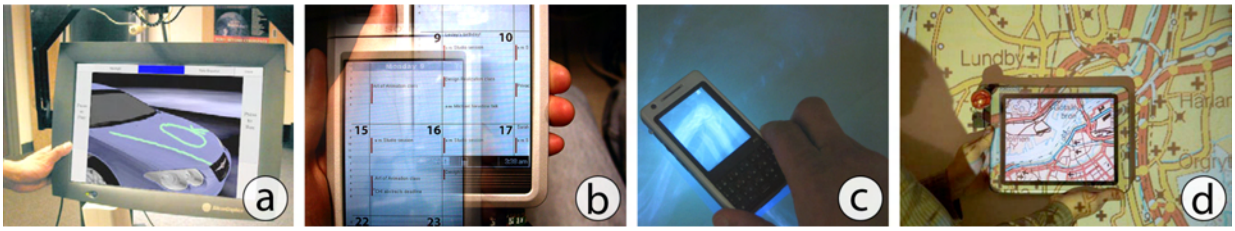


**Figure 2.12:** Interacting with spatially aware handheld devices: a) *Boom Chameleon* allows users to navigate around a virtual model [TFK+02]. b) *Peephole Displays* show a region of a larger virtual canvas [Yee03b]. c) Spatially aware handhelds on a tabletop allow for higher visual resolution [OF09]. d) A tablet PC shows a different visualization of a map [SH06].

Cao et al. present a model for pointing with such *Peephole Displays* taking into account whether the user has prior knowledge of the large virtual space or not [CLB08]. Mehra et al. further studied the difference of static (i.e., the locally shown content is fixed to the device) and dynamic (i.e., the locally shown content moves with the device accordingly) peepholes [MWW06]. Their results confirm other studies in favoring dynamic peepholes. If the underlying virtual canvas is large and the content rather unknown to users, target acquisition time increases [RSR+07]. The *Halo* technique address this problem by drawing circles around objects located off-screen [BR03]. The circles' radii depend on the distance of the object: the further an object is away, the larger the circle gets. However, these circles may overlap and therefore introduce visual clutter. *Wedge* overcomes this limitation by using triangular shapes instead of circles [GBGI08]. In target acquisition tasks, the authors found that the *Wedge* performs significantly better than *Halo*.

### *The Mobile device as Magic Lens:*

Prior knowledge about a virtual canvas can also be obtained by replacing it with a physical one. Mobile devices used in conjunction with large canvases (e.g., a large display) then act as *Magic Lenses* [BSP+93]. In addition, mobile devices feature high pixel densittes (i.e., hundreds of dots per inch) compared to large display technologies. Olwal et al. superimpose mobile devices on interactive tabletops [OF09]. Besides the sub-pixel accuracy in output, the mobile device allows much higher precision in input on the virtual image (see figure 2.12c). Sanneblad et al. extends such techniques with the third dimension [SH06]. Their system allows users to obtain a different view of a map shown on a wall-sized display (see figure 2.12d). Holding the device (here: a tablet PC) steadily and interact with it at the same time. Moreover, high fatigue levels are to be expected for arms and shoulders when interacting with this system.

## 2.5 Interacting through Live Video

Instead of interacting on virtual content on the personal device, live video streams can be used. Users can directly interact on objects shown in such streams according to the paradigm of *Direct Manipulation* [Shn83]. Two different setups can deliver these live images: first, cameras are stationary with respect to the environment delivering perspective different from the user's one. And second, the personal device uses its own camera to allow a view more aligned to the user's perspective. When used on a stationary display, both setups deliver stable images. Today's mobile devices usually feature a camera as well. The *mobility* of such devices is the advantage and disadvantage at the same time. While *mobility* allows for higher flexibility in terms potential targets, the produced camera images suffer from instability. Furthermore, allowing users (and the camera respectively) to move around requires permanently tracking the device. In this section we review both static and dynamic setups. We first describe techniques that make use of external cameras (see section 2.5.1). Subsequently, we focus on the usage of cameras built into mobile devices (see section 2.5.2). We pay close attention to possible tracking solutions for detecting the device's position and (if necessary) its orientation.

### 2.5.1 Fixed Camera Setups

Stationary cameras offer a fixed view into a predefined portion of the environment. Tani et al. use this setup to allow workers in an industrial factory to manipulate distant machinery through computers in a control room [TYT$^+$92]. In this *Hyperplant*, users can click and drag on the live video image of a machine shown on a regular desktop screen (see figure 2.13a). Actuators built into the machines respond to these actions: if users drag a slider in the video image, it moves in the real world (and in the video image respectively) to give immediate feedback. The three-dimensional representation of the environment is created by hand a priori and is hard to change and can further not be adapted to dynamic machinery (e.g., moving robots).

Fixed cameras can further be used to command (i.e., give directions to) movable machines such as robots. In *Sketch and Run*, users can draw strokes within live video to control cleaning robots [SHII09]. This system relies on cameras mounted on the ceiling which observe the entire room. Drawn lines in the orthographic view correspond to paths which are then followed by robots (see figure 2.13b). In contrast to Tani's approach, movable machines are tracked with markers. The only information known upfront is the position of each camera with respect to the room's coordinate system. The robots' speed was limited to the detection speed of the markers. Furthermore, the detection might fail when robots traverse from one viewport to another. Although the interaction takes place on a portable device, the live video images are stable. Nevertheless, the system allows users to move around and control the cleaning robots from any place within the room. As described before, installed cameras reduce the *flexibility* of the environment.

### *Interacting on External Displays through Live Video:*

Chiu et al. use video cameras installed in a conference room to facilitate meetings [CKRW99]. In their scenario, attendees can annotate live video image from a presentation source. Liao et al. extend this metaphor to connect virtual items to physical locations [LLK+03]. A video-representation of the room allows users to annotate slides. These presentation slides can also be dragged between displays. The video representation of the room further allows dragging digital information onto physical objects. For example, users can drag presentation slides onto the printer in the room to obtain a physical copy. The entire interaction happens within the video image. This system relies on fixed cameras similar to Tani's approach. The corresponding virtual representation of the room needed to be done upfront. Hence, this system only allows annotating stationary displays but does not offer a solution for portable computers.

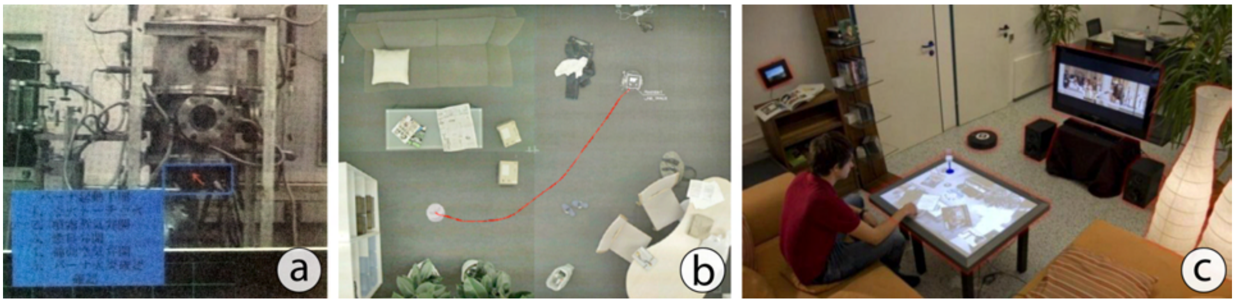


**Figure 2.13:** Interacting through video from static cameras: a) *Hyperplant* allows users to operate machines shown in live video from a control room [TYT+92]. b) In *Sketch and Run* users draw paths to maneuver cleaning robots [SHII09]. c) *CRISTAL* allows the control of home appliances in live video shown on a tabletop [SHS+09].

With the increasing number of digital tabletop systems, users can further interact in live video images using multi-touch input (see figure 2.13c). Seifried et al. present *CRISTAL*, a system showing live video images on a large tabletop [SHS+09]. In a living-room setting, users can control everyday appliances, such as light sources, an audio device, a TV, digital picture frames, or robotic vacuum cleaners (cf. [SHII09]). The authors compared a perspective camera position (i.e., in the room's corner) to an orthographic placement (i.e., in the middle of the room's ceiling). They found that a majority of participants preferred the latter setup. Again, the room geometry needs to be known to allow direct manipulation of objects.

### 2.5.2 See-Through Devices

Today, mobile devices are equipped with cameras and can be used for augmented reality applications in conjunction with large canvases (e.g., a paper-based map). The mobile device would allow a more detailed view of the map. Schöning et al. demonstrate how the mobile camera-equipped device can be used as such a lens [SKM06]. To track the device, two-dimensional visual markers augment the large map. These markers occlude valuable map space and add information unnecessary to the user. Rohs et al. reduce the visual clutter by using dot markers – a two-dimensional grid of black dots surrounded with white rings [RSKH07]. The system's performance is nearly unchanged whereas the map's readability increased significantly (see figure 2.14). Wagner et al. present a toolkit that allows different tracking solutions for mobile phones [WLS08]. The authors were able to reduce computational complexity to allow image processing on mobile devices at interactive framerates. Rohs et al. present a model of target acquisition when using mobile devices as magic lenses [RO08]. This model divides the interaction interaction into two consecutive phases: coarse acquisition followed by fine-control pointing.

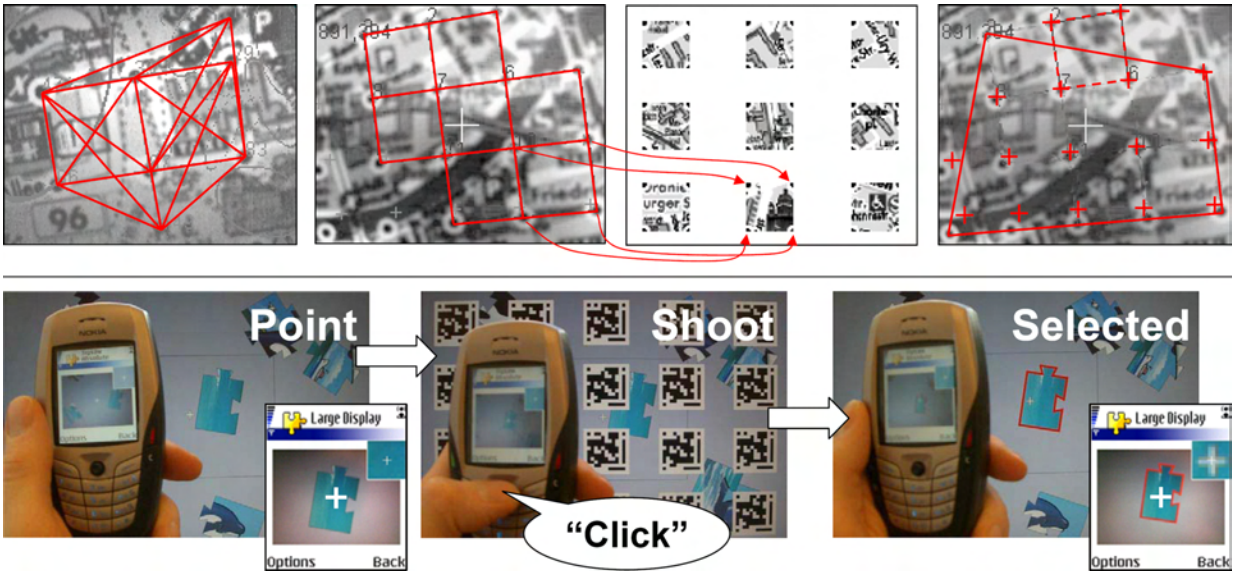


**Figure 2.14:** Pointing with the mobile device's camera: Top shows the *dot* markers used for map navigation [RSKH07]. Bottom denotes *Point & Shoot* procedure to select content on a distant screen [BRS05].

#### *Recognizing Fiducial Markers through Live VIdeo:*

Fiducial markers can also be combined with digital canvases such as large displays. Ballagas et al. present *Point & Shoot*, a system that allows the selection of virtual objects on a remote display through the mobile device's viewfinder [BRS05]. First, users aim at the item of interest using a crosshair shown in the live video image. When they press the joystick (to indicate a selection), a grid of visual markers is temporarily superimposed on the content to identify the targeted item. After finding the item, the markers are removed and the item is highlighted to indicate a successful selection (see figure 2.14). The markers can further be used to encode the Bluetooth address of the display they are shown on allowing for a *non-modal* connection procedure. Their system does not allow for *continuous* interaction as at least one marker then needs to be seen permanently by the camera. Furthermore, multiple simultaneous users would increase the time markers would be shown on the screen even if they only would select items in a *discrete* fashion. Alternatively, their system could show some markers permanently to avoid the aforementioned flashing of markers.

Madhavapeddy et al. use camera-equipped mobile devices in combination with markers to *continuously* control content on remote displays [MSSU04]. The markers (known as *SpotCodes*) used allow the detection of the device's position and orientation continuously (see figure 2.15a). Their envisioned system enables users to zoom into certain parts on a world map. As soon as they have selected a country, a list of airports is shown on the mobile device. These markers can further be used to rotate the mobile device which is then translated into manipulating a slider. The virtual slider can be shown on either the external display or the mobile device depending on the scenario. Since the markers are shown virtually, this system allows for a *flexible* environment. Their displayed size needs to be adjusted to allow for distant interaction.

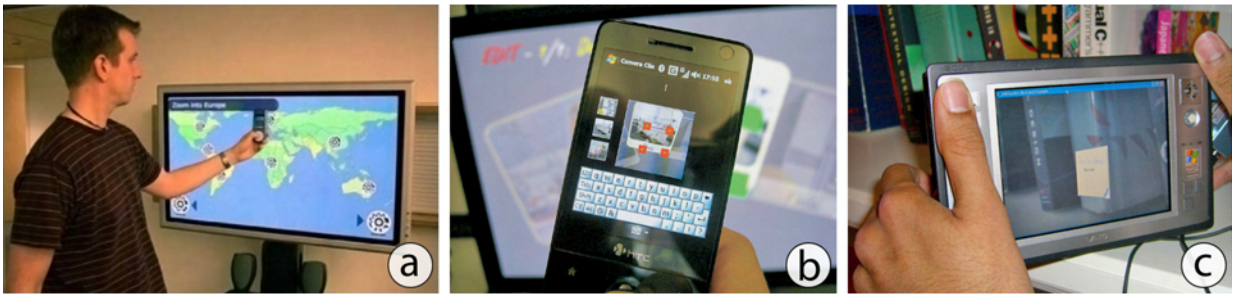


**Figure 2.15:** Virtual content shown in live video on mobile devices: a) The device's position and orientation is used to interact with content (through markers) [MSSU04]. b) Dynamic markers are used to obtain the transformation between mobile device and remote display in 3D [PJO09]. c) The *iCam* allows to place content at three-dimensional positions [PRA06].

Besides measuring position (two-dimensionally) and the device's orientation, the distance to an external display can be lead to new interaction techniques. Pears et al. present a system that identifies the spatial relationship between the mobile device and the external display it is pointed at [PJO09]. The display shows a simplified marker that allows the calculation of a homography (i.e., the transformation between the mobile device's local coordinate system and the target display's coordinate system). The crosshair shown on the mobile device is replicated on the external screen (see figure 2.15b). Users are able to select, move and rotate images similarly to previous systems. As the system is also able to detect the distance to the external display, users can scale images by moving away or getting closer to the screen. While this system is of great importance for our approach, the drawbacks of markers (or pointers) shown on the remote display still remains; limiting the potential for multiple simultaneous users.

*Interacting through Live Video in Three Dimensions:*

Physical objects and their locations can be augmented with markers as well. Rekimoto et al. present the *NaviCam*, a palmtop device that tracks itself with respect to the environment using a camera and color-coded markers [RN95]. Digital information is superimposed on the physical world seen in live video depending on the direction the device is pointed at. To reduce visual clutter caused by fiducial markers (both virtual and physical) is to let the environment locate the device. Patel et al. present an infrastructure to detect the position of a mobile device in three dimensions [PRA06]. Their system further determines the rotation around all axes allowing for six degrees of freedom. When users place local information into the environment, a laser is used to determine the distance to the object. The absolute position of the digital information can then be calculated using the relative information obtained with the laser and the absolute positioning information given with the location sensors (see figure 2.15c). While this system is accurate, it is not robust against dynamic changes in the environment (e.g., moving a book in a shelf).

## 2.6 Cross-Display Interaction

We have reviewed interactions taking place on single displays. However, users may want to interact across displays. A common task is to move digital objects from one screen to another. In such cases, the involved displays may differ in their properties according to our categorization (see section 1.1). Furthermore, a large number of screens may coexist in certain environments (e.g., laptops, tabletops and projected displays in meeting rooms). Interaction techniques need to be designed for the displays' properties to allow for moving objects across a randomly chosen pair of displays. Nacenta et al. present a taxonomy that classifies the *cross-display movement* based on three domains: *referential* (i.e., how to select a display), *relationship* (between input space and display configuration), and *control paradigm* (i.e., at which stages can the user actually control the movement) [NGAS09]. Techniques for content transfer across displays have two characteristics: first, users have to reach both originating and destination display. And second, the entire operation is performed on a personal display only. In this section, we first review cross-display interactions that require both displays to be reachable (see section 2.6.1). Subsequently, we focus on the content transfer operated only on a personal device (see section 2.6.2).

### 2.6.1 Reaching both Displays Physically

Several techniques have been developed that require the user to actively *carry* the digital information to the destination display. *Taking* the data from one display to another has been explored by Geißler et al. [Gei98]. Their system allows users to place their hand on the icon of interest causing it to disappear behind hand. To place the item on a different display, users place their hand on an empty region causing it to appear again. Streitz et al. used this technique in the i-LAND project to transfer objects between portable and large wall-sized displays [SGH+99].

This technique does not allow an association between user and data when multiple users are interaction simultaneously. This problem can be solved by detecting finger-prints or personalized devices. Ullmer et al. associate use wooden blocks (referred to as *mediaBlocks*) to data [UIG98]. Besides controlling content on one display with certain arrangements of the blocks, these blocks can also be used to carry information from one screen to another (see figure 2.16a).

### *Discrete Information Transfer:*

To place information more precisely, *Pick-and-Drop* uses digital pens with assigned IDs [Rek97]. These pens further allow distinguishing different users (assuming users have their own pens). To initiate the transfer, users click on an item using their unique pen. The data is then stored associated to the pen's ID. As soon as users click on another display, the corresponding data is placed at this location (see figure 2.16b). Furthermore, the data is invisible throughout the interaction. Users need to memorize what the carry on their pen. Kohtake et al. present *InfoPoint* [KRA01] as an extension. The system uses a pen with an embedded display to allow for the carried information's visibility (see figure 2.16c). While originally designed for cross-appliance data transfer (e.g., transferring a document to a printer), it can be used for cross-display content transfer.

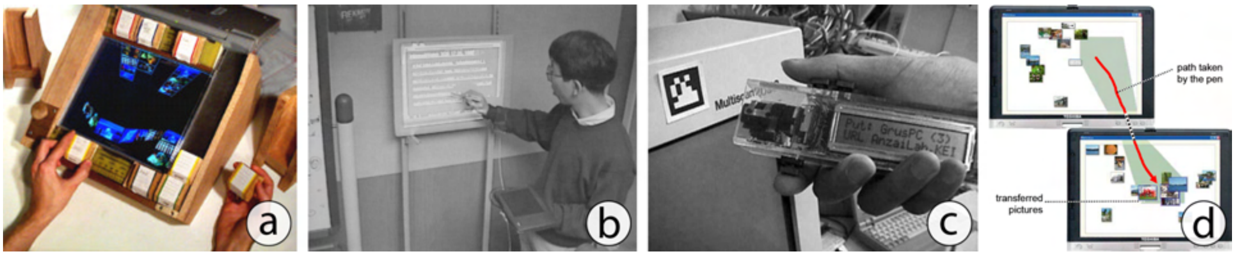


**Figure 2.16:** Transferring content across reachable devices: a) The wodden *mediaBlocks* carry information [UIG98]. b) *Pick-and-Drop* uses digital pens to transfer information [Rek97]. c) *InfoPoint* embeds ea display into the pen to show carried information [KRA01]. d) *Stitching* allows content transfer through a single pen stroke [HRG+04].

To further personalize (and potentially secure) cross-display object movements, Nacenta et al. present *Corresponding Gestures* [NASG05]. Instead of touching the item of interest with a pen, users draw a predefined gesture which selects the objects (*pick*). The users can go to the target display and place the object there by drawing the same gesture (*drop*). In a comparative study, the authors found that *Pick-and-Drop* is slightly faster than *Corresponding Gestures* for two reasons: first, drawing the gesture takes up more time than just touching the object with a pen. And second, the recognition of a gesture might fail requiring the user to draw it again. Compared to techniques that work at a distance (see section 2.6.2), both techniques were preferred least.

### *Continuous Information Transfer:*

Cross-display object movement can also happen in a continuous fashion. At the same time, displays are directly adjacent to each other or at least in the vicinity of the user. In *ConnecTables*, users can move items from one screen to another as if they were a single display [TPMT+01].

Similarly, *Synchronous Gestures* require an explicit connection between the two devices up front [Hin03]. The subsequent interaction remains the same as both displays form one logical workspace. *Stitching* on the other hand allows both the connection and immediate content transfer at the same time [HRG+04]. This technique requires users to quickly move from one screen to another within a small time interval (see figure 2.16d). After dragging an item to another display and releasing the pen, a *remote postfix menu* appears allowing for an operand selection. One concern raised during their study was that users wanted to know how to *unlink* both devices. This seems to be of great importance when security and privacy have to be considered.

Nacenta et al. describe techniques requiring the user to interact on both the originating and the destination display as *two-sided* [NASG05]. Accordingly, *one-sided* techniques require the user to only operate one device (either originating or destination display depending on the direction of the operation). The authors define crucial factors that influence on *two-sided* techniques. Most importantly, both displays need to be reachable and further recognize the used input device (e.g., a pen). Another criterion is the symmetry of the operation which usually is *symmetric* for *two-sided* techniques. This means that the object can be transferred back and forth using the same kind of interaction. Naturally, the overall task completion time for *two-sided* techniques depends on the distance between both displays which may require users to move to the destination display when it is out of their arm's reach.

## 2.6.2 Content Transfer at a Distance

Several display arrangements do not allow the aforementioned transfer techniques: They may not be reachable at all or they have different input capabilities (e.g., mouse versus direct touch). Most common are techniques that allow content transfer from and to a large public display using a personal, mobile device. One of the most distinguishing factor of such transfer mechanisms is whether the content's final position can be controlled. If users can influence on the position, it then is important how accurate this control is.

### *Screen-level versus Pixel-level Accuracy:*

In the *CityWall* project, several pictures are shown on an interactive wall-sized display sorted by time [PKS+08]. The system allows users to upload their *flickr*[3] pictures onto the display by using a customized application [PSJ+07]. However, receiving pictures (i.e., copying an item to the mobile device) is not possible. The *Hermes Photo Display* allows users to upload as well as receive pictures from a nearby public display [CDF+05]. Sending a picture is done by selecting it on the mobile device and then copying it to the remote display using Bluetooth. If users want to receive a picture, they touch it on the display and initiate a transfer on the mobile device. This does not allow multiple users to select content simultaneously. The *Snap and Grab* system uses the built-in camera to take a picture of content on the remote display which is then transferred to the mobile device [MMH07]. This technique allows simultaneous selections (see figure 2.17a).

[3] http://www.flickr.com

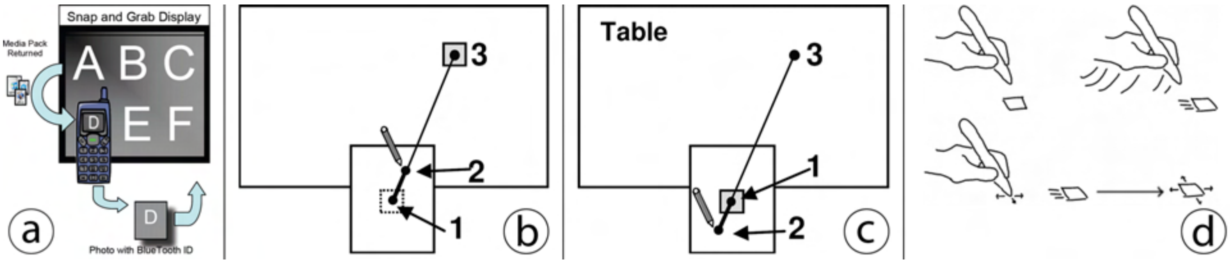


**Figure 2.17:** Transferring content from and to distant displays: a) With *Snap and Grab*, users can take a picture to obtain content from a nearby display [MMH07]. b) and c) *Pantograph* multiplies the pen's motion to reach distant regions; *Slingshot* uses the inverse motion of *Pantograph* [Has03]. d) *Superflick* allows users to control the object's motion after releasing it with subsequent pen gestures [RGS$^+$06].

Similar to throwing models, the aforementioned systems allow *accuracies* on screen-level rather than on pixel-level. To enable more precise content placement users need to control the placement up front. *Pantograph* allows users to drag an object towards the desired target display [Has03]. At the same time, the object itself moves further with a linear factor in the same direction (see figure 2.17b). While pixel-accuracy is still not possible due to multiplying the movement, the object's final position is much more precise compared to screen-level accurate systems. *Slingshot* uses the exact opposite concept by simulating a behavior similar to a catapult [Has03]. In this system, users drag the finger or pen away from the target display as if they would span a sling (see figure 2.17c). Multiplying the movement similar to *Pantograph* leads to the same accuracy. In a comparative study, Nacenta et al. found that *Pantograph* was easier to understand than the inverted motion used in *Slingshot* [NASG05].

### *Open-loop Information Transfer:*

Besides *closed-loop* techniques (i.e., the user is in control during the operation), several techniques use *open-loop* approaches. In such techniques, users only initiate the interaction but have no further control while the object movement is in progress. Moyle et al. present *flick* gestures for pen-based interfaces [MC02]. In their system users touch the object they wish to move with a pen and flick it into the desired direction. Users can only influence the direction of the operation directly. The distance is controlled by the pen's speed. Both parameters cannot be changed once the pen has been released from the surface. To allow for substantially large distances, *Press-and-Flick* considers the pen's pressure as well. Both the pen's pressure and the flick itself determine the object's speed and its final location [NASG05]. *Superflick* presents a modification to allow limited user interactivity during the object's movement [RGS$^+$06]. Users flick the item as described previously. The final position can be corrected immediately after flicking the object by moving the pen into the desired direction (see figure 2.17d). For example, moving the pen to the left causes the object to move left during its movement while moving the pen backwards slows down the object. Users have more control and can achieve higher accuracies.

### *Representing the Destination as World-in-Miniature:*

Moving content directly to its destination on another screen is realized by using a miniature representation of the target display. The most common technique is the concept of *world-in-miniature* [SCP95]. Dragging items onto the representation of a remote display allows both transfer and positioning them. Furthermore, reorganization of items on a remote display can be facilitated [BB04]. The accuracy then depends on the magnification (or in this case downsizing) of the remote content. If the content is shown with half the size (in pixels), the accuracy therefore is halved as well. If keeping the accuracy is crucial to the task, the system may only show the important parts of remote content similar to *WinCuts* [TMC04]. The accuracy is constant (i.e., it is the same as on the local display) within this region. Repositioning content across large distances is only possible by repositioning the local porthole as well. Another option is to use two of such portholes to transfer content from one position to another one further away [KFA$^+$04]. The limited accuracy of *world-in-miniature* interfaces can also be increased by using the *Bubble Cursor* technique [GB05]. *Bubble Radar* is one of these systems that allow precise selection on a potentially downsized content representation [ANSG06].

## 2.7 Summary and Discussion

This chapter presented an overview of various techniques interaction with external displays and devices. We first investigated the different phases occurring when interacting with an external screen. Subsequently, we focused on the connection between a mobile, personal device to remote displays. We further reviewed how users can interact with distant displays in general followed by a detailed overview of existing techniques that facilitate pointing at external screens using a mobile device. As handheld augmented reality currently receives huge attention, we described existing research projects and showed how this paradigm may be used to interact with distant displays. Besides interacting with a single display, we also showed available systems for transferring content from one display to another. In figure 2.18, a subset of the presented approaches is included into the taxonomy introduced in section 1.2. This spatial arrangement of systems serves as a basis for identifying problems related to interacting with external displays.

By reviewing systems for environments with *limited flexibility* only, it becomes obvious that the *distance* is a limiting factor for most prototypes. Especially when *relative* pointing is used, only one system allows for a *non-modal* connection procedure [MSSU04]. This system further only enables users to interact at a *limited distance*. Most *absolute* pointing, on the other hand, allow for connecting in a *non-modal* fashion. It is noticeable that only two systems appear to accomplish this at *variable distances* [SHII09, SHS$^+$09]. Furthermore, nearly all systems (with either *relative* or *absolute* pointing) allowing *non-modal* connection mechanisms use *direct* pointing methods. Most surprisingly, no system was found that incorporates *modal* connection mechanisms at *limited distances*. It can be further noted that connections at *limited distances* occur in a *non-modal* fashion. It further appears that the *distance* is a limiting factor for connecting in a *non-modal* way already in environments with *limited flexibility*. The environment's limitation in these prototypes can be explained as follows: first, the spatial layout needs to be known a priori for cross-display operations [ON01]. Second, the interaction technique needs a special input device such as a digital pen [Rek97]. And third, additional equipment needs to be present in the environment to track the mobile device's position and orientation [MHT04].

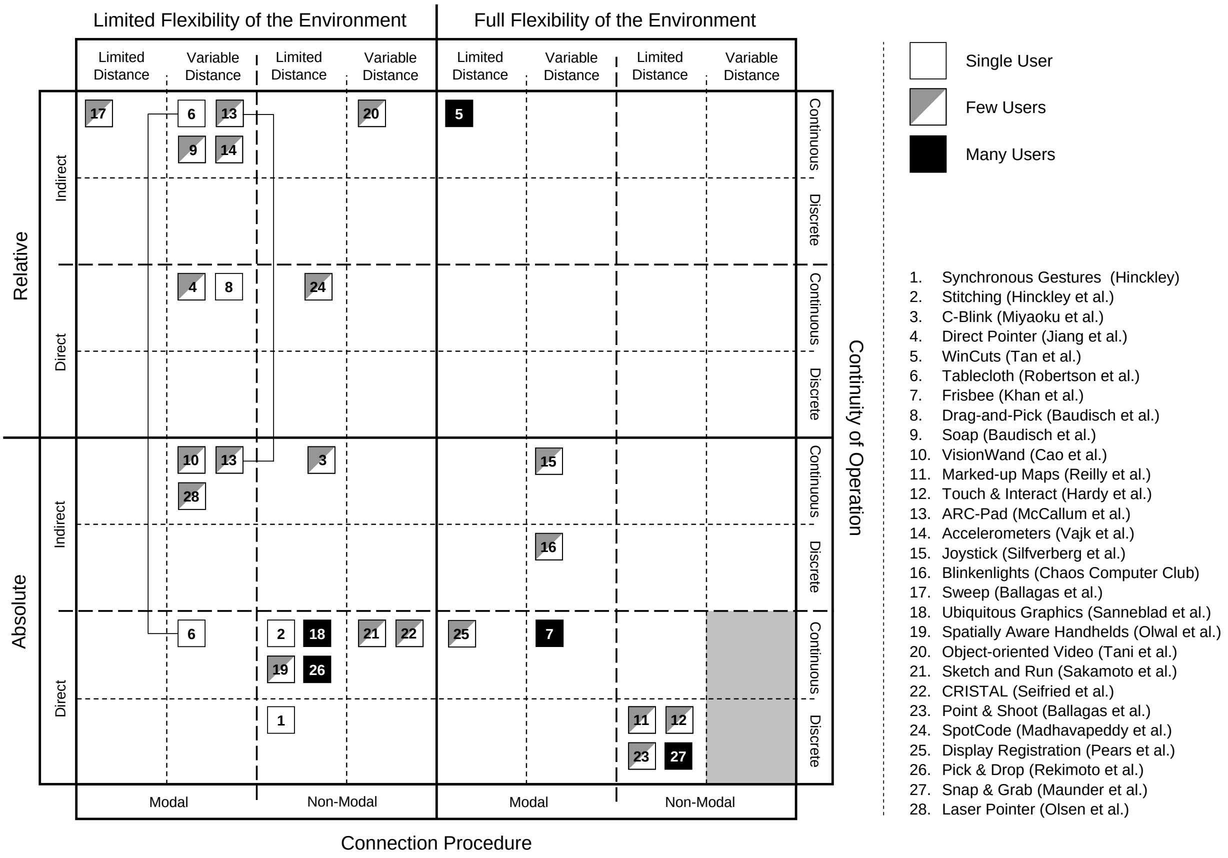

Limited Flexibility of the Environment
Full Flexibility of the Environment
Limited Distance
Variable Distance
Limited Distance
Variable Distance
Limited Distance
Variable Distance
Limited Distance
Variable Distance
Relative
Absolute
Indirect
Direct
Indirect
Direct
Continuous
Discrete
Continuous
Discrete
Continuous
Discrete
Continuous
Discrete
Continuity of Operation
Modal
Non-Modal
Modal
Non-Modal
Connection Procedure
17
6
13
9
14
20
5
4
8
24
10
13
3
28
15
16
6
2
18
21
22
25
7
19
26
1
11
12
23
27
Single User
Few Users
Many Users
1. Synchronous Gestures (Hinckley)
2. Stitching (Hinckley et al.)
3. C-Blink (Miyaoku et al.)
4. Direct Pointer (Jiang et al.)
5. WinCuts (Tan et al.)
6. Tablecloth (Robertson et al.)
7. Frisbee (Khan et al.)
8. Drag-and-Pick (Baudisch et al.)
9. Soap (Baudisch et al.)
10. VisionWand (Cao et al.)
11. Marked-up Maps (Reilly et al.)
12. Touch & Interact (Hardy et al.)
13. ARC-Pad (McCallum et al.)
14. Accelerometers (Vajk et al.)
15. Joystick (Silfverberg et al.)
16. Blinkenlights (Chaos Computer Club)
17. Sweep (Ballagas et al.)
18. Ubiquitous Graphics (Sanneblad et al.)
19. Spatially Aware Handhelds (Olwal et al.)
20. Object-oriented Video (Tani et al.)
21. Sketch and Run (Sakamoto et al.)
22. CRISTAL (Seifried et al.)
23. Point & Shoot (Ballagas et al.)
24. SpotCode (Madhavapeddy et al.)
25. Display Registration (Pears et al.)
26. Pick & Drop (Rekimoto et al.)
27. Snap & Grab (Maunder et al.)
28. Laser Pointer (Olsen et al.)

**Figure 2.18:** Classification of implemented interaction techniques in surveyed projects. The inspection of the diagram reveals that no system exists that considers a *non-modal* connection in a *flexible* environment at *variable* distances.

When inspecting systems that allow for interacting in *flexible* environments, *modal* connection mechanisms have been researched widely. Again, it appears that the *distance* is the most limiting factor. While connecting with a separate *mode* is realized at *variable distances*, establishing a connection in a *non-modal* fashion is again only realized when the external display is *close*. Besides using NFC, two systems employ the mobile device's camera for interacting through live video [BRS05, MMH07]. It appears that the mobile device's camera is the limiting factor in these prototypes: Such cameras usually come with wide-angle lenses allowing for a large field of view at short distances. This, in turn, limits the interaction itself as a nearby display is already shown relatively small in the viewfinder of the device. It is further mentionable that all these systems (both NFC-based as well as camera-based) only allow for *discrete* interactions. When looking at the number of simultaneous users, only few systems have been proposed that conceptually allow a large group of users. However, this is beyond the scope of this thesis and is only mentioned for completeness. Another interesting observation is that *absolute*, *direct*, and *continuous* operations are not widely researched [KFA$^{+}$04]. Most interestingly, all systems allowing for interaction at *variable distances* only enable connecting in a *modal* way. Furthermore, when inspecting the dimension of *non-modal* connection mechanisms, it shows that no system has been proposed that allows for interacting at *variable distances*.

As denoted by the shaded area in figure 2.18, the taxonomy suggests that *absolute* and *direct* interaction techniques that allow for *non-modal* connection mechanisms at *variable distances* in *flexible* environments is an interesting field of research. During our analysis, it became apparent that the *distance* seems to be the most influencing factor in environments with *limited flexibility* as well as *flexible* environments. For this reason, we will present two prototypes (see chapters 3 and 4) to investigate the *distance* in more detail. With the findings of these prototypes, we then propose our new model that considers the aforementioned combination of attributes.

# II

# Direct versus Distant Interaction

Chapter 3

# *LucidDisplay*: Placing Mobile Devices on External Displays

*When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.*

**– Arthur C. Clarke –**

The discussion and analysis in chapter 2 revealed that the distance (and the spatial relationship respectively) between the mobile and the external display is one of the main factors influencing the modality of the connection process. By placing the mobile device directly on top of the external display, users can interact with remote content through their own device [BIF04, OF09]. Furthermore, a direct link can be established between the devices is established from the user's point of view. In this chapter we revisit the use of a touch-enabled mobile display being superimposed directly on the external display. At the same time, we focus on the special condition of (1) having different pixel densities (i.e., number of dots per inch) on the mobile display and (2) both displays being accessible and interactive at the same time. These special attributes allow the mobile device to play two different roles. The local sub-pixel accuracy allows for more visual detail and (if used with a stylus) higher input resolution when it acts as *see-through* device. In our observational study we found that users generally prefer the mobile device in combination with the external display. In contrast to a *see-through* device, both the mobile device and the external screen show their individual content. In the latter case, content can be transferred by performing a single tap on the item of interest. However, the traditional *Drag'n'Drop* works similarly across the display's bezels. In an observational study we found that users generally prefer the mobile device in combination with the external display. The experiment further revealed that the single tap as transfer mechanism is preferred over the traditional *Drag'n'Drop*.

The remainder of this chapter is structured as follows: We first discuss the resulting factors when the mobile device is superimposed directly on the external screen (see section 3.1). We then describe different scenarios for such settings: (1) the mobile device acting as *see-through* display and (2) both devices showing individual content (see section 3.2). Subsequently, we describe our prototypical setup and our applications (see section 3.3) and present results from an observational study (see section 3.4). We summarize this chapter and discuss our prototypes (see section 3.5).

## 3.1 Factors of Directly Superimposed Displays

When a display is physically superimposed on another one, certain characteristics are present. On the one hand, both displays are accessible and can be reached by users. Assuming the external display to be interactive, users now have two interaction canvases. This opens op new possibilities: first, users can either move the mobile device or the content shown on the large display. And second, users can perform cross-display content transfer across the mobile device's bezels. On the other hand, mobile devices usually have higher pixel densities (i.e., hundreds of dots per inch) compared to large screens (i.e., less than one hundred dots per inch). In this section we introduce these two differences in more detail.

### 3.1.1 Having Both Displays Accessible

As mentioned before, placing the mobile device directly on top of the tabletop, the external display is accessible to users as well. Input events can now occur on both displays either simultaneously or sequentially. Furthermore, allowing users to perform a single stroke across from one display to another (across the bezel of the mobile device) allows content transfer with the original *Drag'n'Drop* metaphor [HRG$^+$04]. However, users have to cross the bezel which causes the finger to move vertically down and up respectively depending on the drag direction (see figure 3.1). The vertical distance that is required depends on the mobile device's height. While the mobile device in our prototype still has a noticeable height, we assume that future devices will become thinner allowing for horizontal finger movements without any vertical one. Nevertheless, our prototype demonstrates how dragging content between displays (based on the location of the mobile device) can be realized.

As users are now able to directly interact with the target display (i.e., without using the mobile device), coarse interactions will most likely happen on the tabletop display only, assuming it is capable of direct touch input. The user's touch input on the tabletop may influence on the region shown underneath the personal display. If the mobile device shows this region (i.e., it is not completely displaying independent content), its view needs to be updated as well. For example, a user may rotate a map shown on the external display while the mobile device acts as a see-through display (and a magic lens respectively). The personal device then needs to update its view to still allow for a correctly aligned map (and the satellite image respectively). On the other hand, manipulating objects on the mobile display may also influence the information on the external screen. For example, users may place waypoints on the map and subsequently move the personal display somewhere else on the external screen which turns the waypoint to be visible on the tabletop. Any kind of input events need to be communicated immediately after it occurs. In both directions, this can be done using the matrix describing the transformation between the image plane of the mobile display as well as the external screen.

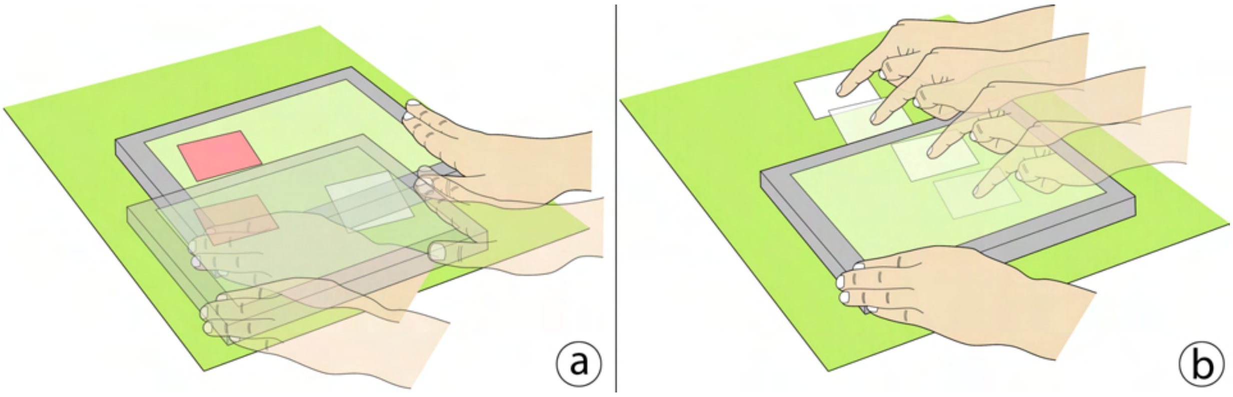


**Figure 3.1:** Accessing the external display: (a) Users can move the mobile device on the external display's surface. The red item resides on the local display, the white is shown on the external screen but can be seen through the mobile display when it is moved on top. (b) Content may be transferred using *Drag'n'Drop*.

Applications on the external display may also allow moving the entire canvas. For example, users can drag a map to another location to see a different city. This again influences the graphical representation shown on the mobile device. It also means that users now have two options of moving content so that it is shown on the mobile device: they either virtually move the entire canvas on the external display or simply reposition the mobile device. Moving the entire canvas although the mobile device is placed on top of it may feel awkward as it bypasses the laws of physics (assuming elasticity of the large canvas). If the interface would behave physically correct, the canvas would stretch on one side and jolt on the other side respectively. Furthermore, people tend to use their spatial memory to locate objects [RCL$^+$98]. If they move the entire canvas, the spatial relationships between icons still exist but their absolute location changes. This may then decrease the users' ability to locate content items quickly. On the other hand, this method may be preferred if the region of interest is located at a position on the external display that is hard to reach. This happens, for example, when this region is at the far end on a tabletop.

### 3.1.2 Local Sub-Pixel Accuracy

When the content of the larger display is shown entirely on the mobile device, it is usually downscaled (depending on the difference in pixel resolution). This leads to a loss in accuracy if users interact on a smaller representation of the large screen's content. The control-display (CD) ratio is significantly larger than 1:1; meaning that one pixel on the mobile device covers at least three pixels of the target display [Jac96]. This then affects the precision of both input and output. In addition, the *fat finger problem* further lowers the accuracy when users interact with smaller content. The precision can be slightly increased by (1) using a stylus instead of a bare finger or (2) by *shifting* techniques as presented by Vogel et al. [VB07]. To allow for a CD ratio of 1:1 the mobile display could only show a portion of the large display. The question then is how to (1) choose this region of interest and (2) change the viewport if long-distance drag operations are needed. These two issues can be addressed by allowing users to place the mobile device directly on the larger display. The mobile display then shows the region it currently covers.

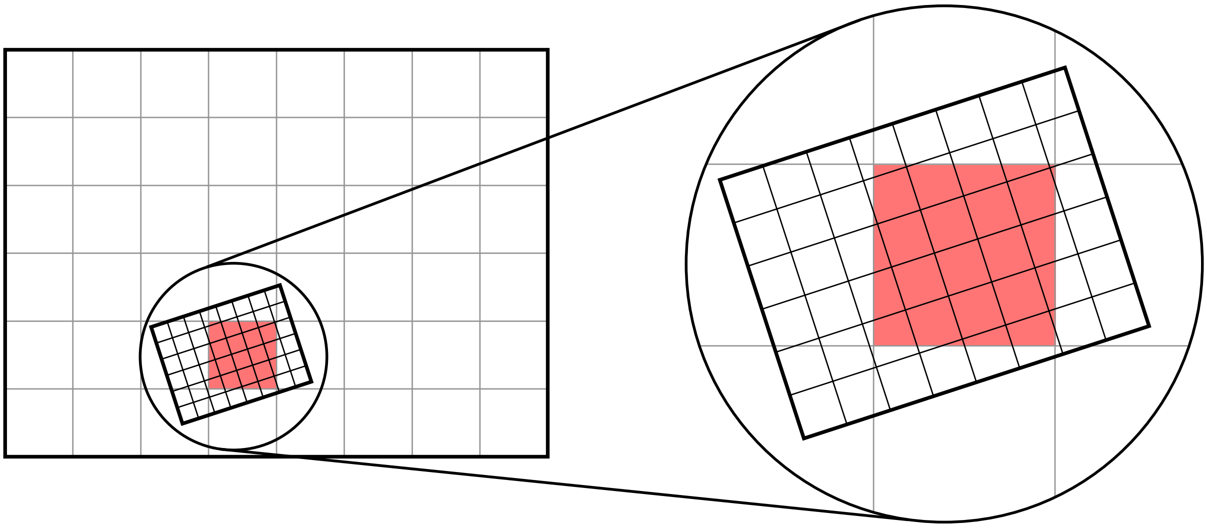


**Figure 3.2:** Sub-pixel accuracy of directly superimposed displays: a mobile device with the same pixel resolution (but smaller physical size) is superimposed on a large display. Multiple pixels of the mobile device thus cover a single one on the larger screen.

Today's mobile devices usually feature a denser pixel resolution compared to large displays. For example, a regular 42" TV may have a resolution of 1920 × 1080 pixels leading to a density of 52.5 dots per inch (dpi). On the other hand, mobile devices may only have a diagonal of 3.5" while having a resolution of 320 × 480 pixels resulting in approximately 165 dpi. When the mobile device is placed directly on top of the larger display covering a certain region, the CD ratio is lower than 1:1; meaning that several pixels on the mobile display cover one pixel of the external screen. This high-precision area allows for (1) higher visual detail and (2) more accurate input for precise operations based on the high visual output (assuming that a stylus is used for interaction). The higher local pixel density requires the content to be rendered with more resolution as well. Otherwise, the content would simply be scaled which rather reduces the quality than increasing it. One solution is to use vector-based graphics which can be scaled without any loss of quality. Another approach is to have all information at very high resolutions which are then downscaled for the external display. Furthermore, the enabled sub-pixel accuracy only is useful if the target display is able to process it. Otherwise, the values of all input events need to be rounded which rather decreases the accuracy. Device-independent coordinates solve this by representing each location of an event in real-world measurements (e.g., fractions of inches). To facilitate the input processing, the content on each of the displays is then also given in such coordinates.

The concept of having a region with increased input and output resolution is referred to as *focus plus context screens* [BGS01]. However, these systems usually are static leaving no option of moving the focus (i.e., high resolution) area. If users want to have certain content in the focus region, they have to move the content item into it. As described before, users can either drag the entire content canvas to bring certain information into the context area when a mobile device is placed on the external display. The mobile device further allows moving the focus region instead of the item of interest [BIF04]. Nevertheless, the focus region is always defined by the mobile device's screen real-estate. If a continuous operation within a larger area is required, fixed focus plus context screens need to be zoomed out. Having a movable device acting as the focus display, users can perform such an operation bimanually [Gui87]. The dominant hand performs the input itself, while the non-dominant hand moves the focus device. In summary, this method allows for operating in regions larger than the focus region only.

## 3.2 Roles of the Mobile Display

When a mobile device is superimposed on a larger display it can have different roles. On the one hand, it may show its own content independently of the external display. On the other hand, the mobile device can act as a *see-through* device allowing for a physical magic lens or toolglass [BSP$^{+}$93]. In this section, we briefly explain and discuss the different roles of the mobile device, starting with the *see-through* approach which allows for higher precision in input and output (see section 3.2.1). Subsequently, we illustrate the use of the mobile device as an individual content display with opaque background (see section 3.2.2). We then present a combination of both approaches by using alpha-transparencies on the mobile device (see section 3.2.3).

### 3.2.1 See-Through Device

If the mobile device does not show any content on its own, it can be used as a *see-through* interface similar to a *toolglass* [BSP$^{+}$93]. When users move the personal display on the external screen, additional information can be shown according to its position. By further only allowing input through the mobile device, the options for items shown on the external screen can be individualized based, for example, on access rights of the corresponding user. Input events received on the mobile display are transformed into the external display's coordinate system using the spatial relationship between the personal device and the target screen. The external display is then responsible for consuming the input event and updating its view as well as the mobile display's content. Due to noticeable higher pixel density of mobile devices, certain interactions such as reading text are much more feasible. If users want to achieve the same output accuracy on larger screens with less pixel density, they would have to zoom in to allow for the same size measured in pixels. To allow for aligned views, however, the shown content on the mobile device needs to be updated permanently according to the mobile device's current position.

Instead of showing the same content as the external display, the mobile device can present a different view of the same data. For example, the external display could show a map while the mobile device renders a satellite image of the region it covers (see figure 3.3b). This type of behavior is often referred to as *magic lens* [BSP+93]. The higher pixel density of the mobile device further increases the level and detail and the readability of remote content at the same time. The display size of the mobile device still limits the level of detail. For example, if the mobile screen's pixel density is twice as high as the external display's one, the level of detail is also doubled only. To overcome this limitation, a virtual screen (i.e., a rectangular shape) may be placed on the external screen. The covered area by this rectangle defines the visual output on the mobile display. The location of the mobile device then is not important. If now more detail is required, users can downsize this region which results in zooming into the displayed information on the mobile device. In this setup, repositioning the mobile device as well as manipulating information on the external display again may influence the content on the personal display. Hence, the mobile device's content has to be updated permanently.

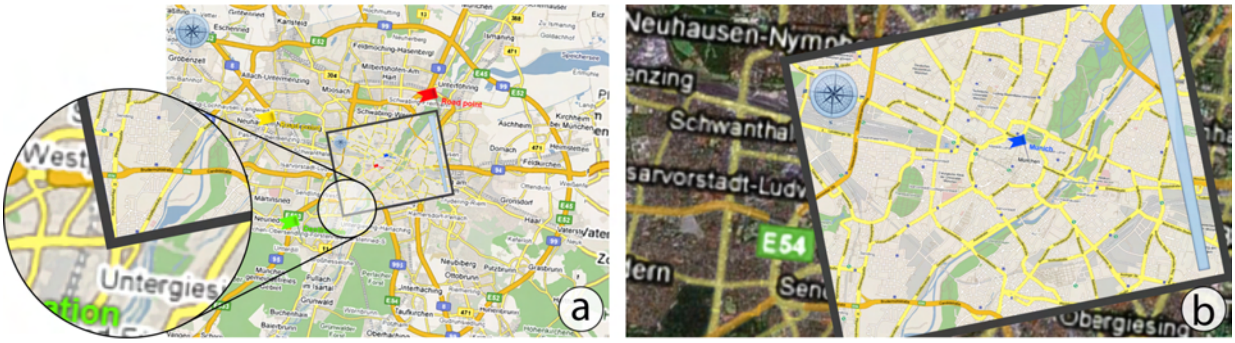


**Figure 3.3:** Using the mobile device as *see-through* device or *magic lens* (conceptual sketches): (a) The *see-through* mode allows for higher accuracy in input and output. (b) The mobile device, acting as a *magic lens*, shows a different visualization of the same data.

### 3.2.2 Fully Opaque Content

Naturally, both the mobile device and the tabletop show their own content. We refer to this form as the *desktop* view. Such a view contains a background as well as icons which represent different information units (e.g., text documents, media files, etc.). Both content views are further separated from each other, meaning that changes on one display do not affect on the content of the other one. The position of the mobile device on the large display further does not have any effects on the content shown on both screens. When users place their mobile device on the external screen, opaque backgrounds cause the region underneath the personal display to be temporarily invisible. This means that users cannot interact with this region directly. This kind of placement can further be used to *hide* content shown on the external display (see figure 3.4). However, this limits the full overview of information on the large canvas and may influence on the tasks other users want to perform. Nevertheless, interacting with the mobile display's content is possible at all times, also when the device is currently not located on the external screen.

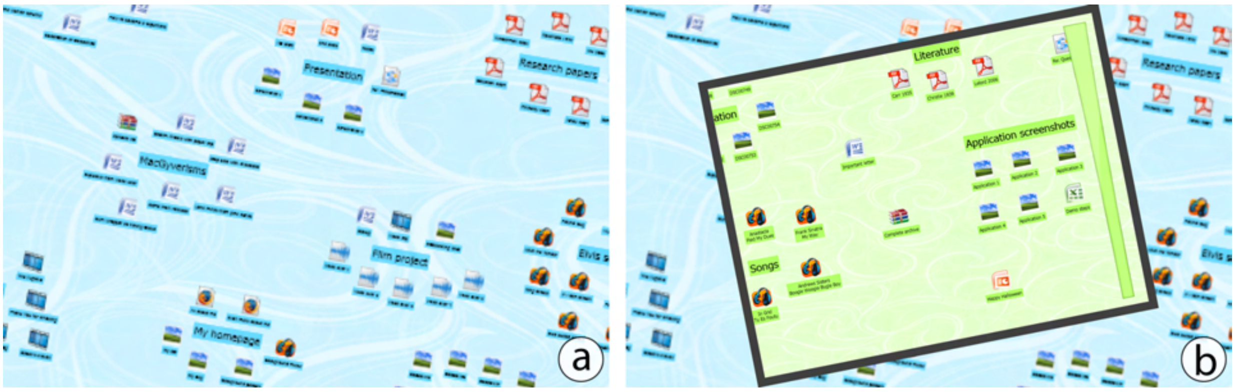


**Figure 3.4:** The mobile device in *full opaque* mode (conceptual sketches): (a) The desktop on the external display without the mobile device. (b) The mobile device superimposed on the external screen showing its own content. Information displayed underneath is hidden.

When the mobile device shows its own content with an opaque background, interacting with the external screen through the mobile display per se is impossible since information displayed underneath is invisible. Having two devices with their independent content (i.e., personal information on the mobile device versus public content on the external screen) may raise the need for transferring content between both displays. The previously presented interaction using a single tap would only work when content is moved from the mobile device to the large display. Visual feedback can be given in this case by, for example, fading out the transferred content. Addressing information located underneath the mobile device in turn is impossible. In order to allow the transfer in both directions using the same interaction technique (i.e., *two-sided technique*), a well-known solution exists in *Drag'n'Drop*. Users can drag content across the mobile display's bezels in both directions. This is possible since both screens are accessible during the interaction.

### 3.2.3 Translucent Display

As an alternative to dragging content across the mobile display's bezels, we decided to use a semi-transparent view on the mobile device. We refer to this as the *translucent mode*. Whenever a mobile device is not placed on the external display, the personal content naturally is shown fully opaque. However, as soon as the device is placed on the target display, the local layer becomes semi-transparent. By doing so, users are able to observe the content located underneath the mobile display (see figure 3.5). The mobile device comprises two layers – one for local content and one for content residing on the external display. In this setting, transferring an item can be understood as moving it from one layer to another. This may be done by tapping on an item (regardless of its layer). The tapped item then moves to the other layer (i.e., from the external screen to the mobile device or vice versa). In this setting, the background layer also needs to be updated permanently. Therefore, the external display has to notify the mobile device if either (1) the transformation between both displays changed, for example, when the mobile device has been moved or (2) when content shown underneath the mobile display has been changed.

Selecting multiple items using a rubber-band is impossible with this solution. One solution to address this issue is to have one layer as the *active layer* while the other one is the *passive layer*. When the personal content layer is active, it is shown with less transparency. This in turn means that the background is only slightly visible to the user. When the remote layer is the active one, the foreground layer showing personal content is displayed with high transparency. Users are then mainly able to see the background layer. Turning a passive layer into an active one is achieved by lingering with a finger on an empty region on the mobile device's desktop. A *free* region is defined through a location at which neither the foreground nor the background layer has an icon. Similarly, transferring from one layer to another is done by tapping the respective icon. This transfer method still works in both ways (i.e., passive to active and vice versa). Finally, the selection of multiple items can be performed on the active layer only. By drawing a free-form shape around icons shown on the respective layer, the user is able to select them.

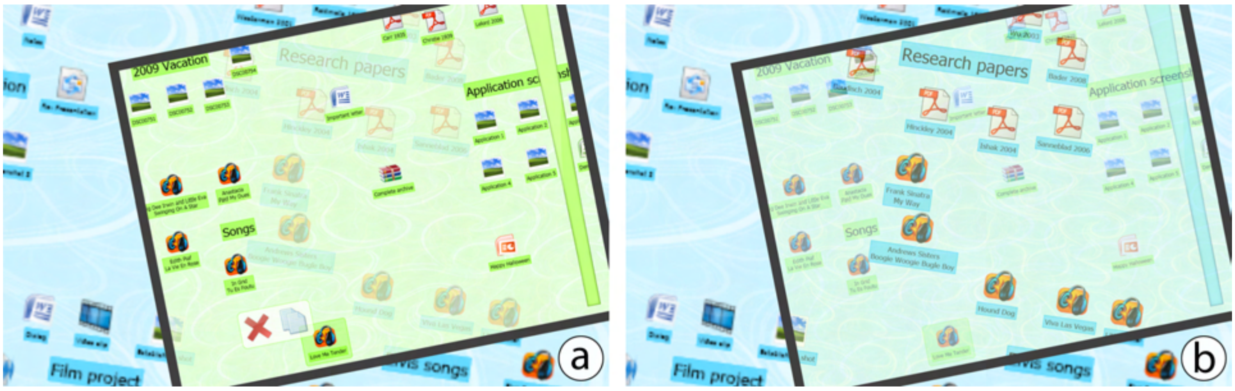


**Figure 3.5:** Local versus remote content layer as *active layer* (conceptual sketches): (a) The foreground layer is the *active layer* with an opacity of 25%. (b) The background layer as *active layer*. The local layer now has an opacity of 75% almost appearing fully transparent. Separating foreground and background perceptually occurs when moving the tablet PC.

## 3.3 A Proof-of-Concept Prototype

To test the usability of the different roles directly superimposed displays, we implemented a prototype called *LucidDisplay*. The system uses a large tabletop acting as external screen and a tablet PC as mobile device. Our system is similar to the so-called *Fovea-Tabletts*® [GER$^{+}$07]. Although we present the different versions of our prototype by only having one mobile device, we show how different tablet PCs can coexist on the tabletop. In this section, we first present an implementation that allows for tracking multiple mobile devices on the tabletop (see section 3.3.1). We then describe our applications and briefly introduce the overall implementation that was used in an observational study (see section 3.3.2).

## 3.3.1 Tracking the Mobile Device

There are three possibilities to track the mobile device: first, the environment tracks the exact location of all displays. Second, the mobile device knows its own position and communicates this information to all external displays in the environment. And third, the respective target display identifies a mobile device once it is placed upon its surface. In our prototypes, we decided to use the third approach to demonstrate the effects of directly superimposing a mobile display on a large screen. Having a mobile device tracked and identified by the respective external display implies the target display from the user's point of view. In other words, the system only assumes a tight connection between two screens when a mobile display is directly superimposed on another screen. Such tight couplings exist if the personal device is placed directly on top of the secondary display. In contrast to *Ubiquitous Graphics*, our prototypes employ an interactive tabletop [SH06]. This large display allows users to place the mobile device on top of if while avoiding the need of permanently holding the device.

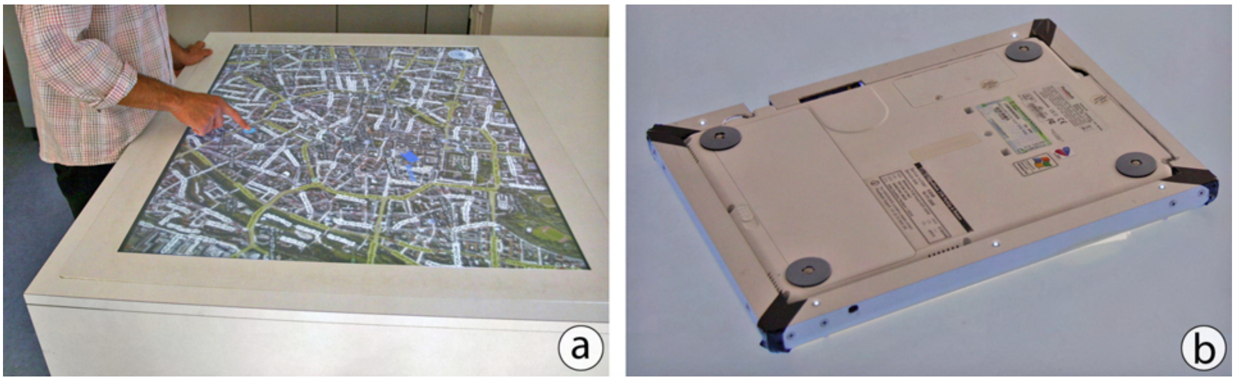


**Figure 3.6:** Devices used in our prototype: (a) Our external display is a multi-touch tabletop based on FTIR [Han05]. (b) The mobile device is embedded into a frame which contains infrared five LEDs.

Our tabletop system uses the principle of *frustrated total internal reflection* (FTIR) which was first introduced by Han [Han05]. Figure 3.6 shows our prototype devices. This method relies on infrared light traveling inside acrylic glass. The light is reflected totally within the panel when it is beyond a critical angle between its movement vector and the panel's surface. Another (soft) material touching the surface frustrates this total internal reflection letting light escape the panel orthogonally. This emitted light can then be captured using a camera that operates in the infrared spectrum by for example using an infrared filter. The back-projected nature of the original idea can be avoided through thin form factors while still relying on infrared light [HIB$^+$07]. SMART Technologies present a different approach which uses infrared LEDs in the display's bezels [SMA10]. Cameras in the corners detect but not identify objects (i.e., missing infrared light at certain angles) and calculate their position using angulation similar to systems using visible light [BHB07]. The aforementioned systems either still have a low resolution (i.e., thin form factors) or cannot identify objects. Furthermore, larger objects cause shadows in which other contact points cannot be detected when the system uses LEDs in the display's bezels. In our prototype, we chose to use FTIR for our tracking. Similar to recent approaches, we ensure that the tracking is not confused by sources emitting infrared light (e.g., lamps or the sun) by using a filter on the camera. This filter can only be passed by light with wavelengths close to the ones emitted by the LEDs attached to our surface (i.e., ±5 nm).

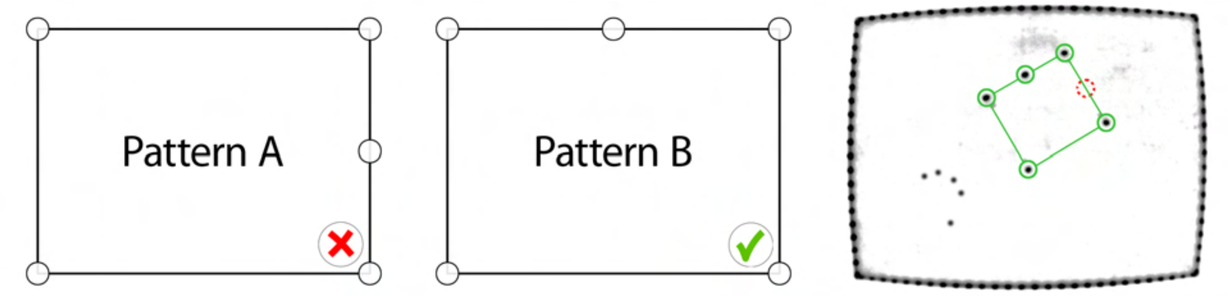


**Figure 3.7:** Different LED placements for mobile devices: Two possible patterns for LEDs on a mobile device. Right shows a sample frame from our tabletop with both a tablet PC as well as a hand being placed on it. In this scenario, pattern B can be detected.

As mentioned before, FTIR tracks soft objects touching the surface. However, the tablet PC we have used in our prototypes cannot be detected without modifications. Instead of detecting a contact point on the acrylic, the panel may be covered with a thin flexible foil which also serves as projection canvas. Contact points occur on the surface when the foil is being pressed down [HKI08]. While contact points from solid objects are now being detected as well, another problem arises. According to the laws of physics, one plane superimposed on another (i.e., the tablet PC on the tabletop) ideally results in three contact points. These points also carry the entire weight of the upper plane. Even if four points could be detected (e.g., the four corner points), determining the rotation of the mobile device within the tabletop's plane is ambiguous. In general, the three points would be sufficient for tracking the mobile device's position as well as its orientation. The positions of these points are not known a priori making it hard to determine the spatial relationship between the mobile device and the tabletop. Nevertheless, three fixed (i.e., known a priori) contact points placed in a linearly independent way would allow for identifying the mobile device. To ensure the tablet PC's visibility in the camera image, we decided to use a frame for the mobile device with built-in infrared LEDs. These LEDs are placed uniquely to allow for the identification of multiple devices at the same time. For stability reasons, we decided to use five LEDs which are placed as follows: first, four of them are located in the corners of the frame and thus serve as reference frame of the device. The fifth LED is placed at a predefined position on the frame's border. For each device, the last one is placed differently. Figure 3.7 denotes two placements of the LEDs for two individual devices.

Besides the points introduced by the frame, other touch points (i.e., fingers) may also occur on the tabletop simultaneously. The input needs to be processed in order to detect a device's pattern in the overall set of points. In our prototypes, we realized in multiple steps: first, the system takes two points out of the set that have the same length as the tablet's diagonal. Second, the system tries to match other points according to the locations of other LEDs in the frame using a small threshold. And third, when all five points have been found, the algorithm refines the detected shape according to the detected points. By doing so, we obtain the tablet's position on the tabletop as well as its rotation (i.e., the angle of the tablet's coordinate system with respect to the tabletop's coordinate system). These values are then used to construct an invertible transformation matrix between both displays. This allows for geometrical transformations of the content according to the display it is shown. Similarly, each input point coming from the tablet PC can be transformed into the tabletop's coordinate system using this matrix and its inverse respectively. Running this algorithm for all known LED configurations, the system is also able to identify the tablet in case multiple ones are present. This method fails if a pattern similar to the tablet's LED configuration is present on the tabletop since fingers and LEDs cannot be distinguished. One solution is to use a blinking pattern with a frequency higher than users could constantly tap on the tabletop.

### 3.3.2 Applications and Implementation

To demonstrate the aforementioned interaction possibilities with a tablet PC directly superimposed on a tabletop, we decided to implement two applications. In the first scenario the tabletop is rendering a digital map on its canvas while the mobile device allows for greater detail (i.e., higher pixel density) as well as a different view (i.e., a satellite image of the map's portion). As direct interaction with the tabletop is possible but rather coarse due to lower input resolution, precise interaction was mainly performed on the tablet PC. Such accurate interactions are, for example, defining waypoints for a planned route (see figure 3.8). Users can define a waypoint by performing a double tap on the respective point within the map. Once a pin is created, users can change the color or its label. They are further able to move the waypoint around freely on the mobile device. De-selecting a pin is realized by tapping on a free location on the map (i.e., no other waypoint is present). To select one pin in order to, for example, change its color, its label, or delete it, users have to double-tap the respective waypoint on the mobile device. As a side effect, the ability of identifying multiple mobile devices allows for showing routes only on the user's tablet PC. This in turn reduces visual clutter on the tabletop.

Interaction may also happen on the tabletop. Users can move the map by dragging it using a single contact point. The changes are communicated to the tablet PC to updates its view. With simple two-finger gestures, users can scale and rotate the map. For scaling, these fingers are moved closer to each other (and further away from each other respectively) to zoom out (and zoom in respectively). Rotating the map is done by rotating two fingers around a center point. Rotate and scale operations usually occur in parallel as users are most likely not able to precisely move their fingers along the fingers' connecting line (and around the rotation center respectively). The center of both operations is defined by the mid-point of the line connecting both contact points. The same interaction can be performed on the tablet PC as well. However, the device used in our prototype only allowed single-touch interaction. While moving the map was performed in the exact same way as moving it on the tabletop, we needed to change rotating and scaling. We decided to use a compass which allowed users to rotate the map by performing a circular drag operation within the compass. For scaling the map, we decided to use a simple slider within which users could drag to increase (and decrease respectively) the level of detail. These were also communicated to the tabletop to allow for correctly aligned visualizations.

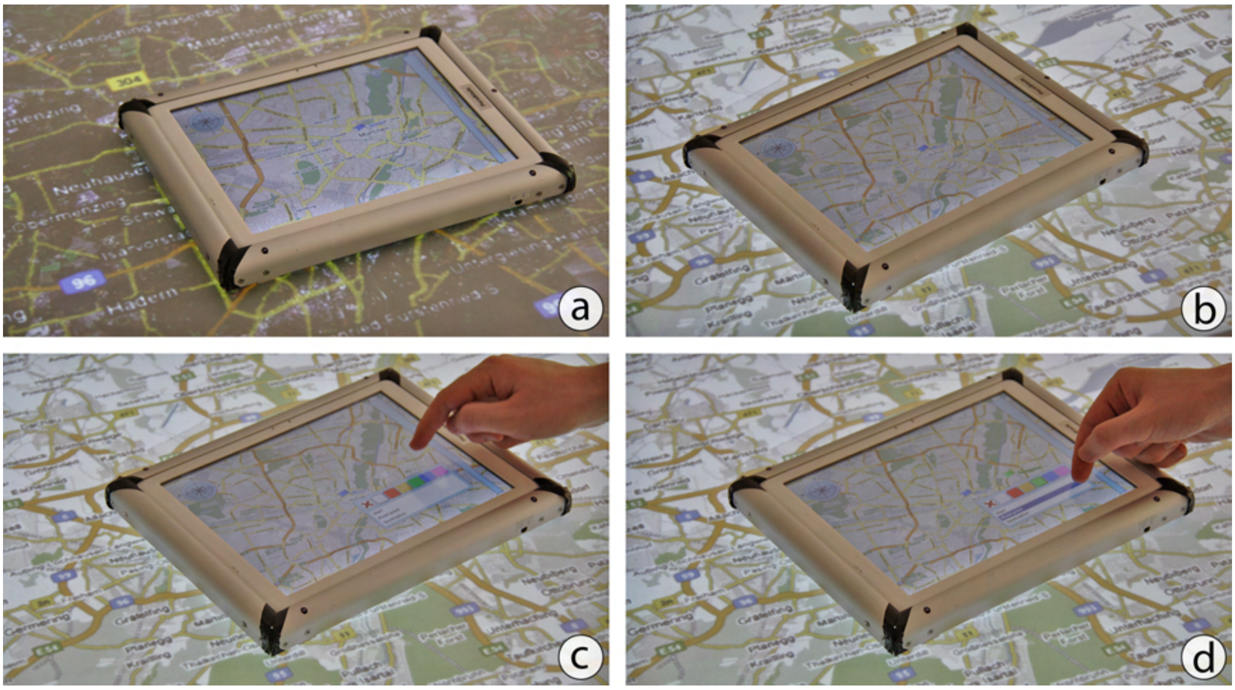


**Figure 3.8:** The tablet PC as *see-through* interface with the map application: The tabletop either renders a satellite image (a) or a street map (b) whereas the tablet PC shows a detailed view with higher resolution of the correspongin location. Users can define waypoints (c) and further manipulate them (d).

In the second application, we chose two independent content devices. This means, that both the mobile device as well as the tabletop showed their own content. While the map application was mainly used to test the *see-through* behavior presented earlier, this application allowed us to verify both the *fully opaque* as well as the *translucent* mode. As both devices hold their own content, we wanted to test the information transfer back and forth in the aforementioned way. In the fully opaque mode, users are able to use *Drag'n'Drop* across the mobile device's bezels. To do so, users started to drag an item on the mobile device. Once they reached the boundary of the mobile device while moving the item with a certain speed (i.e., more than ten pixels per seconds), a target area was shown on the tabletop display (see figure 3.9). The target area is based on (1) the velocity of the drag and (2) its direction. This approach is often referred to as Dead Reckoning [TKW08]. This is necessary as contact points on the mobile device's frame cannot be tracked. From the system's point of view, a drag contains two individual drag operations – one on the mobile device and one on the tabletop. If users continued the drag operation within the target area, the two drag operations were joined to form a single one. This dwell time needed to be less than one second to still allow for a smooth drag operation. By knowing the position of the mobile device, the interaction works likewise in the other direction.

While *Drag'n'Drop* worked in the translucent mode as well, users are able to perform a single tap on an item in order to place it on the other layer (i.e., not the layer it originated from). Figure 3.10 denotes the content transfer using a single tap. We further allowed users to switch between the layers shown on the mobile device. This allowed for selecting multiple items on the active layer by performing a free-form selection. Switching the layers can be performed by lingering on an empty region as described earlier. As long as the local layer (i.e., the mobile device's own one) is shown, all input events are directly processed by the mobile device. When the remote layer is the active one, each input event is sent to the tabletop and is transformed into the tabletop's coordinate system for further processing. Similar to the map application, the entire canvas can be dragged virtually by performing a drag operation with a single contact point. While this has no effect on the mobile device in the *fully opaque* mode, such changes need to be communicated in the *translucent* mode. Furthermore, dragging the canvas of the mobile device does not have any effect on the tabletop's content. If users perform such an operation on the tablet PC while the remote layer is the active one, the changes affect the tabletop's content.

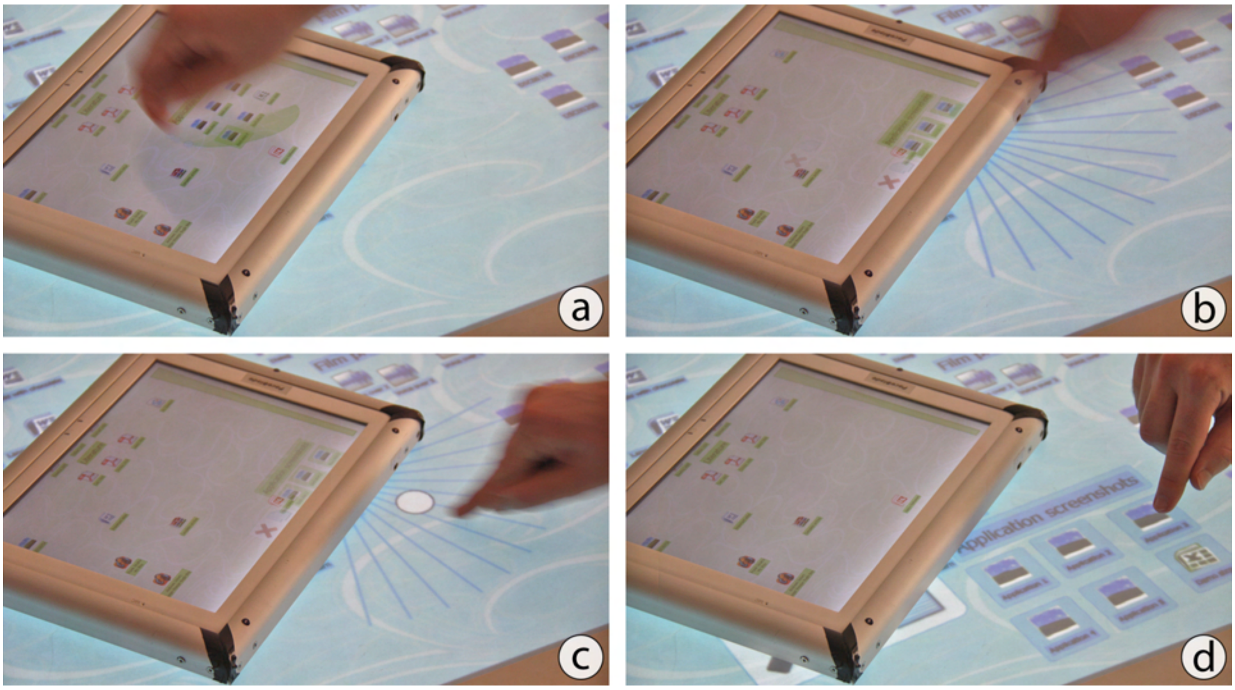


**Figure 3.9:** *Drag'n'Drop* across the tablet PC's bezels: (a) The user selects a group of items with a free-form selection. (b) Dragging the content close to the tablet PC's edge causes the target area to be displayed on the tabletop. (c) The first contact point on the tabletop needs to be within the target area. (d) The items have been transferred successfully.

Each of these applications has been implemented using the same prototype. As stated before, the tabletop uses a back-projected screen and allows for direct-touch interaction through the principle of FTIR. The screen's diagonal has a physical length of 50 inches while having an aspect ratio of 4:3. The tabletop's width is 39.4 inches and its height is 29.6 inches. The resolution of the projector used was 1024 × 768 pixels leading to a pixel density of 26 dots per inch (dpi) in both horizontal and vertical direction. The mobile device is based on a *PaceBlade*[1] *SlimBook P120*. The device has a screen diagonal of 12.1 inches while having the same absolute resolution as

[1] *PaceBlade Technology*: http://www.paceblade.com/

the tabletop. The mobile device's pixel density is 106 dpi (i.e., about four times higher than the tabletop's one). This means that each pixel on the tabletop was covered by 16 pixels (i.e., 4 × 4 pixels) on the mobile device. The code for all applications is written in C# (.NET Framework 3.5) and WPF. The layouts of different desktops have been defined a priori using an XML file. The map tiles have been downloaded from Google Maps [2].

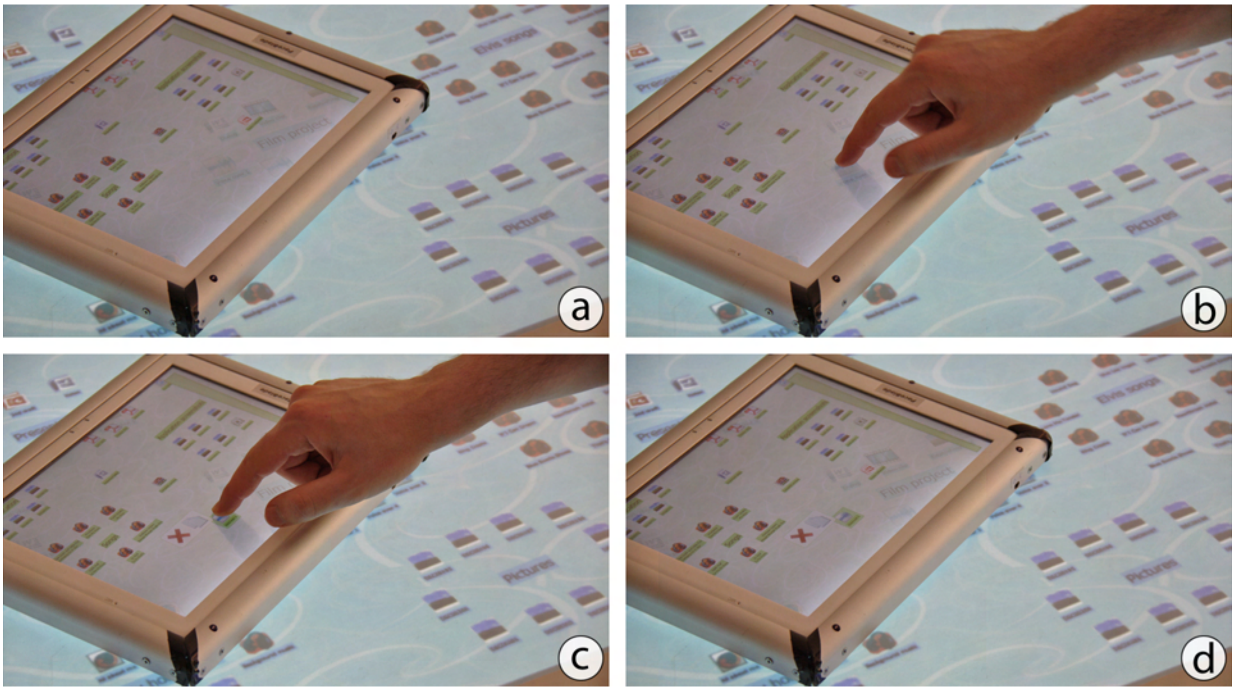


**Figure 3.10:** Cross-layer content transfer in the *translucent* mode: (a) The remote layer slightly shines through the local layer. (b) The user taps on a remote item. (c) The item is brought to the local layer. (d) The content has been transferred successfully.

## 3.4 Observational Study

In a preliminary observational study we wanted to test the previously presented prototypes. To gain first insights into having a movable mobile device on a larger canvas, we recruited volunteers that interacted with both the map application as well as the desktop application. When the map application was used, we decided to render (1) a satellite image of Munich on the tabletop and (2) the corresponding street map on the tablet PC. We had 13 participants in our study (four female), ranging in age from 21 to 29 years (average age was 24.6 years). Eleven of them were (post-graduate) students in computer science. All but two reported that they already are familiar with touch-based interaction (eight with multi-touch). Most of them, however, used this type of interaction on mobile phones. Seven had perfect sight while the vision of remaining participants was corrected to perfect sight. None of them reported any color blindness.

[2] *Google Maps*: http://maps.google.com/

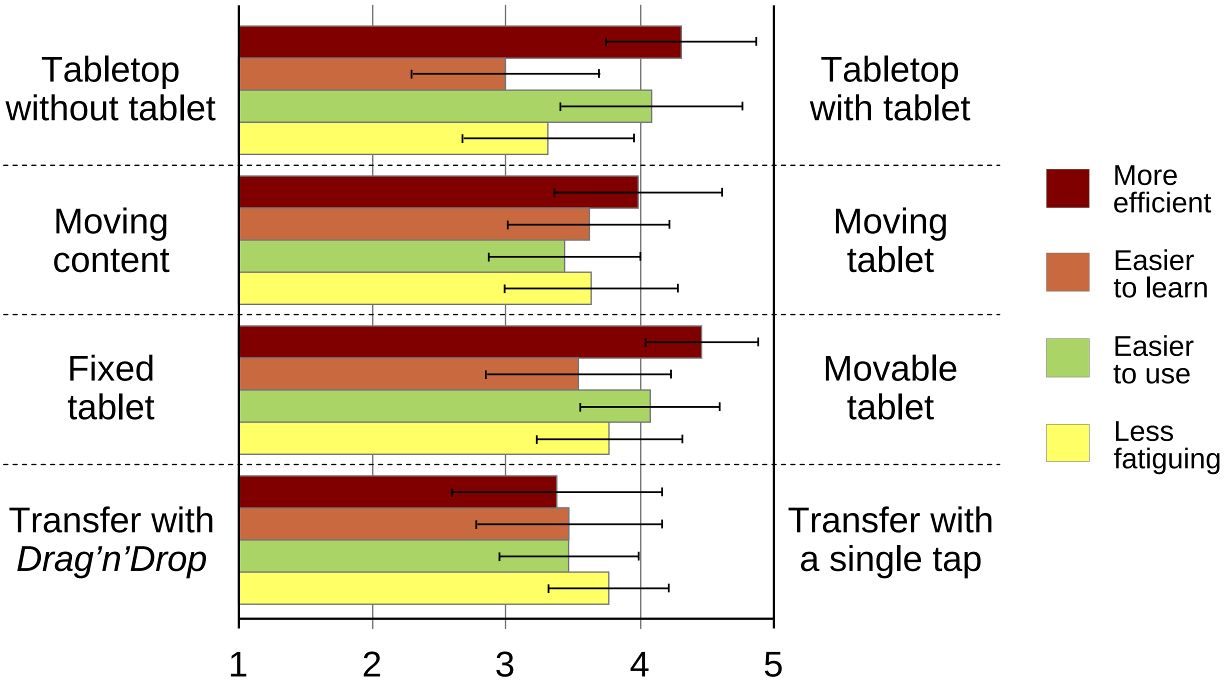


**Figure 3.11:** Subjective ratings by our participants when comparing two conditions against each other. Error bars indicate ± standard error of the mean.

Each of the participants was given a short training of the two applications and their features. Subsequently, they could play around with the system until they felt comfortable using it. We then required the participants to use the mobile device in *fully opaque* mode. In this setting, they had to drag items of their choice from one display to another across the tablet PC's bezels. After ten items (five in each direction) were transferred, we switched the mobile device into the *translucent* mode. Now the participants had to transfer ten items using a single tap. For five of the four transfers, they were not allowed to move the focus device imagining that it has been screwed to its position at the lower center of the tabletop. In this case, they had to move the canvas on the tabletop to bring the item of interest into the tablet PC. When they were allowed to freely move around the mobile device, it was their choice whether to move the content or move the tablet PC. After completing these tasks with the desktop application, they were presented with the map application. We asked them to create two routes by placing two waypoints at exact locations. In one case, they were allowed to use the tablet PC while the other route had to be created without it. Each session lasted for about 20 minutes and was video-recorded for later analysis. Once participants completed the tasks, they had to a post-questionnaire.

Based on our understanding of the applications as well as the system itself, we had several assumptions that we wanted to verify using these tasks: first, we hypothesized that users will prefer using the tabletop in combination with the tablet PC when creating the routes in the map application. Second, we assumed that participants prefer a movable focus region (i.e., the tablet PC) as opposed to a fixed region. Third, users would also prefer moving the mobile device instead of moving the content underneath (if possible). And fourth, dragging content across the tablet PC's bezels will be less preferred compared to transferring content using a single tap. We aimed to verify these assumptions by analyzing the video data recorded during the study. We paid close attention to the participants' impromptu feedback while thinking aloud. Furthermore, we observed the actions they performed when they had the choice, for example, moving the tablet PC versus moving the content. In post-questionnaires we asked for the participants' subjective feedback using standardized device-assessment questions [DKM99].

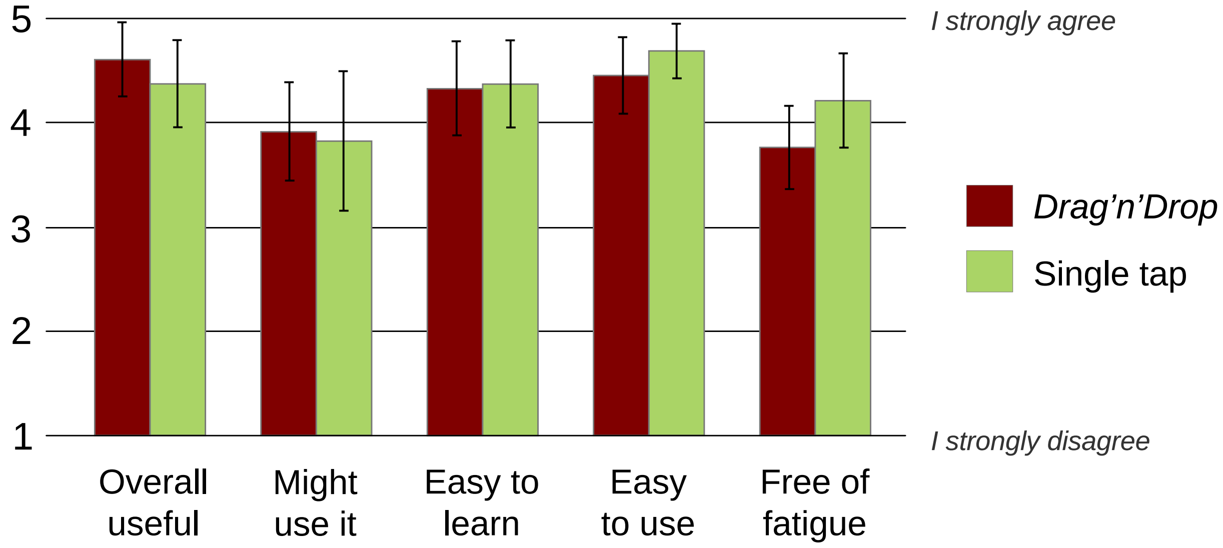


**Figure 3.12:** Separate ratings for *Drag'n'Drop* and single tap content transfer. Error bars indicate ± standard error of the mean.

First and foremost, we wanted to figure out whether the tabletop alone is preferred over the combination of both devices. We used ratings that compared both setups on five-point Likert-scales (1 equals "tabletop without tablet PC" and 5 equals "tabletop with tablet PC"). 77% of our participants rated the combination as being more efficient and 62% stated that the combination is easy to use. On average, both setups were ranked easy to learn. One participant thought that the tabletop alone was much easier to learn while three participants stated that the combination was the easiest. When asked which setting was less fatiguing, only 23% favored the stand-alone multi-touch table. In contrast, 46% stated that the combination introduced less fatigue. This may be explained by the touch sensitivity of the used table causing fatigue in the participants' fingers when moving the entire canvas. Nevertheless, these results show a trend indicating that our assumption of the combination being preferred may be correct. We also wanted to know how a fixed focus region (i.e., the tablet PC) with a movable canvas on the tabletop compares to a movable one. We again used a five-point Likert-scale (1 equals "fixed focus region" and 5 equals "movable focus region"). 85% of our participants clearly favored the dynamic solution as being the more efficient one. This is inline with the observations during the user study. It appeared that participants needed overall less time when moving the tablet PC compared to moving the canvas on the tabletop. This, however, was only observed in the videos recorded during the session without any logging mechanisms. The majority further stated that the movable mobile device is more efficient (69%), easier to use (62%), and less fatiguing (85%). This indicates that our assumption of a dynamic mobile device being preferred over a fixed one may be correct. Figure 3.11 summarizes the results of these questions.

We were further interested in the opinions of our participants regarding the two different content transfer mechanisms (i.e., *Drag'n'Drop* and using a single tap). As illustrated by figure 3.11, both techniques received rather good ratings. We again used a five-point Likert-scale to directly compare both techniques (1 equals "*Drag'n'Drop*" and 5 equals "single tap"). 54% of our participants rated the single tap as being more efficient, easier to learn, and less fatiguing. When asked about the ease-of-use, 62% of the participants were undecided. Nevertheless, with 31% stating that the single tap was easier to use while only 8% favored *Drag'n'Drop*, a slight tendency is indicated. These ratings can be explained by the rather thick device used (i.e., about one inch of height). One solution to avoid the rather thick device influencing on the results is to use a top-projected sheet of paper as mobile device. 77% further stated that the visualization of the target area was helpful in accomplishing a *Drag'n'Drop* operation across the tablet PC's bezels. When we asked participants about the techniques individually, *Drag'n'Drop* was considered more fatiguing by 11% compared to the single tap (see figure 3.12). In all other categories, both techniques received equal ratings with no value differing in more than 5%. These results slightly indicate that our assumptions may be correct. Nevertheless, further studies (e.g., with the aforementioned sheet of paper) need to be conducted.

## 3.5 Summary and Discussion

In this chapter we have presented a first prototype that enables users to place their mobile device directly on top of a large external screen (i.e., no distance between both involved displays). From the user's point of view, the connection is initiated when the mobile device is superimposed directly on the tabletop. While being implemented differently (i.e., a steady connection between all devices throughout the entire interaction session), users do not have to connect the devices up-front in a separate mode. This setup has two important factors due to the accessibility of both screens: first, both displays are now accessible and (if both screens are interactive) allow users to directly interact with them. And second, the (usually) higher pixel density of mobile devices allows for sub-pixel accuracy which turns the overall system into a dynamic focus plus context screen [BGS01]. These factors also allow for various roles of the mobile device. While it may act as *see-through* device enabling higher input and output resolutions within a limited region, the mobile device can also host the user's personal content. In the latter case, we distinguished two modes: first, the mobile device can render its canvas in a *fully opaque* way hiding remote content located underneath the device. And second, the content can be rendered in a *translucent* way allowing for remote content being visible.

When the mobile device is directly superimposed on the external screen, it first needs to be identified by the large display. We implemented a prototype that uses a large multi-touch tabletop for both interaction and tracking purposes. We placed infrared LEDs in frames of mobile devices to identify them and detect their position and rotation with regard to the tabletop's coordinate system. Based on this prototype, we created two application scenarios: first, a satellite image was rendered on the tabletop while a tablet PC showed the street map of the corresponding region. In this application, users gained higher detail of a certain portion of the map by placing the tablet PC on the desired area. Similarly, the mobile device can show the same content with higher resolution turning it into a pure *see-through* device. And second, we displayed individual content canvases on both screens. In this application, we wanted to test both the *fully opaque* as well as the *translucent mode*. In both modes, dragging content across the bezels of the mobile device was possible. In the *translucent* mode users were also able to transfer content between devices using a single tap on the respective item.

Based on the prototype and the two applications, we conducted a preliminary observational study. We found that the usage of a dynamic focus region (i.e., a movable mobile device) is preferred over only having the large interactive tabletop in terms of *ease-of-use*, *fatigue effects*, and *efficiency*. Most surprisingly, participants in our study rated both settings equally in terms of *easy-to-learn*. In this category, we expected the tabletop to be rated higher considerably as using both devices introduces more necessary skills. Nevertheless, preferring a movable region featuring a higher resolution over a fixed one is in line with Hsiao et al.'s findings [HCH$^+$09]. Regarding the two content transfer mechanisms, the single tap was clearly favored over the well-known *Drag'n'Drop*. The technique does not require any dragging but users need to move the entire mobile device instead. The low ratings of *Drag'n'Drop* had another source according to the feedback we gained during the study. Users disliked (1) the surface characteristic on the tabletop as being too rough and (2) the change in height when dragging across the mobile display's bezels as too large. Both factors may have influenced on these ratings and favored the single tap method. Nevertheless, combining the fact that moving the device is favored over moving the content may with the time-saving method of a single tap, the content transfer *through the display* may still be the better choice. However, this needs to be verified in the future. We hope that this would further lead to more quantitative data as well as more qualitative feedback. Furthermore, recent advantages in touch screen technologies suggest thinner portable devices (such as Apple's iPad [3]) which are capable of multi-touch input.

Overall, the prototype described in this chapter fits into the taxonomy (see chapter 1) as follows: the input is performed in an *absolute* and *direct* fashion while allowing both *discrete* and *continuous* interaction. Placing the tablet PC on top of the tabletop, we were able to create a *non-modal* connection mechanism at a short (i.e., *limited*) distance. As the device is tracked by the tabletop, the environment only has *limited* flexibility. In the following chapter we will investigate the interaction with external displays using mobile devices at a distance. At the same time, we analyze the effects on the remaining two criteria – namely *modality* and *flexibility*.

[3] *Apple iPad*: http://www.apple.com/ipad/

# Chapter 4

# *MobileVue*:

## Relative and Indirect Pointing on Distant Screens

*It is common sense to take a method and try it: If it fails, admit it frankly and try another. But above all, try something.*

**– Franklin D. Roosevelt –**

In the previous chapter we showed how users can interact with an external display that is directly accessible. However, as discussed in chapter 1, several displays may be out of the user's reach due to their size (i.e., influences the viewing distance) or placement (e.g., protection against vandalism). These issues do not allow interacting with the external display by directly superimposing the mobile device. Instead, users need to be able to interact on the remote display using their mobile device at-a-distance. In this chapter we[1] review the use of the mobile device as a *relative* and *indirect* pointing device. We focus on the different phases as described in chapter 2: (1) *connecting* to the target display, (2) control a remote pointer (*acquire* content), and (3) *manipulating* content with the mobile device. We present an architecture that facilitates the target display selection for environments with multiple screens by narrowing down the potential target displays according to their characteristics. The results are further presented depending on (1) their distance to the user and (2) their temporal availability. In an experiment regarding the pointing phase, we found that the mobile device as optical mouse appears to be the most efficient technique. Furthermore, we show that placing the tool palette in the manipulation phase on the mobile display is more efficient than having tool palettes on the external display when interacting in a *relative* and *indirect* fashion due to less pointer movements required.

[1] The work presented in this section has been published in several scientific papers [JBR08a, BJB09, JBR08b]. The scientific plural refers to all authors of these publications – namely Sebastian Boring, Andreas Butz, Marko Jurmu, and Jukka Riekki.

The remainder of this chapter is structured as follows: We start by describing the necessary infrastructure to allow selecting from a larger set of displays in the environment (see section 4.1). Subsequently, we review pointing techniques for external displays once users are connected (see section 4.2). After users addressed content, we show how the mobile display is of use when remote content needs to be manipulated (see section 4.3). We conclude this chapter by summarizing the approach and discussing its strengths and weaknesses in detail (see section 4.4).

## 4.1 Selecting the Target Display

When *relative* and *indirect* pointing is used, connecting to the target display cannot be done in the *non-modal* fashion as presented in chapter 3 without further modification. Users have to explicitly select the device they want to interact with. At the same time, the system needs to represent surrounding displays according to the users' perception to facilitate easy and fast selections. A simple solution is to present a list to users allowing for a selection of the display of interest. With increasing numbers of displays in our surroundings, such lists may be rather long. On the other hand, the number of items in this list can be further reduced by excluding (1) displays that are unavailable (e.g., already occupied by other users) and (2) displays that are not capable of performing the requested task. For the latter case, however, the environment needs to know (1) the intended interaction of users and (2) the requirements of the intended task. In case a display close to them is not available, users may either wait until it is available (*temporal selection*) or go to another suggested display that may be further away (*spatial selection*). The users' mobile device needs to facilitate such selections. This type of connection handling is often referred to as *context-aware leasing* [JPR07]. In our prototype we utilize this concept of connecting to displays in order to explore the dependency between temporal and spatial selection and its according representation on the mobile device.

In this section, we first discuss the properties of displays in the environment as well as their influence on the selection process (see section 4.1.1). Subsequently, we describe the overall infrastructure and its inter-component communication to allow for spatial as well as temporal selection (see section 4.1.2). To allow users to make their final decision based on waiting time and distance to the display, we implemented a client application running on a mobile device. This front-end visualizes the results and enables users to select the best display (see section 4.1.3).

### 4.1.1 Static and Dynamic Properties of Displays

As mentioned before, presenting a list with *all* displays in the surrounding may lead to increased selection times due to the limited screen real-estate present on mobile devices. Several displays can be excluded a priori if they do not meet the minimum requirements for a certain task. For example, viewing pictures stored on a personal device may not be wanted on a display built into a ticket machine due to its size and (possibly) frequent usage. On the other hand, obtaining a ticket for public transportation has to happen on a display that is actually capable of issuing the ticket. Hence, the user needs to state these minimum requirements in the very beginning. Considering numerous properties, however, this may be cumbersome and time-consuming. On the other hand, embedding the connection process into an application, may overcome this by letting the application define the minimal set of requirements of a target display. Similarly, if these properties are defined for each target display as well, an incoming request from a mobile device can be matched to determine the suitability of the corresponding display.

Properties describing displays can be broken down into two major categories - *static* and *dynamic properties*. Static properties define characteristics of a display that are not easily (or not at all) changeable. Such properties include the display's size (either physical size or pixel resolution), or the space it resides in (e.g., privacy-related information). They can be preconfigured in a display once it is set up in the environment and most likely will not change over time. Dynamic properties, on the other hand, may change more frequently than static ones. These properties, for example, include the availability of the application binaries on a certain display. It is apparent that this property may change over time since additional binaries (i.e., applications) may be installed on or removed from the target display. In contrast to preconfigured static properties, the dynamic ones need to be determined at runtime. For the aforementioned characteristics this appears to be a trivial task. For example, the software running on a display may check for the corresponding application once a request arrives.

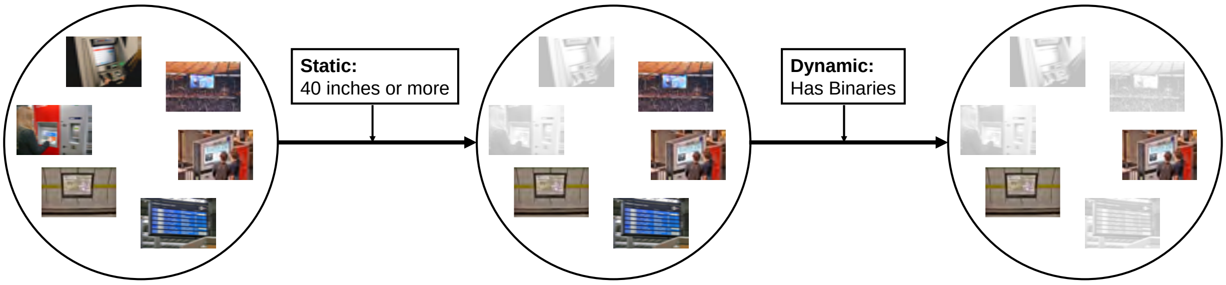


**Figure 4.1:** Sub-selection of displays with static and dynamic properties: first, *static* properties are used to decrease the amount of displays (here: screen size). Subsequently, *dynamic* properties are used to determine the remaining displays and present them with their *spatial* and *temporal* properties. The faded displays have been discarded during the process.

Dynamic properties may also include the user. The most common one is the *spatial availability* of a display. It describes whether the display is closer than the maximum *distance* the user would move. In large spaces, this property may be a powerful determination of a display's suitability for a given task. For example, if users start a display discovery request while they are located on the one end of a shopping mall, the display on the other end may not be the preferred one. To determine this property, the user needs to be tracked (i.e., the environment knows where the request approximately is coming from) whereas the positioning accuracy needs to meet the environment's size. This means that the smaller the environment is the more precise the tracking needs to be in order to reduce the resulting list of potential target displays. The closest display (i.e., the display with the best spatial availability), however, may already be in use by another person. The display may be close but cannot be used at the moment.

Certain tasks may require the use of a display for certain time period only. For example, users should be able to obtain a ticket for public transportation within a short time (assuming a good and understandable user interface). Furthermore, the usage of large displays may be restricted in time as well in order to also allow others to use the display. One concept in doing so is to grant a *lease* to a display for a certain period of time depending on the task. In such scenarios, a display may be unavailable at the moment, but is usable again once a lease expires. In this scenario, the environment is able to determine the maximum waiting time for each display. We refer to this component as the *temporal availability* of a display. Since this property will change frequently, it also falls into the category of dynamic properties. The combination of both spatial and temporal availability then results in a more complex, overall availability of a display. Determining the best suitable display for the user based on this complex property is a rather difficult task as it highly depends on the user's preferences. For example, a display close-by may be in use for the next ten minutes while another one further away is immediately available. The system does not know whether the user prefers waiting over walking to the display further away. While both spatial and temporal availability are important for the overall selection, they need to be presented to the user for a final decision based on individual preferences.

As shown in figure 4.1, the static and dynamic properties can be used to limit the set of potential target displays. First, the static properties already filter out displays that are not usable in any case (even if they would be in range and currently available). Subsequently, the filtered displays are checked whether they are capable of running the application. The remaining two properties (i.e., the displays' spatial and temporal availability) may then be presented in a two-dimensional view to allow the final decision being made by the user.

### 4.1.2 Obtaining the Displays' Properties

Several steps need to be done to allow the aforementioned selection and connection process: (1) connect all displays in the environment for communication purposes, (2) allow ad-hoc communication between a mobile client and a display in the environment and (3) roughly locate the user for distance measurements. We chose to only use a coarse positioning method assuming that the environment is considerably large. For this we introduce new components in the environment as described in this section. Furthermore, we need to present the results of a discovery to users on their mobile device (see section 4.1.3).

Each display in the environment is driven by a computer attached to it. These computing entities need to communicate to each other to determine the best subset of displays for a user's request. In our architecture, we rely on the *Fuego* toolkit, a publish/subscribe routing system [Tar08]. This toolkit is similar to the *EventHeap* but allows greater flexibility as it is purely based on the HTTP request/response model [JF02]. Clients subscribe to certain events they wish to receive. These events are messages that can be identified by their name and the content types. Through this routing system, all control messages are sent between displays. Ideally, the mobile terminals are integrated into this toolkit as well. For positioning reasons we chose a separate connection as described in the following.

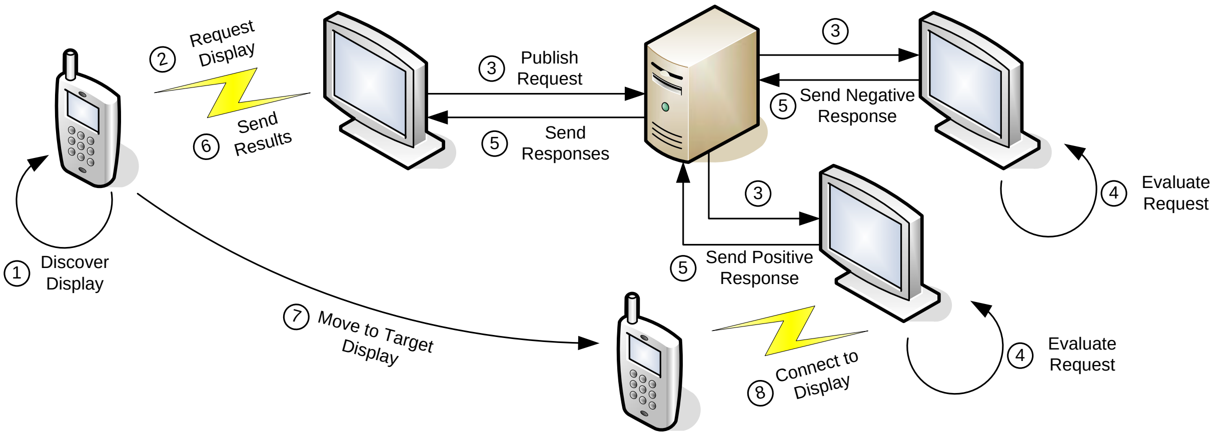


**Figure 4.2:** Communication during the selection process: After discovering a nearby display (1), the request is sent (2). The request is sent to all displays with location information (3). Each display evaluates the request (4) and replies with a positive or negative response (5). The results are sent to the mobile device (6) allowing users to select a display and walk to it (7). Once arrived there, they can initiate the connection over wireless LAN (8).

If the displays are stationary and their location is known, using the aforementioned communication is sufficient. However, mobile clients need to be located once they send a request. Naturally, this may work with WiFi localization techniques which in turn require multiple routers to be present in the environment [CCLK05]. With the assumption that the user is at least close to one display, we can use the range limitation (i.e., couple of yards) of Bluetooth as an advantage. This approach does not allow for inch-level accuracy, but at least give a clue of which display is the closest to the user. Considering large scale environments (such as shopping malls), this knowledge is sufficient. For example, walking 100 yards to the next display is not different than 110 yards from the user's point of view. In our architecture, the mobile device uses the closest Bluetooth-enabled display (discoverable by a service UUID) and communicates through this channel with the display. The display itself acts as a gateway to the entire infrastructure. The distance of all other screens to this *seed* display can be immediately determined. As the display density may be higher than necessary for entire Bluetooth coverage in the environment, not every screen needs to be Bluetooth-enabled. It is sufficient to allow full coverage in the space. Additional Bluetooth hotspots may be installed in the environment if the density is much lower than required allowing for the same approach.

After discovering a nearby display (i.e., within the Bluetooth coverage area), the mobile device sends its request containing the restrictions needed for the application such as the display's minimum screen size and the maximum distance. The close display then enriches this set with location information and publishes this information on the network. Other registered displays will get this message and evaluate whether they are still in range. If so, they reply with their current status regarding availability. The nearby display sends this information back to the mobile client allowing users to select the target display. After selecting it, the nearby display gets notified and initiates the leasing in two steps: first, it reserves the display either immediately (if possible) or places a lease after the current interaction. Second, it sends connection information (i.e., IP address and port) to the mobile client. The user can now walk to the display (if necessary) and connect to it once the lease is active. In case the lease could not have been set immediately, the mobile client gets notified when it is active. Figure 4.2 summarizes this procedure.

### 4.1.3 Representing Properties on the Mobile Device

The aforementioned properties can be used to narrow down the list of results based on both static and dynamic properties. The spatial and temporal availability can only partly be taken into account if the user has entered preferences such as the maximum distance (*spatial*) or the maximum waiting time (*temporal*). The remaining displays that meet all the criteria now need to be visualized to allow for user selection. In a first attempt of visualizing the results, we chose to use well-defined and widely used graphical representations of both time and distance. The parameter time can be depicted by a clock-based metaphor whereas distance can be illustrated using a radar-like interface. Radar views usually give both direction and distance. A clock on the other hand only gives the time in a radial view. As we are not attempting to obtain the user's direction, we chose to replace the direction parameter with the time parameter (see figure 4.3). In this representation, the angle from north up indicates the time (increasing clock-wise) whilst the radial component (i.e., the distance form the center) shows the distance of a display.

The limited screen real-estate of mobile devices does not allow for showing a live preview of all the displays (e.g., taken by a webcam). If users are not familiar with the environment they need to be able to identify a display based on its visual appearance rather than just on its spatial and temporal properties. To overcome the screen's limitation, we allow users to select a display in the visual representation and request additional information. These include the display's location (e.g., ”*next to Starbucks*”) as well as a photograph of the display and its surroundings for later identification. Selecting a display is done by either using the arrow keys of the mobile device or (if possible) direct touch. If the selected display is suitable, the user can send a request the display which causes the mobile device to send a message to the nearby display. If users are familiar with the environment, they can, of course, also request the display directly from the overview representation. The response is visualized either with a checkmark indicating the successful acquisition of a target display (and prompting the user to move towards it) or an error screen allowing users to go back and select another display in the environment.

Showing our prototype to potential users revealed that mixing spatial and temporal information lead to a confusion as several users still perceived the time as directional parameter. When we explained the time-component, users understood it and were able to determine the best display. Nevertheless, we thought about an alternative visualization to overcome this issue. As shown in figure 4.3, using a two-dimensional layout with *distance* and *time* as *x*- and *y*-axes respectively leads to a corridor of suitable displays which is placed around the diagram's diagonal. Displays outside this corridor are either immediately available but far away or close-by but still in use for some time. Users can personalize their corridor by using their own trade-offs between walking and waiting. The perception of this visualization was preferred over the previous one.

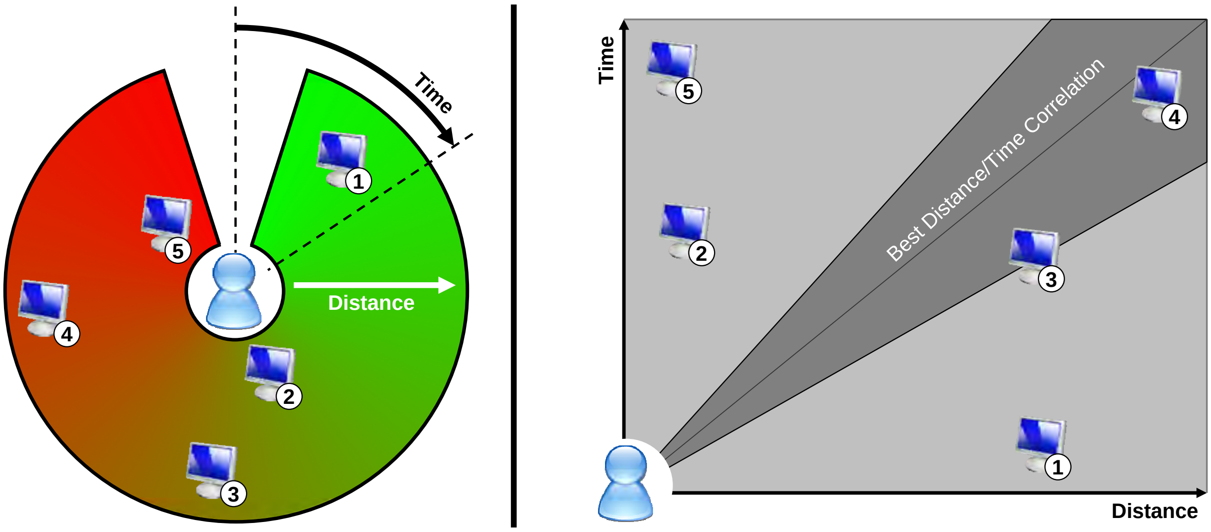


**Figure 4.3:** Different visualizations of temporal and spatial availability: Left denotes the first approach used for target selection. The radial component illustrates the *spatial* availability, the clock-wise component depicts the *temporal* availability. The right visualization is a two-dimensional graph showing the same displays. The dark area gives the best distance-to-time correlation. The displays' numbers denote the same displays in the environment.

When they chose a screen, got a lease and finally connected to it (i.e., they walked to the display where they are connected automatically), users can select the content to be transferred onto the target display in a list-based interface. Once the content of interested has been chosen, they initiate the transfer causing it to be shown on the external screen. Furthermore, a personal pointer is displayed. In case multiple users connect to the same display for collaboration purposes, the pointers can be distinguished by color. In the next section, we present techniques to control these pointers using the sensing capabilities built into the users' mobile devices.

## 4.2 Pointing on the Target Display

For controlling a pointer on a display, personal computers and laptops offer a mouse or equivalent devices such as a TrackPoint, a touchpad or a graphics tablet. While this interaction concept seems suitable for single users in desktop environments, it needs to be extended for use in public spaces. Nevertheless, it provides an acceptable mental model as starting point since users are familiar with controlling a pointer in a *relative* and *indirect* fashion. However, controlling a remote pointer using modern mobile devices can be done in several ways. One obvious solution is to utilize the device's joystick and directional keys to steer the pointer in the respective direction. Recently, such devices also have cameras and acceleration sensors built-in which in turn can be used to control such remote pointers as well. These sensors may allow for greater flexibility in terms of the movement's direction (i.e., it is not only possible in discrete angles) as well as the pointer's velocity (i.e., users can adjust the movement).

In this section, we discuss three techniques to control a personal pointer on a remote screen using the mobile device's input and sensing capabilities. First, we introduce the three distinct strategies for continuous control (see section 4.2.1). Subsequently, we present an evaluation which reveals that using either the device's movement or its acceleration is faster but may introduce errors (see section 4.2.2). Finally, we discuss the techniques and present limitations (see section 4.2.3).

### 4.2.1 Relative Pointing Mechanisms

The simplest control is to map key presses on the mobile device and convert them into a constant motion of the pointer. This binary behavior causes the pointer to either move with constant velocity in the given direction or remain still at its position. The mobile device's joystick is commonly used to implement this approach (cf. [SMK01]). This technique gives both direction and movement at the same time with a single key press. If directional keys (i.e., arrow keys) are used, only eight angles are possible. This limitation comes from the stand-alone design for mobile device menus, which usually do not require pointing at arbitrary positions. If these keys would sense the pressure though, they could of course allow more angles. Furthermore, the missing pressure information only allows for constant motion increasing the travel time (and the overall task time respectively) for distant targets. Decreasing the constant CD ratio is possible by using ballistic pointer movements as known from traditional operating systems. Once the cursor reaches a certain speed, users will most likely stop it too late resulting in so-called *overshooting effects* requiring them to perform a backward movement towards the target. As this technique mimics a scrolling behavior to some extent, we refer to it as *Scroll*.

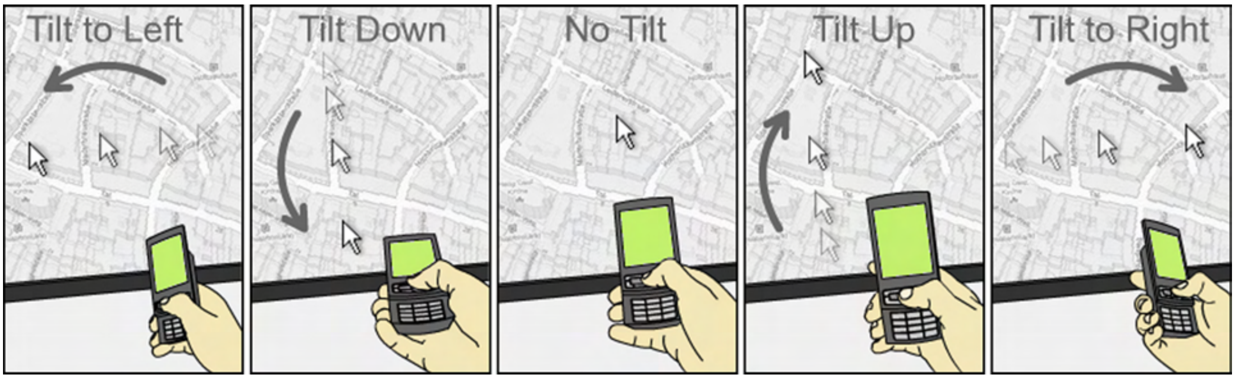


**Figure 4.4:** Controlling a pointer by tilting the mobile device: the more the device is tilted in a direction, the faster the pointer moves in this direction.

To overcome the problems given by the constant pointer control, we use the device's acceleration sensors (cf. [VCBE08]). Similar to the input style used by analogue joysticks, we map the tilting angle of the device to the pointer's speed on the external screen. The direction of a tilt also determines the direction of the pointer. The more the user tilts the phone in a certain direction, the faster the pointer moves in that direction. The built-in acceleration sensors give different gravity measurements according to the device's movement and inclination. We can construct different mappings between the sensor values and the two-dimensional mouse movement. One option is to use the tilting angle as indicator for acceleration. This means that the pointer accelerates in the given direction as long as the user tilts the device. Stopping the pointer is then done by bringing the device back into the initial state. A more common way is to use the rotation around the device's local *x*- and *y*-axes to control the speed in each direction. This mapping allows moving the pointer in arbitrary directions while the pointer's speed is still controllable. We decided to use the latter mapping in our implementation (see figure 4.4). For both mappings, however, the zero point (i.e., the point where no movement occurs) is a small interval around the device's regular viewing orientation. This allows users to keep the pointer steady although the device would sense small changes in acceleration due to slight hand movements. As this technique determines the pointer's speed and direction by tilting the device, we refer to it as *Tilt*.

The third technique maps the device's movement linearly to the pointer's motion (similar to *Sweep* [BRS05]). The device's camera is used to determine motion changes by using *optical flow analysis* [HS80]. In contrast to *Sweep*, we decided to evaluate the motion on the external display's computer to allow for high update rates and comparable results. We acknowledge that letting the mobile device evaluate the camera image would lead to higher scalability. If users want to move the pointer horizontally, they simply move the device along its local *x*-axis. Similarly, if they move the device along its local *y*-axis, the pointer moves vertically. Both movement directions, of course, can be combined in a single (potentially diagonal) movement. The device's camera does not need to be pointed towards the screen during the interaction. If users wish to decrease the CD ratio temporarily, they simply rotate their wrist (and the device respectively) which results in higher optical flow within the camera image. More precise controls are achieved by linearly moving the device (see figure 4.5). We refer to this technique as *Move*.

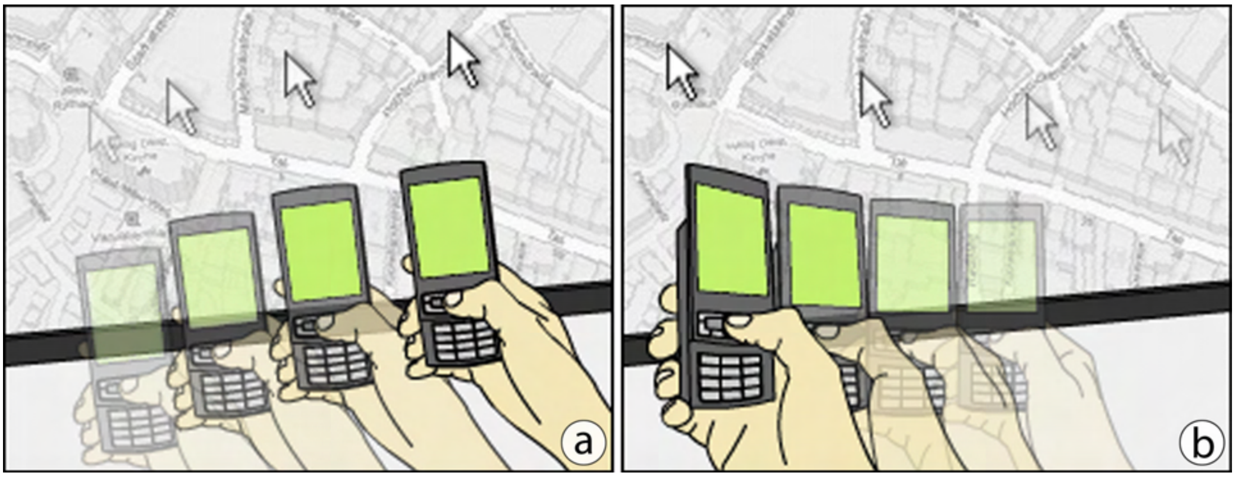


**Figure 4.5:** Moving versus tilting the mobile device: a) shows the linear mapping when the mobile device is moved. b) depicts how tilting the device increases the pointer's speed.

We utilize the mobile device's select button (e.g., pressing the joystick) to indicate a selection. Pressing the button can be translated into a *down* event. Releasing it in turn corresponds to an *up* event. If the pointer is moved in between these two events, the pointer is able to perform a *drag* operation. Moving the device without the button being pressed results in *hover* operations. By simulating a mouse button on the mobile device, we can reproduce common behavior known from traditional cursor-based user interfaces (e.g., ***W**indows*, ***I**cons*, ***M**enus*, ***P**ointers*). This only simulates pressing the left mouse button. Right clicks, however, can be enabled by pressing a secondary button on the device or pressing the select button longer while keeping the device still. The latter seems to be rather difficult (at least for *Tilt* and *Move*) as the natural hand tremor may cause slight movements.

### 4.2.2 Evaluation

In an experiment we tested the suitability of the aforementioned techniques for controlling a remote pointer on an external display. Participants of our study acquired targets of different sizes and positions using each of the techniques. This section briefly describes the experimental evaluation and its results regarding selection time and error rate. A detailed description of the experiment and its results can be found in [BJB09].

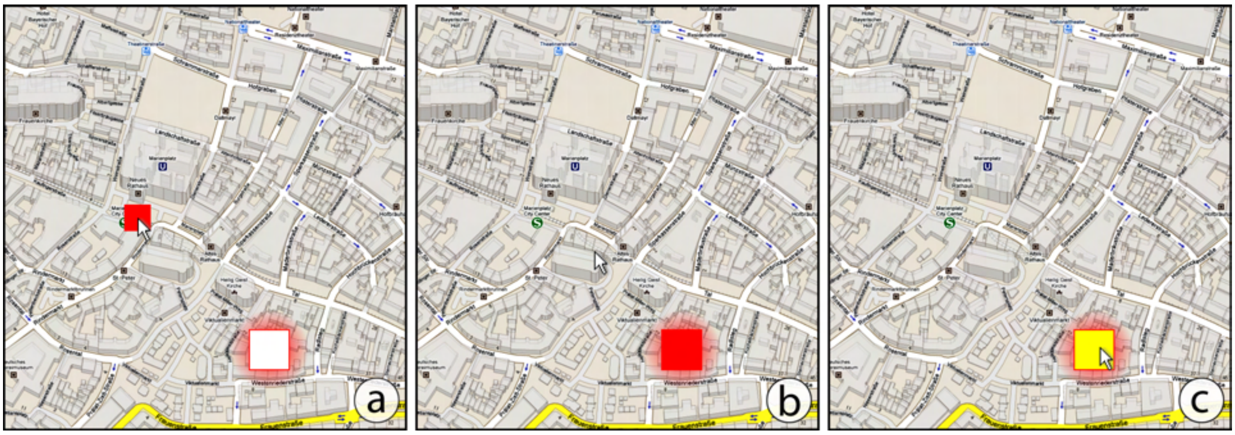


**Figure 4.6:** Procedure of a single trial: before the start of a trial the display shows both the start button and the target (a). After starting the trial, the start button disappears and the target is rendered in solid red (b). Hovering over the target causes it to turn yellow (c).

To identify potential strengths and weaknesses, we asked participants to acquire targets of three different sizes, two different distances (from the display's center), and eight different directions (i.e., the eight generic orientations). These parameters were chosen in a way that targets never reached the screen's boundaries to observe potential overshooting effects. The overall task has been modeled as a multidirectional tapping task [DKM99]. At the beginning of each trial, participants had to select a start button which was rendered as a solid red square in the display's center (see figure 4.6a) causing it to disappear (see figure 4.6b). Then they could move the remote pointer freely towards the target using the current technique. Visual feedback was provided when the pointer was inside the target's boundaries by turning the target into a yellow square (see figure 4.6c). The trial was completed, when the participant pressed the device's select button while the pointer was inside the target. Pressing the button while the pointer was outside the target increased the error count. In contrast to other Fitts' law experiments (cf. [FBB$^+$05]), participants had to press and release the button while the pointer was inside the target. We asked our participants to acquire the targets as fast as possible while maintaining a low error rate.

During the test we recorded both the task completion time (i.e., the time from pressing the start button until a successful target selection) as well as the error rate (i.e., the number of unsuccessful target selections). We further recorded the pointer's trace for each trial to identify possible overshooting effects. The results show that *Scroll* was – as expected – the overall slowest technique while *Move* was the fastest. For small target sizes *Tilt* performed worse than the other two techniques which may be explained by overshooting the target leading to increased movement times. For medium-sized and large targets (48 and 72 pixels respectively), *Move* had a mean selection time of 1401 ms (*Tilt*: 1868 ms) while *Scroll* had a significantly higher selection time of 2023 ms. The selection time improved by 31% for *Move* and 8% for *Tilt* respectively. The improvement of *Move* for small targets (24 pixels) was considerably lower (13% compared to *Scroll*). Surprisingly, *Tilt* was slower than *Scroll* and resulted in a loss of performance by 15%. The results are consistent for both close and distant targets (see figure 4.7a).

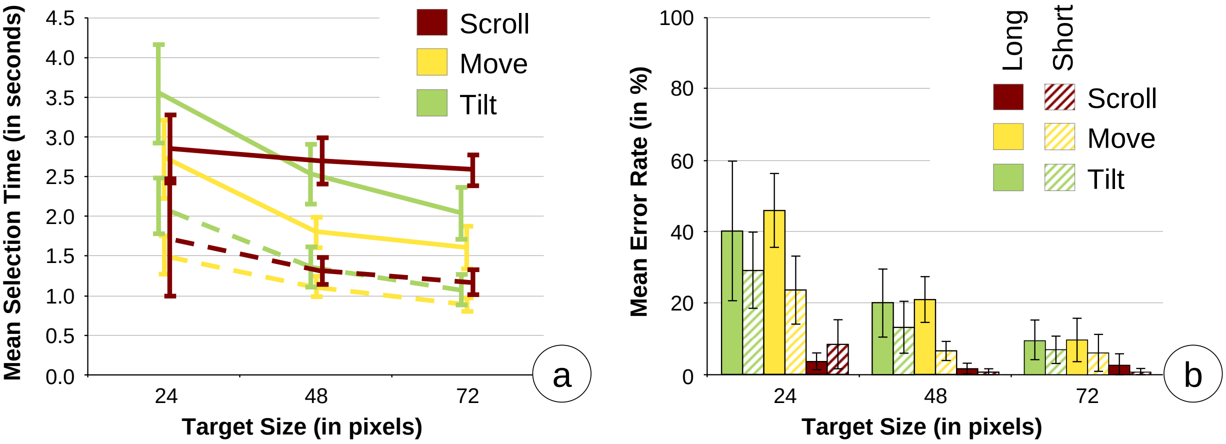


**Figure 4.7:** Evaluation of task times and error rates of relative pointing: a) shows the mean task times, b) denotes the average error rates. Error bars represent 95% confidence intervals.

As shown in figure 4.7b, Scroll was by far the least error prone technique. Especially for small- and medium-sized targets, *Move* and *Tilt* showed similar error rates (i.e., no significant difference) which were significantly higher compared to *Scroll*. The distance between the start button and the target also had an effect on the selection time for small- and medium-sized targets. It can be stated, that the closer a target is, the less error prone is the technique. This can again be explained with potential overshooting effects. While this holds true for *Move* and *Tilt*, the error rate of *Scroll* was consistent regardless of the target's size and distance. For large distances, *Move* had an error rate of 46% for small targets and 21% for medium targets respectively. For large targets the error rate decreased to 9%. *Tilt* showed nearly the same error rates. An explanation for the high error rate of smaller targets is due to slight phone movement while pressing and releasing the phone's select button. For short distances, *Move* had error rates of 23% for small targets (*Tilt*: 29%) and 6% for medium-sized and large targets (*Tilt*: 13% for medium-sized, 7% for large targets).

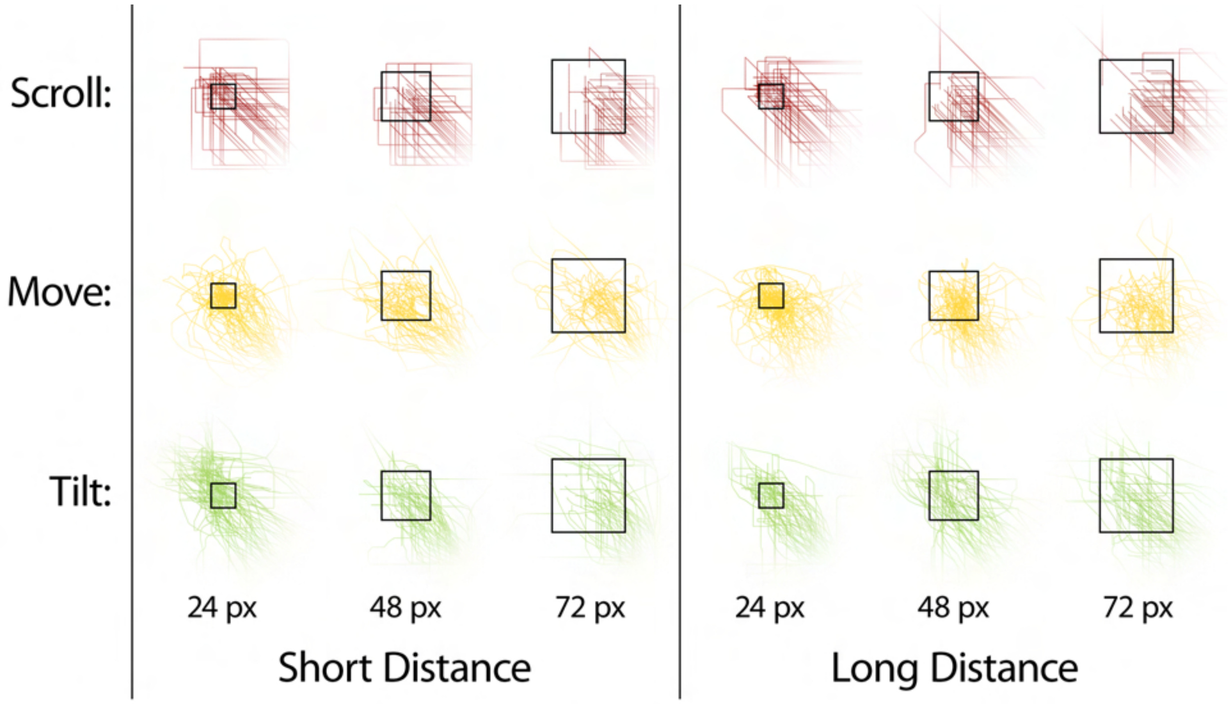


**Figure 4.8:** Traces of relative pointing: Left (right) shows traces for all participants for short (long) distances. The targets are placed diagonally at 225° (*direction*: NW).

The extremely high error rates in conjunction with high selection times for small targets of *Tilt* can be explained with the overshooting effect as shown in figure 4.8. During our study, we observed this effect numerous times. Participants moved the pointer towards the target and accelerated it (i.e., increasing the tilting angle of the device) to get there faster. However, the pointer eventually became too fast to control causing it to overshoot the target. Participants had to go back in order to correctly select the target and complete the trial. While this seems obvious for targets far away, the effect also occurred for short distances. Here we assume that participants were not able to smoothly accelerate the pointer causing a short but fast movement. As denoted in figure 4.7c, the overshooting effect occurs for all techniques but is significantly worse for *Tilt*.

### 4.2.3 Discussion

In our experiment we were able to identify strengths and weaknesses of three different techniques that allow *relative* and *indirect* pointing. In terms of selection time, *Move* and *Tilt* perform better especially for large targets compared to *Scroll*. For small and medium targets, both techniques introduce high error rates which need to be addressed. The error rate for *Tilt* resulted from the overshooting effect. This effect could be reduced by decreasing the pointer's movement speed at the expense of higher selection times. For *Move*, the error rates were introduced by the slight hand movement when pressing the select button. Participants pressed the button while the pointer resided within the target's boundaries but released it when the pointer had moved out. This can be addressed by turning off the pointer's motion while the button is pressed. However, this would not allow for dragging an item on the remote display. Another solution is to suspend the motion analysis for a single frame for both pressing and releasing the button. The rather low operation speed of *Scroll* could be improved by using a ballistic behavior or increasing the overall speed. The latter case would cause the pointer to jump numerous pixels per second which may increase the risk of an overshooting effect.

Remarkable was an observation of different postures during the study for both *Tilt* and *Move*. These postures lead to different ratings in terms of perceived fatigue for wrist and arm. While participants moved the whole arm when using *Move* in the beginning of the study, they sooner or later switched to turning the wrist only. This increased the error rate (due to the overshooting effect) but decreased perceived fatigue. For *Tilt*, several participants used both their hands to hold the device. This resulted in (1) a more stable control of the pointer and (2) less fatigue for shoulders and arms. Several other participants further pressed their arm towards their body in order to only move the lower arm. This resulted in less fatigue and also a more stable control as the degrees of freedom for the mobile device decreased. Overall we believe that *Tilt* introduces a certain "skill" factor and is thus suitable for gaming applications. We also needed to decide whether *Scroll* or *Move* will be used in our final application. As the items shown on the remote screen are large enough, we chose *Move* over *Scroll* due to its more flexible movement characteristics. In the next section, we apply *Move* as pointing mechanism to determine how content shown on the external display can be manipulated.

## 4.3 Manipulating Remote Content

Our solution of moving a pointer on the external display allows addressing and dragging content. Depending on the content other manipulation tasks may be needed such as rotating, scaling, or transferring it between the external display and the mobile device. Users need to be able to either switch between modes or use certain gestures for different options [KKM+06]. Both approaches have their limitations especially when the number of options increases: the first one requires the user to either press a button or choose the respective mode in a menu. In the latter approach, users have to learn and remember different gestures for each option. As the movement of the mobile device in our prototype is already used to control the pointer, it is hard to distinguish gestures from regular move events. Furthermore, the mobile device only has a limited set of keys which in turn would only allow for a restricted set of modes. We chose to use a menu-like approach as known from regular WIMP applications. The question then is where these menus and their items are located considering that multiple users may interact simultaneously.

In this section, we discuss four placements of such menus (i.e., tool palettes in our prototype). We thereby pay close attention to multi-user capabilities. First, we present our four placements, namely *shared*, *personal*, *temporal-personal*, and *ultra-private* (see section 4.3.1). We then present the results of a preliminary study we conducted (see section 4.3.2). Finally, we discuss the placement strategies and present their limitations (see section 4.3.3).

### 4.3.1 Personal versus Public Controls

In traditional WIMP applications, controls (we refer to these as tool palettes) are placed on predefined positions on the screen. In this control paradigm, users place their pointer on the control they want to use and select it. Subsequently, they move their pointer to the item they wish to apply the control to. The control item in the menu thereby stays highlighted allowing users to see the currently selected one. As a first approach, we chose to place one tool palette on a fixed position on the screen. All users can access the palette and select a tool. Having multiple users, however, does not allow for the aforementioned feedback solution. We decided to employ the users' mobile display to show the currently selected tool. This approach allows users to select individual tools through their personal pointers but eventually increases *macro attention shifts* [HOHS07]. Figure 4.9a denotes the approach.

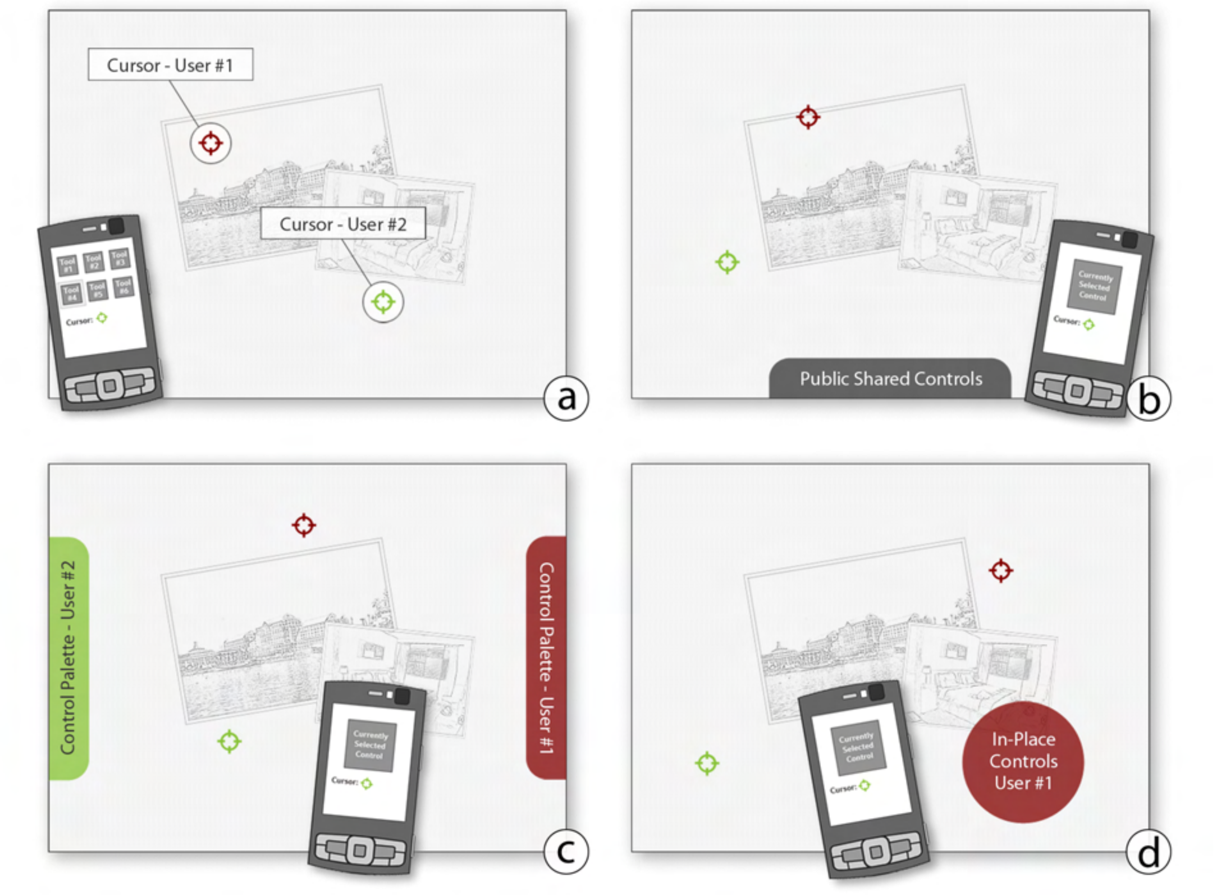

Cursor - User #1
Cursor - User #2
Tool #1
Tool #2
Tool #3
Tool #4
Tool #5
Tool #6
Cursor:
a
Currently Selected Control
Cursor:
Public Shared Controls
b
Control Palette - User #2
Control Palette - User #1
Currently Selected Control
Cursor:
c
Currently Selected Control
Cursor:
In-Place Controls User #1
d

**Figure 4.9:** The four different tool palette placements: (a) depicts the *ultra-private* controls entirely displayed on the mobile device. (b) shows the *shared* tool palette accessible by every user. (c) denotes the *personal* tool palettes displayed on the public display. (d) shows a recently "called" *temporal-personal* menu (here by user #1).

To allow for feedback on the external display, users can have their *personal* tool palettes. Similarly to *shared* controls, they remain fixed on the screen (e.g., the first user's one is on the left border, the second user's one on the right). The one-to-one relationship of users and their menus does not scale for a large group of people as the screen real-estate of the external display is limited as well. Furthermore, users still have to move the pointer to the control, select it and move it back to the content. This increases the overall task time causing longer interactions. On the other hand, the user's eye does not have to travel between the external display and the personal one as the selected tool is visible throughout the interaction. This visibility then does not allow for private interactions. Figure 4.9b shows the approach.

To overcome the travel distance of a user's pointer, we use *temporal-personal* menus. These menus stay hidden unless they are ”called” by their users. This approach is similar to the well-known context menus in modern operating systems. Once users call their menu, it pops up at the current pointer position. With this, the pointers are already in the menu area confining the movement to the tool selection. After a tool has been selected, the palette is hidden again. This type of controls does not waste any screen space in general but only when called for a short amount of time. However, the selected tool is again only visible on the mobile device's display causing aforementioned attention shifts. Furthermore, calling this menu may distract other users. This approach is depicted in figure 4.9c.

In all previously described placements, the mobile display shows the currently selected tool at all times (eventually causing attention shifts). In contrast to all others, an obvious placement is to put the menu directly on the mobile display in an *ultra-private* style. The currently selected tool is highlighted on the personal display. Furthermore, users can select a new tool using the directional keys in the two-dimensional menu layout. Naturally, this approach causes the highest amount of attention shifts as the users' eyes permanently move between external and personal display. Especially when eyes move back to the external screen, they have to refocus as well as locate the cursor. This approach is illustrated in figure 4.9d.

| **Technique** | **Travel Distance** | **Screen Waste** | **Eye Movement** |
|---|---|---|---|
| Private | None | None | Yes |
| Personal | Far | Permanent | No |
| Temporal-Personal | Near | Temporary | Partly |
| Shared | Far | Permanent | Partly |

**Figure 4.10:** Classification of control placement strategies. The shaded areas indicate potential shortcomings of a strategy. *Temporal-personal* appears to be the best trade-off.

After describing the placement strategies, we can identify four key aspects of such: first, the selection of a control is motivated by the user's need to accomplish a task as fast as possible. This is influenced by the pointer's travel time between content and control. Second, placing controls requires screen real-estate either temporarily or permanently. The more screen space is taken by the controls, the smaller becomes the actual interaction area. And third, distributing information across displays (here: personal and external screen) causes attention shifts which may increase the overall task time. Figure 4.10 summarizes these key aspects for each placement strategy. In theory, it appears that the *temporal-personal* one is the best trade-off.

### 4.3.2 Evaluation

In an experiment we compared the four placement strategies regarding task time. We further paid close attention to the attention shifts occurring during the task. Participants repeatedly had to select a tool and subsequently apply it to an image shown on the external display. For this, they got six different controls, namely *move*, *rotate*, and *scale* and furthermore *brightness*, *contrast*, and *saturation*. This section briefly describes the task, the procedure and the experiment's results.

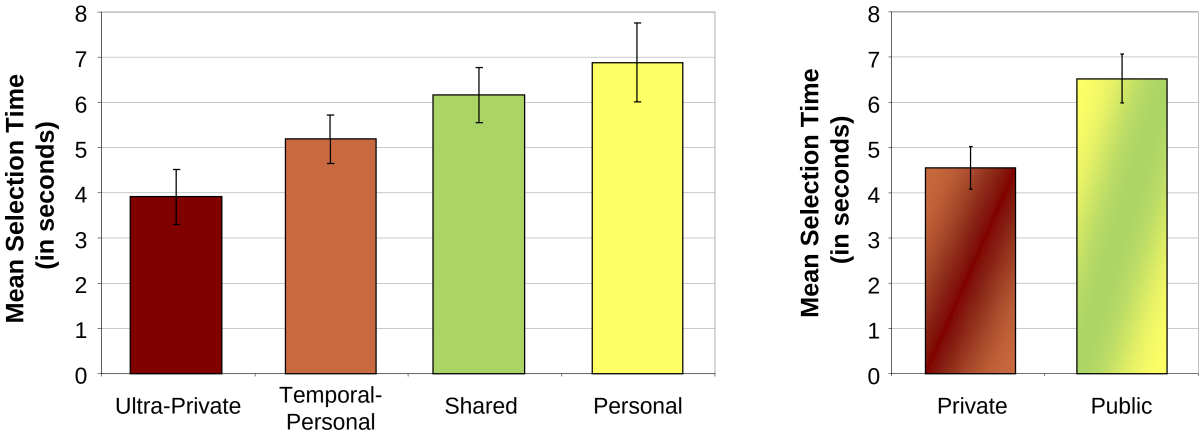


**Figure 4.11:** Evaluation of task times for different control placement strategies: left shows the mean selection times of each of the placements. Right shows an aggregation of *ultra-private* and *temporal-personal* (i.e., *private* controls) as well as *shared* and *personal* (i.e., *public* controls). The error bards represent 95% confidence intervals.

The participants of our study were standing in front of the same display used in the previous study regarding the pointer control. We recruited 12 participants (five female). These participants were not the same as the ones performing the pointing experiment. Their task was to select a tool using the current tool palette placement and apply it to an image located in the display's center. The tools to be selected were presented in random order. A restriction was that two subsequent trials did not have the same tool. The tool to be selected was shown on the external display and the pointer remained still in the display's center. To start a trial, users pressed the mobile device's select button. Now they were able to move the pointer on the external screen. Depending on the current palette, they either moved the pointer towards a palette and selected a tool, or selected the tool directly on the mobile device. We measured task time from the moment participants pressed the start button until they activated the tool on the image (i.e., a *down* event occurred on the image with the correct tool). In case they selected the wrong tool, they had to go back and select the correct one. Naturally, this heavily increased the overall task time. All participants received a short training period until they felt familiar with the interface and all its controls. As the system (and one of the placement strategies in particular) has been developed for multiple users, we always had two participants performing the task simultaneously.

We counterbalanced the presentation of the different tool palettes among our groups of participants. For each palette, participants had to perform 18 trials leading to an overall measurement of 72 data points per participant. As expected, the *shared* and the *personal* palette performed worst in terms of the overall task time due to their increased pointer travel distance. Both *shared* and *personal* palette did not differ significantly in task time. The *temporal-personal* naturally had a lower task completion time as it does not require long distances. There was a significant difference only compared to the *personal* palette. No significant difference could be found between *temporal-personal* and the shared palette. The *ultra-private* palette on the other hand outperforms all other palettes significantly (all $p < .001$). The higher performance of the *ultra-private* palette is surprising to a certain extent considering the enormous eye movements required for this palette. The results are summarized in figure 4.11.

In post questionnaires, participants rated the four palette types by their own preference in a 5-point Likert scale (1 equals ”I dislike the palette” and 5 equals ”I like the palette”). Ten of our participants preferred the private palettes (i.e., *ultra-private* and *temporal-personal*) over the others. In the subjective ratings, the *ultra-private* palette was the overall favored one, but only slightly higher than the *temporal-personal* one. The consistently low ranking with 2.9 for the other two palettes (i.e., *shared* and *personal*) indicates that cursor travel times are unwanted by our participants. When asked why they prefer the *ultra-private* palette in particular, they stated that the palette ”in-hand” is easily to learn and might be operated in an eyes-free way.

### 4.3.3 Discussion

Our experiment showed that private controls (shown on the mobile device) are both faster and perceived better by users compared to public ones. Numerous eye movements between the external display and the mobile device were not recognized as distracting as thought originally. Participants further stated that they were able to quickly locate their own pointer on the external screen again after switching to another tool. Somewhat surprising was the high task completion time of the *temporal-personal* palette compared to the *ultra-private* one. As both only require very short distances between the palette and the content, we thought that this palette would perform better. When asking the participants about the higher task time, they stated that for activating the menu they had to press a different button. There was a macro attention shift between the remote screen and the mobile keypad causing a delay. For the *ultra-private* palette, this attention shift was not as severe since users had their finger on the directional keys while holding the phone. Participants did not have to reposition their finger on the mobile device’s keypad.

Placing the tool palette on the mobile device has one major shortcoming. While the virtual resolutions increase on mobile devices, their screen real-estate does not to preserve their portability. The set of tools being visible at the same time is rather limited. One solution to this problem could be to have the tool palette as context-sensitive menu. Usually not all controls are applicable for the item the remote pointer currently resides on. These controls could be removed temporarily from the tool palette and added again once they can be applied to content. This approach is similar to context menus used in modern operating systems. However, this strategy may harm memorizing the interface for eyes-free control. Nevertheless, *ultra-private* controls seem to be a promising candidate for our prototype.

## 4.4 Summary and Discussion

In this chapter, we have investigated the three distinct phases of interaction, namely *connecting* to a display, *pointing* at content and *manipulating* it. These three stages have been implemented in a single application. At the same time we conducted experiments for the stages (except for the *connection* phase) and implemented the final system accordingly. Our system allows selecting and connecting to a display in the environment that is suitable for the given task. Subsequently, users can point on the external screen using the sensing capabilities of their mobile device in a *relative* and *indirect* way. After the content of interest has been addressed, the mobile device is for manipulation purposes such as translating, scaling, or rotating the content (see figure 4.12). The manipulation can be done by selecting the respective tool on the mobile device's display and using the device's movement for input. For example, rotating an image is realized by first choosing the rotate option, placing the pointer on the image and then pressing and holding the device's select button. When the button is pressed, the pointer remains still while the movement of the device controls the rotation. Releasing the button allows the pointer control again.

In experimental evaluations we determined (1) the best suitable technique for controlling a pointer on the remote display and (2) an adequate placement for each user's tool palette. For the remote pointer control we found that mapping the device's motion linearly (using the device's built-in camera) to the pointer's movement works well when the targets are large enough (i.e., larger than 48 pixels). We also found that this technique introduced several errors for small targets. For the manipulation task we chose larger targets to allow for minimal error rates while maintaining a short task times. In the manipulation task we investigated the placement of control sets (i.e., tool palettes) to allow for switching between different tools. Personal palettes that require little or no screen space on the external display were overall favored by the participants and further resulted in short task times. The low times are explainable by short (or even no) necessary pointer movement. Regarding task completion times and subjective ratings of our participants, placing the tool palette on the mobile device's screen was favored. The small screen does not allow for a large set of tools. This can be overcome by using context-sensitive items (i.e., only showing tools that are applicable).

During our studies we discovered two major issues which we think are related to *relative* and/or *indirect* pointing: first, each user has to have an individual pointer shown on the external screen which (1) allows others to follow the user's actions and (2) covers a (small) region on the remote display. The more users interact simultaneously, the more pointers are visible decreasing the effective screen real-estate (i.e., the space that can be used for content). And second, connecting to a display in a *modal* fashion may harm cross-display operations (i.e., transferring content from one display to another) in certain cases. In the following we discuss both issues in more detail and suggest a novel interaction model for distant displays.

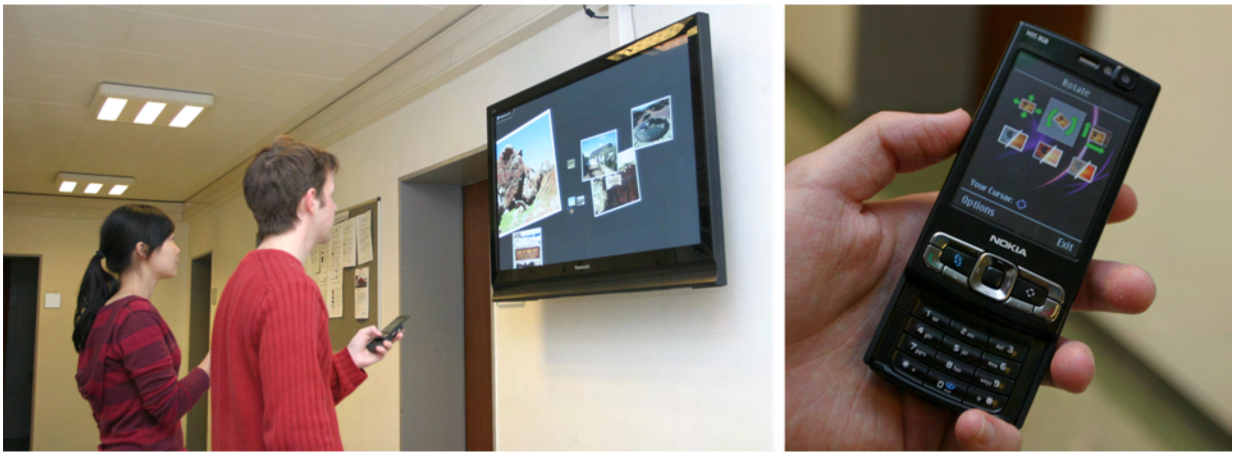


**Figure 4.12:** Two users interacting on the public display: left shows both users controlling their pointers individually. Right denotes the visual interface presented on the mobile device.

## 4.4.1 Indirect Pointing with Multiple Users

Our prototype employs the mobile device as *relative* and *indirect* pointing device which behaves similarly to a computer's mouse. However, this limits the number of users that can interact at the same time. As each user has a corresponding pointer on the remote display, they need to able to locate and track their own one. A solution to this problem is to use color-coding which assigns a unique color to each of the pointers. Only a small number of colors are unambiguously distinguishable at a time. Furthermore, color-blind people can only distinguish colors based on their brightness. The number of users still remains restricted. Another option is to use shapes instead of colors. While this may allow for a broader set of possible cursors, the problem of locating and tracking still increases with the number of users. In traditional interfaces, identifying the pointer's position is often done by moving the physical device rapidly back and forth within a small radius. The pointer can thereby be located as the only moving object on the screen. The increasing number of users in turn will limit the success of such a technique since the likelihood of two users performing the action at the same time increases.

Besides the existing problem of locating and tracking a pointer, each cursor further occludes a small portion of the display. A critical mass of people would occlude enough screen space to turn it unusable for all users. This number can be determined by the required size of a pointer and the provided display space. The display space suggests a certain distance to it in order to fully view the entire canvas. When the distance to the display increases, the physical size of pointers needs to be enlarged as well to still allow for their visibility. As the viewing distance depends on the screen's physical size and the pointer's size depends on the viewing distance to the display we can assume that the pointer's size increases with the display's size. A pointer always occludes roughly the same portion of the screen. In modern operating systems, pointers use approximately 0.21" (screen properties: 96 dots per inch, 17" diagonal), which is about 0.03% of the screen space. The field of view a pointer needs to cover (with a 34" distance to the display is 0.4°). Considering ultra-large displays (i.e., mounted in a stadium) the maximum viewing distance can be up to several hundreds of yards. The pointer then covers a much larger portion of the display.

Users may need controls to manipulate the shown content depending on the application. Giving each user individual controls on the external screen further limits the number of users. In our prototype, the *personal* controls on the remote display are tied to the display's borders. This placement only allows four users at a time. While a *shared* tool palette overcomes this limitation, it introduces another as this menu can get overcrowded. With this, the aforementioned problems are given within a much smaller area and are thus becoming worse. To only occlude screen space temporarily, users could call a context-like menu at the very position their cursor resides. This solution again may not scale as the menu occludes a constant portion of the display which in turn influences on the interaction of others. Although increasing eye movements between the personal and external display, placing tools on the users' mobile device seems to be the only solution that scales even for a large number of people. Local controls also allow for more private interactions.

### 4.4.2 Cross-Display Interactions

Interacting on a single display, the aforementioned interaction phases (i.e., *acquire display*, *acquire content*, and *manipulate content*) with small extensions for tool selection are applicable. Interacting across displays, however, requires the pointer to move in a logical way from the user's point of view. For example, if users move a pointer from one display to its right neighboring screen, the underlying infrastructure needs to know this configuration as well in order to allow for a correct transition. Furthermore, two displays that are adjacent logically but not physically (e.g., two screens with a two yard gap between them) either cause the pointer to jump to the next display or result in an invisible pointer for a certain amount of time. In any case, both approaches need to know the display arrangement (and possible the displays' physical properties) a priori. This then limits the environment's flexibility in two ways: first, mobile devices need to be tracked permanently if they are included in such cross-display operations. And second, the existing three-dimensional representation of the environment needs to be changed whenever displays are added or removed. If such steps have been realized, the perspective of users still remains an open issue unless they are tracked [NSC$^+$06].

To overcome these restrictions and allow for more flexible setups, mobile devices can act as temporary storage containers. This would extend the aforementioned model of interaction phases: first, users acquire the originating display. Second, they point at the content to be transferred and store it on their mobile device. Third, users disconnect from the current external display and connect to the destination screen. Fourth, they point at the location they want to place the content at. And fifth, users then initiate the transfer of the temporarily stored content back to

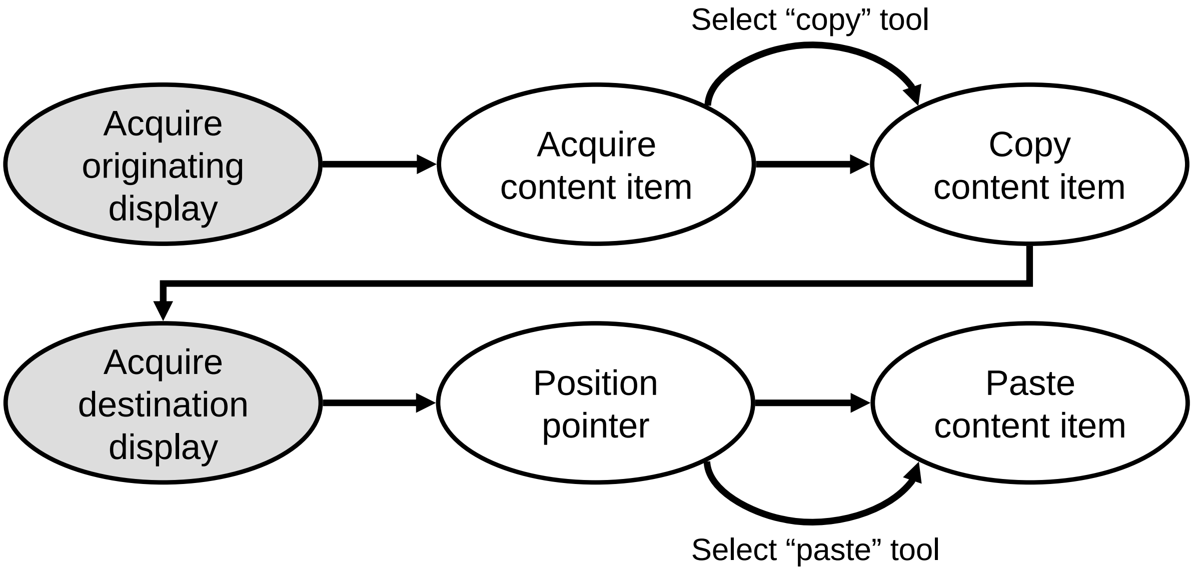


**Figure 4.13:** Interaction phases for cross-display operations.

the external display. This solution is not sufficient as it requires several steps including two connection procedures which take several seconds. Figure 4.13 is extends the original phases to allow for cross-display interactions. This also means that mobile devices such as tablet PCs that are joining the environment temporarily also need to be discoverable. The spatial availability then is questionable as these devices may move causing a change of their distance to the user more often. Our presented visualization would turn into a dynamically changing screen.

### 4.4.3 Absolute Pointing as Promising Alternative

Two characteristics of displays – namely the *number* of users as well as the *flexibility* of the environment – seemed to be limited by relative and indirect pointing. As pointers on the remote display consume a certain amount of screen space, the number of users is restricted depending on how much screen space is still needed for the application to be usable. On the other hand, the flexibility of the environment in cross-display operations can only be increased if either the environment is known a priori or users disconnect from one screen and reconnect to another. Besides these factors, bystanders are aware of the user’s actions since a personal pointer is always shown on the remote display. However, this issue is beyond the scope of this thesis. Although the uniform approach (i.e., having the same type of input on various displays) is already ensured by *relative* and *indirect* pointing, it appears that some of the limitations cannot be overcome by this type of interaction. In the following we discuss whether *absolute* pointing (as an input alternative) can overcome the aforementioned limitations.

In contrast to *relative* pointing that almost necessarily is in an *indirect* fashion, *absolute* pointing can be performed either in a *direct* or in an *indirect* way. The *indirect* approach still requires a pointer to be shown on the external display causing similar issues compared to relative pointing.

Direct interaction on the other hand requires that the user's action and the system's response occur at the same location. One prominent example for such input is a direct touch interface. As stated before, public displays either are not always reachable directly or users do not want to touch them directly due to security (e.g., the visibility of the interaction in public) or hygienic reasons (e.g., a layer of grease on the display). The solution needs to incorporate the personal device as users trust them at all times. In chapter 3 we already showed how the personal device can be used to interact with remote content in a *direct* and *absolute* way. If the display is unreachable, however, the presented approach needs to be changed. One solution is to transfer Tani's idea to mobile devices [TYT$^+$92]. The local video shown on the mobile device then turns the mobile device into a *see-through* interface [WFB$^+$07].

*Absolute* pointing further requires users to point towards the display they intend to interact with. By doing so, they implicitly identify the target display. Combined with the use of the mobile device's video, this approach can be used to connect to an external display by analyzing the visual content. In this case, the aforementioned interaction phases can be reduced to *acquiring* content and *manipulating* content leaving out the *connection* phase. From the user's point of view the phase of acquiring the screen will hence become a part of the content acquisition. This would further improve cross-display interactions: moving the pointer from one display to another now works without the need of disconnecting from one display and reconnecting to the other one. In the remainder of this work, we present an approach and its architecture allowing for such *absolute* and *direct* interactions on external displays still maintaining the same input on all screens.

# III

# Through the Display Interaction

Chapter 5

# *Shoot & Copy*:

## Discrete Pointing through Live Video

*Discovery consists of seeing what everybody has seen and thinking what nobody has thought.*

**– Albert Szent-Györgyi de Nagyrápolt –**

As shown in the previous chapter, relative and indirect pointing allows for interaction on large and possibly distant displays. However, such techniques are limited due to personal pointers shown on the external screen. First and foremost, users have to explicitly select the display they want to interact with a priori. With an absolute approach, on the other hand, users implicitly select their target display. For example, pointing on content on an external display using an indirect device such as a laser pointer immediately reveals the target display as well. These indirect approaches still show a personal pointer on the external screen leading to the problems identified in chapter 4. In this chapter we[1] present *Shoot & Copy*, a system that allows pointing at external displays in an absolute and direct fashion. At the same time, users aim their mobile device at the desired content which is then shown in their mobile device's viewfinder. Upon content selection (i.e., the item underneath a crosshair), the mobile device takes a picture which is then analyzed by a centralized component within the environment to identify the targeted content. This means that from the user's point of view, the connection process is embedded into the selection phase. The mobile device maintains a permanent connection to this centralized instance which is established automatically during the application launch. This setup further allows the selection of content

[1] The work presented in this section has been published in a scientific paper [BAB$^{+}$07]. The scientific plural refers to all authors of these publications – namely Sebastian Boring, Manuela Altendorfer, Gregor Broll, Otmar Hilliges, and Andreas Butz.

independently of the target display without the need for users to connect to it in a *modal* way. In a qualitative evaluation we found that users liked the idea of selecting content by taking a picture although the overall selection time (i.e., the time between taking the picture and actually receiving the content) in our implementation was rather long.

The remainder of this chapter is structured as follows: We start by motivating the approach of content selection by taking images (see section 5.1). Subsequently, we describe the necessary steps of computer vision to identify an external display purely based on the mobile device's viewfinder content (see section 5.2). Based on this architecture, we present a proof-of-concept implementation and its performance and qualitative evaluation (see section 5.3). We then extend the infrastructure to allow the detection of multiple independent displays (see section 5.4). We conclude this chapter with a summary and a discussion (see section 5.5).

## 5.1 Motivation

Large, high-resolution displays are found in our everyday life (e.g., in subway stations, shopping malls, or airports) showing commercials, late-breaking news, and weather forecasts. Since these screens are passive broadcasters without any interaction possibilities, potential users have to memorize the shown information until they have a chance to review it (e.g., on their computer at home). Users may forget the shown information as they are flooded with such throughout their day. Accessing information immediately through a mobile device, naturally also works but still requires the user to re-enter the displayed information (e.g., a URL) on the mobile device. While this approach reduces the risk of forgetting information of interest, it also increases the interaction time. Having a *modal* connection procedure (as described in chapter 4) would further not allow such spontaneous and short interactions. A simpler approach needs to be found that allows for *non-modal* connection mechanisms (i.e., transparent to users) while people select the content located on an external display with their personal device.

When we see information of interest in our surrounding, there are several ways of increasing and ensuring that we memorize the information. One obvious solution is to take pen and paper and write down important parts (e.g., product name, or a URL to a store). The note taken on paper, however, needs to be transported back into the digital world if users want to seek for additional information. Furthermore, the problem of memorizing the information is now shifted to memorizing that one has written down something. As modern mobile devices usually have high-resolution cameras built-in, information seen on a poster, placard or even digital display can be captured by taking a picture of it. Although the data is now available in a digital format, it is still encoded within the image which in turn is not necessarily machine-readable. Respective owners of these pictures know what kind of information they convey. While being digital, the information needs to be translated from one format (e.g., a URL shown in the image) to another (e.g., a URL typed on a keyboard). The time from capturing the information until viewing additional content may still increase the risk that users forget the content.

As the taken picture is in digital form, one can think that this procedure allows for retrieving additional information as well. For example, taking a photograph of a poster advertising a new music album could be translated to sending short samples of each song. To retrieve information, a connection needs to exist between the mobile device and the external display (or an associated service). Information can be further manipulated on the mobile device once it has been brought there. This would also manipulate the content on the remote screen at the same time. In any case, however, the taken picture needs to be analyzed first to identify the content that has been captured. Taking all items in the viewfinder into account, the captured region of an external display can be identified. Thus, the technology for communicating information via visual patterns blends in with the media that display them – the screen becomes the marker.

## 5.2 Visual Recognition of an External Display

To allow the aforementioned interaction, the target display needs to be identified by the content shown in the mobile device's viewfinder and the taken picture respectively. This procedure can be broken down into three stages: first, content items need to be separated from the background in the picture. Second, the item in the picture's center needs to be identified according its visual representation as well as the geometrical layout of neighboring items. And third, local input needs to be transformed correctly to select the correct item. In this section, we present the theoretical foundation of these steps. We further present the necessary procedure using computer vision. We thereby assume that users take a picture with their mobile device of a nearby display. The picture does not need to include the entire display. The content of interest, on the other hand, has to be in the picture's center (i.e., underneath a shown crosshair) similar to [BRS05].

### 5.2.1 Detection of Content in the Viewfinder

Before any matching can take place, the content needs to be identified and separated from the background. We assume that there is a high contrast between the background and the foreground layer. This difference allows for better separation of them by the human eye and allows for a better recognition [AHH$^+$02]. Similarly to the visual recognition process of the human eye, we use this contrast to separate the content from the background. As pictures taken by mobile devices usually have a lower quality compared to high-end digital cameras, the image needs to be preprocessed slightly to allow for a maximum recognition rate. The first step is to reduce noise and further increase the contrast of the image which later allows a clear identification of the content's boundaries. The increased contrast then allows separating the content from any background using a binary filter (i.e., black if the brightness of a pixel is under a given value, white otherwise). The image now only contains black (i.e., background) and white (i.e., content) regions. As we are only interested in the content and not in the background at all, we can dismiss the background by finding connected components in the image. Each connected component thereby describes a rectangle in which the item is in (i.e., the bounding box), and contains the binary sub-image. These so-called *blobs* can be further filtered in two steps: first, blobs that are either too big (e.g., the screen in its entirety) or too small (e.g., remaining noise in the picture) are deleted. Second, blobs that entirely lie within another correct blob are also removed for ambiguity reasons. The remaining blobs now need to be matched to existing content on the external display. Figure 5.1 summarizes the image preprocessing.

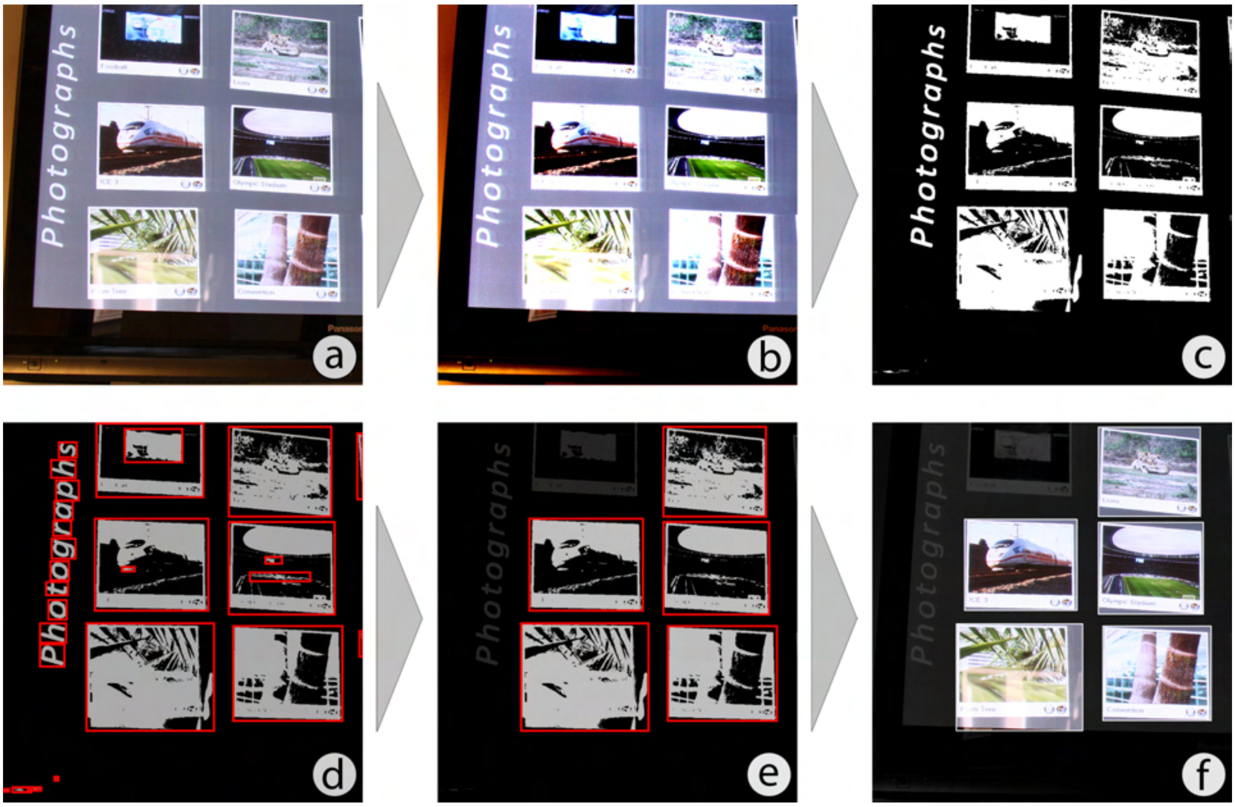


**Figure 5.1:** Preprocessing of pictures taken by a mobile device: the contrast of the original image (a) is first increased (b) for better contour detection. Thresholding the image allows removing the background (c). A blob detection finds connected components (d). These blobs are then filtered by removing too small (or too large) blobs as well as dismissing blobs that are placed entirely inside others (e). The remaining blobs are then used to ”cut” the regions of the original image (f).

If users hold the device exactly in front of the display without any rotation (i.e., the horizontal axes of both displays match), these blobs already give a good match between the taken picture and the external screen. This restriction is nearly impossible if we want to allow multiple users interacting on an external display simultaneously. Multiple users lead to arbitrary angles and distances between mobile device and target display as they have at least slightly different viewing positions. The correct transformation between the mobile device's image plane and the external display's image plane needs to be calculated. Assuming that we have rectangular content, we first need to identify the corner points of each detected blob's image. We chose to use the *Hough Transformation* to first detect the lines within the blob's image [DH72]. The detected lines can then be intersected to identify the corner points. The processing time of the *Hough Transformation* depends on the number of pixels that may be on a line (here: each white pixel in the image). As we are only interested in the outline, we can dismiss all pixels inside the outline (i.e., turn them black). This is done by leaving the first and the last detected white pixel but turning all pixels in between black for both rows and columns. Now, the *Hough Transform* only tests pixels that are already describing the outline of each blob and results in four lines described by their distance to the image's center as well as the angle of the line's normal. To find the corner points of each blob, the four detected lines are intersected. Figure 5.2 summarizes the process.

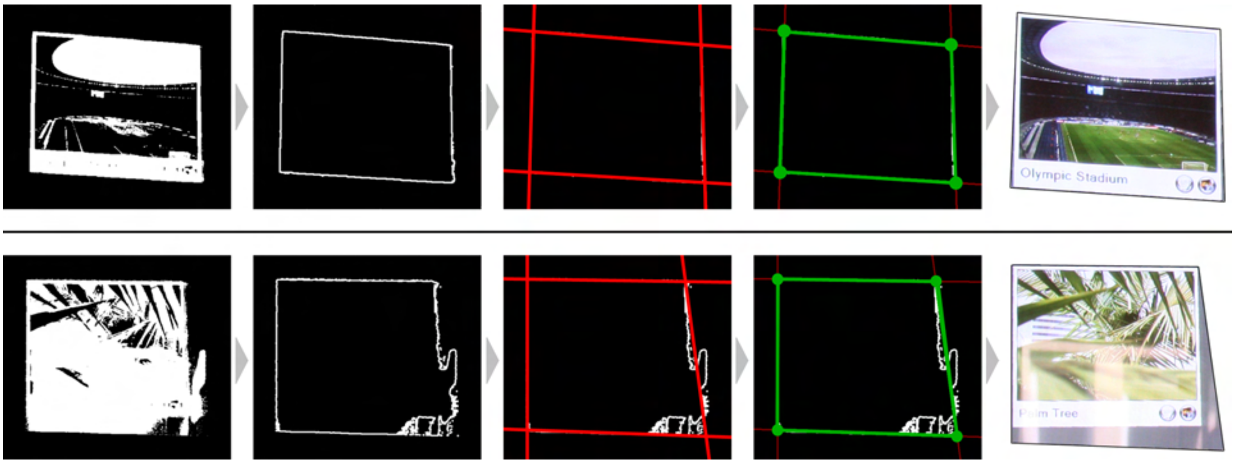


**Figure 5.2:** Detecting polygons in the captured image for a good (*top*) and bad (*bottom*) blob. From left to right: The contour of the detected blob is first calculated. A hough transform calculates lines (denoted in red). Intersecting these lines leads to a polygon (depicted in green). The polygon is used to "cut" the respective portion out of the taken image.

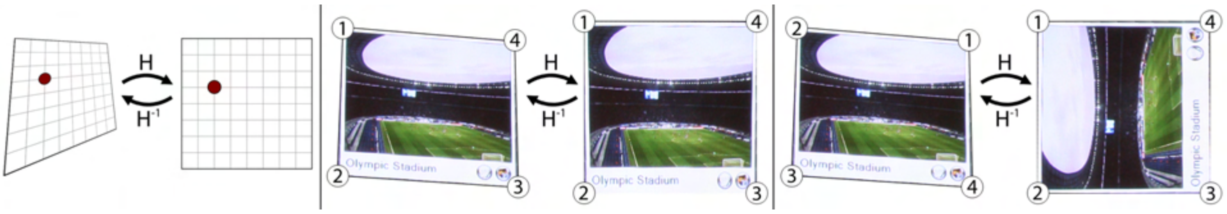


**Figure 5.3:** Rectification of distorted content in the taken picture. Left shows the principle of the *Direct Linear Transformation*. The transformation matrix $H$ is invertible. Center shows the rectification with a correct point matching. Right shows the same process if the points were detected in a different order.

The four detected corner points are used to create a polygon for each blob. This polygon can now be used to counteract the perspective distortion for the visual matching described in the next section. To do so, we apply the *Direct Linear Transform* to rectify the distorted image [AAK71]. Such methods are recently been used in the hardware of digital cameras (e.g., *Casio Exilim H10*[2])

[2] *Casio Exilim H10*: http://www.casio-europe.com/de/exilim/exilimhizoom/exh10/

as well. This procedure allows the transformation of a perspectively distorted rectangle into a two-dimensional one. As the content shown on the remote display is given in two-dimensions (in our scenario), applying this method to all recognized blobs allows a sufficient image matching of two rectified images with the same size. The size of the actual content is still unknown as no matching took place so far. Both the remote content as well as distorted images need to be sized equally in order to match them pixel-wise. Choosing the size of the predefined rectangle is a trade-off between the details shown in an image (i.e., the higher the size, the more details are shown) and the processing time (i.e., the smaller the size, the faster is the recognition process). As users are allowed to hold the device in arbitrary angles with respect to the display's horizontal edge, the detected corner points are possibly not in the correct order. This means, that the first corner point is not necessarily the top-left corner point of the content. Hence, three further rotated images (i.e., 90°, 180°, and 270°) of the rectified image are created. The entire process of rectifying the image and prepare it for image comparison is denoted in figure 5.3.

## 5.2.2 Visual and Geometrical Content Matching

After preprocessing the captured image and filtering possible content items, the remote content and detected blobs (and their rectified images respectively) can be matched against each other. The item with its blob closest to the image's center is used as *reference item*. The four rotated images are then compared against all images that represent content on the external display using simple image differencing (see figure 5.4). The resulting images are further processed with the aforementioned binary filter to state a *matching quality*. This value describes the likelihood that two images match (i.e., the ratio of correct pixels to all pixels). If this value is higher than a certain threshold (we chose 75%), the image, its corresponding rotation (i.e., 0°, 90°, 180°, or 270°), the matching quality and the corresponding remote item are stored. After matching the reference item with all possible target items (i.e., all items on the remote display), the results are sorted by their quality in descending order.

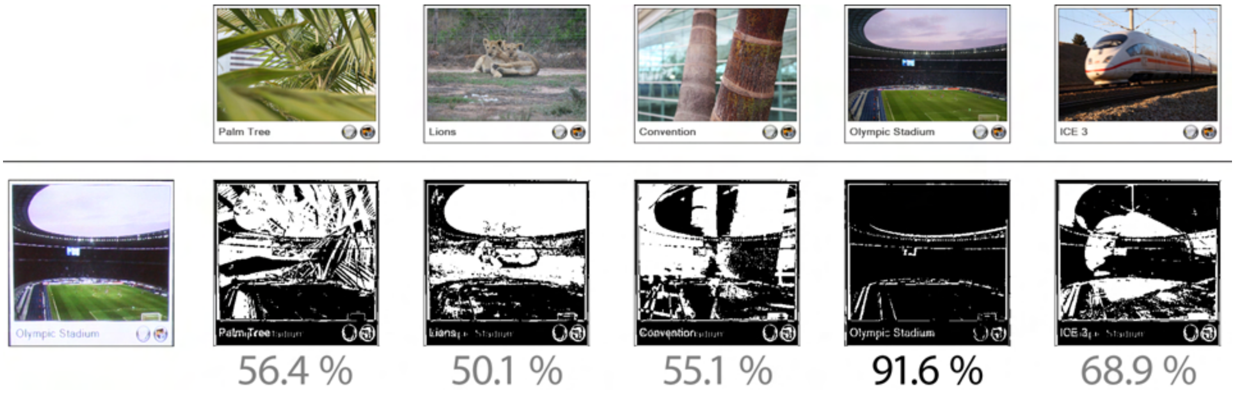


**Figure 5.4:** Similarity based on image differencing. While all images have significantly lower similarity (i.e., between 50 and 70%), the picture ”Olympic Stadium” matches well with over 90%. This image is therefore the candidate for further calculations.

The properties of content items include both visual (i.e., the graphical representation) as well as geometrical values (i.e., location, angle, and size of the item). Hence, if a match has been found, other items and their polygons detected in the captured image can be matched as well. Only if all items match, the *reference item* resulted in a correct match. As for previous steps, the perspective distortion needs to be equalized first. To do so, we first calculate the homography (i.e., the transformation matrix) between the corner points found in the captured image and the corner points of the matching item on the external display again using the *Direct Linear Transform*. If the item has been detected and matched perfectly, the transformation matrix is the exact same for every other polygon (and for the entire display). As cameras of mobile devices usually come with wide-angle lenses, this is only a theoretical case. Hence, the matrix is only an approximation which requires subsequent steps to allow for small error deltas. Each of the detected polygons (i.e., their corner points) except the one of the *reference item* is now multiplied with the calculated matrix. By doing so, we obtain their coordinates with respect to the coordinate system on the external display (see figure 5.5).

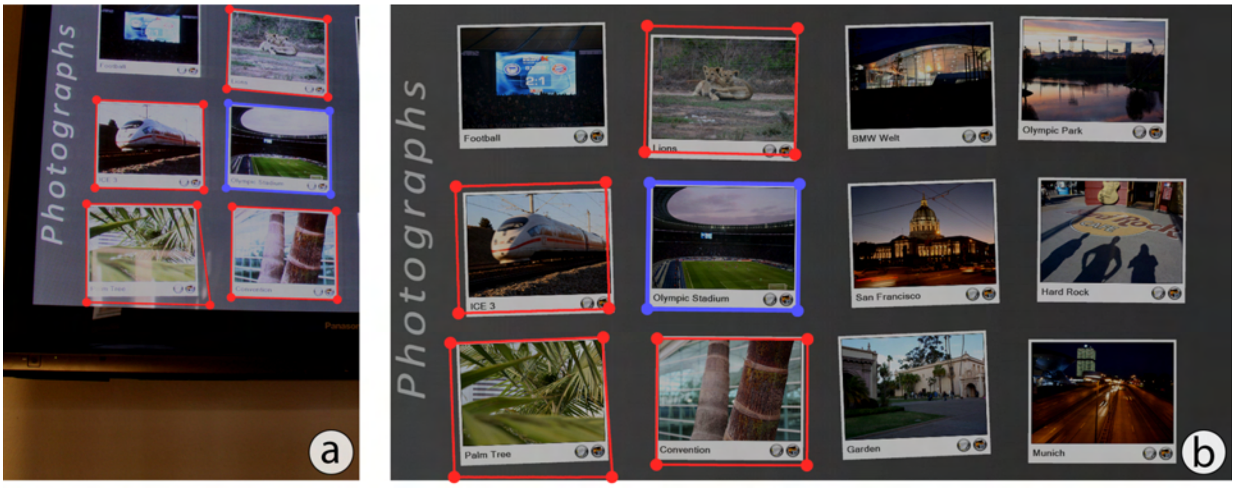


**Figure 5.5:** Matching the location of detected items (red) relative to the reference item (blue). Left shows the taken image and right shows the predicted positions based on the reference item. Although the lower right item is off by several pixel, the system can still match it according to its position. However, the detection shows errors which does not allow for taking the reference item to compute the final transformation matrix.

The polygons calculated for each item with respect to the coordinate system of the target display can now be compared to existing polygons. As mentioned before, we need to allow small errors due to the imperfect matrix calculation. If a polygon is found that does not match its predicted position closely, the process is aborted and the next *reference item* found in the image differencing process is chosen. The criteria for matching an item is the overall Euclidean distance of the polygons' centers as well as the Euclidean distances between each of the corresponding corner points. Only if each of these distances is below a certain threshold (here we chose 10 pixels), the polygons are a correct match. Otherwise, the polygons could not be matched. If all polygons could be matched successfully, the region of the target display has been identified successfully.

This type of matching is scale and rotation invariant which in turn allows a flexible and robust matching. The calculated homography can also be used to transform local input into the display's coordinate system. Figure 5.5 denotes the matching process.

### 5.2.3 Transformation between Mobile Device and Target Display

Now that the external display and the targeted area has been identified, the local input (e.g., for selection purposes) from the mobile device needs to be transferred into the target screen's coordinate system as well. As mentioned before, the calculated matrix between the *reference item* and its corresponding item on the external screen can be used. The detected matrix leads to inaccuracies due to the previously described detection process (see figure 5.5). As the matrix ideally describes the transformation from a pixel in the mobile device's image into the display's coordinate space, the inaccuracy increases if the reference item is rather small compared to the display's boundaries. We need to calculate a more accurate matrix if the technique should allow for precise positioning on the target display. The best matrix for this problem can be obtained by calculating the homography between the display's corner points in the captured image and the original corner points. However, this would require the user to always have the entire display in showing in the viewfinder, which is a limiting factor. Potential lens distortions of the mobile device's camera (which increase at the picture's corners) will further reduce the accuracy again.

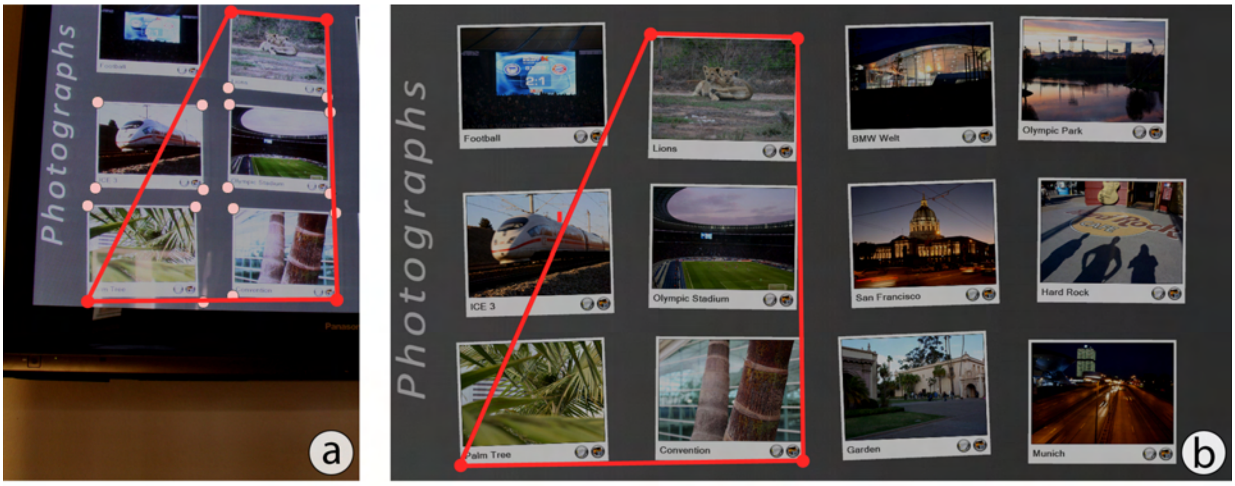


**Figure 5.6:** Optimizing the transformation matrix between mobile device and target display. Left shows the taken image and the four points producing the largest quadrangle. Right shows the four points on the target display. With this point set we achieve higher accuracy.

As taking the entire screen as reference item would limit the envisioned interaction and its flexibility, we chose to approximate the correct transformation matrix. A good approximation, on the other hand, is to use a set of four points that describes the largest quadrangle. Each of the detected polygons already gives four points. The number of detected items multiplied by four leads to the overall number of points. We can obtain the largest quadrangle then by finding two points with the largest Euclidean distance. Repeating this step with the detected points removed gives another point tuple. These four points then form the largest possible quadrangle in the given point cloud (see figure 5.6). For each point we already know the corresponding point on the external display. We obtained these in the previous matching steps when the geometrical positions were analyzed. These point 4-tuples are treated as virtual element spanning a potentially large region on the display. Thus, calculating the homography based on these points gives the matrix closest to the actual transformation (i.e., calculated by the four corner points of the target display).

The inverse of the calculated matrix can now be used to determine the target display's boundaries in the mobile device's image. We can transform input occurred on the mobile device into the target display's coordinate system. We take the moment when the picture was taken also as the moment when input occurred. In our scenario, input is always given in the center of the mobile display denoted by a crosshair overlaid on the video image. Taking a picture further signals a selection process. For content selection, the process described before can be stopped when the correct *reference item* has been found (and the image's center lies within its blob). Increasing the accuracy of the transformation matrix then is unnecessary. Merely being inside the item of interest is enough. Selecting a location for pasting an item needs accurate transformations as this usually is a more fine-grained operation.

## 5.3 A Proof-of-Concept Implementation

To validate the aforementioned theory and gain insights about processing times as well as user acceptance, we implemented a proof-of-concept prototype based on a camera-equipped mobile device and a large display. Our prototype allows users to capture a picture of a display's content which then results in options being shown on the mobile device's display for further for manipulation. We implemented two scenarios: in the first scenario (*selection*), the public display acted as an advertising wall showing recent movie and music covers. Users could select these and listen to 30 second samples of each track on the selected album (and a trailer for movie covers respectively). In the second scenario (*manipulation*), the public display was showing images from various users similar to the *CityWall* project [PKS$^+$08]. Users could select a picture and were able to store the image or write a comment. In this section, we describe the design in detail and present our applications and their implementation. We present a performance evaluation and conclude with a discussion about the prototype and its potential extensions.

### 5.3.1 Content Selection through the Viewfinder

In both of our scenarios, we wanted to enable users to simply take a picture of the content of interest without requiring a separate connection mode. On startup, the device scanned the environment for existing Bluetooth services. Once it discovered the correct one (described by a UUID[3] and

[3] UUID stands for *universally unique identifier* and is described in: http://tools.ietf.org/html/rfc4122

a name), it connected to the display. This step is part of the application launch procedure and is not recognized by users. From the technical point of view, the connection is established before any content is selected. From the user's point of view, this procedure is transparent. While this type of connection is not suitable for multiple displays it still allows for testing the overall idea of selecting content *through the display*. In the next section, we will describe, how we apply this approach to multiple displays in the environment.

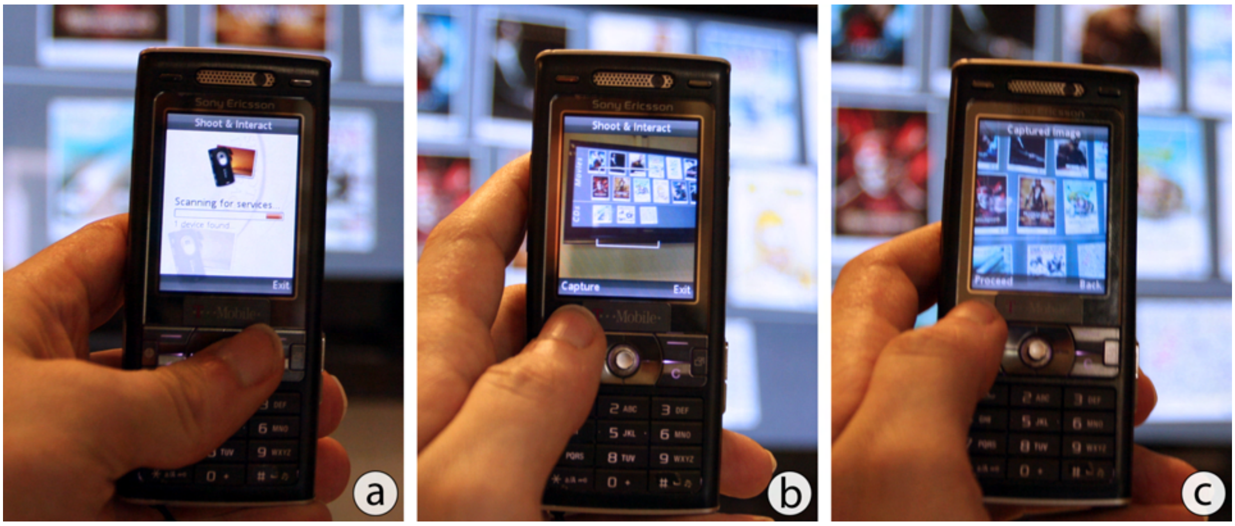


**Figure 5.7:** Connecting and selecting content *through the display*: While the application launches, the system connects to the display already with a specialized service name (a). Then the camera preview is shown which allows users to aim at content of interest (b). After capturing an image, users can decide whether this image has the content in its center (c).

Once the device is connected, a live camera preview is shown. Users can now aim at the content they wish to select. When users are standing further away from the display, the wide-angle lenses built into mobile devices causes multiple content items to be shown simultaneously. While this actually is better for the aforementioned image processing due to several neighboring items, it is more complicated for users to know which item is the one being selected. To facilitate this, the live video is overlaid with a crosshair to allow pixel-exact selection regarding the mobile device's display. Now they can use their mobile device as *absolute* and *direct* pointing device to select content on the display (i.e., the item underneath the crosshair is being selected). If they now press the "select" button, the mobile device takes a picture with a higher resolution with more detail than in the live camera preview for easier image processing. The taken image is then sent immediately to the computer driving the display which initiates the aforementioned content analysis. During this time, the mobile device displays a wait screen letting the user know that the system is gathering the selected content.

If the system was not able to detect the selected content, users receive an error message asking them to select the content again in the described fashion. If the content has been detected, the system sends back a list of options describing which actions users can perform on this item. The list of actions is described for each content item on the computer running the display. In this way, the mobile device acts as a thin client only displaying information gathered from the public screen. As shown in figure 5.7, a thumbnail of the selected item is shown on top of the list. This preview allows users to determine whether the system actually selected the correct content. If the item was the wrong one, they can go back and select it again. The list itself has two meanings: first, users see the possible actions they can use for this item. And second, users can immediately select one of these options. To perform one of these actions, the mobile device only needs to present its generic user interfaces (i.e., writing text, showing an image, and playing an audio or video file). If more complex user interfaces are required (e.g., a drawing canvas), these interfaces need to be described by the content itself in order to be rendered on the mobile device. Different hardware (mostly regarding input) can be a limited factor for this approach. For example, a drawing canvas can benefit from the presence of a touch screen, but has to work on a regular mobile device featuring input through a keypad as well.

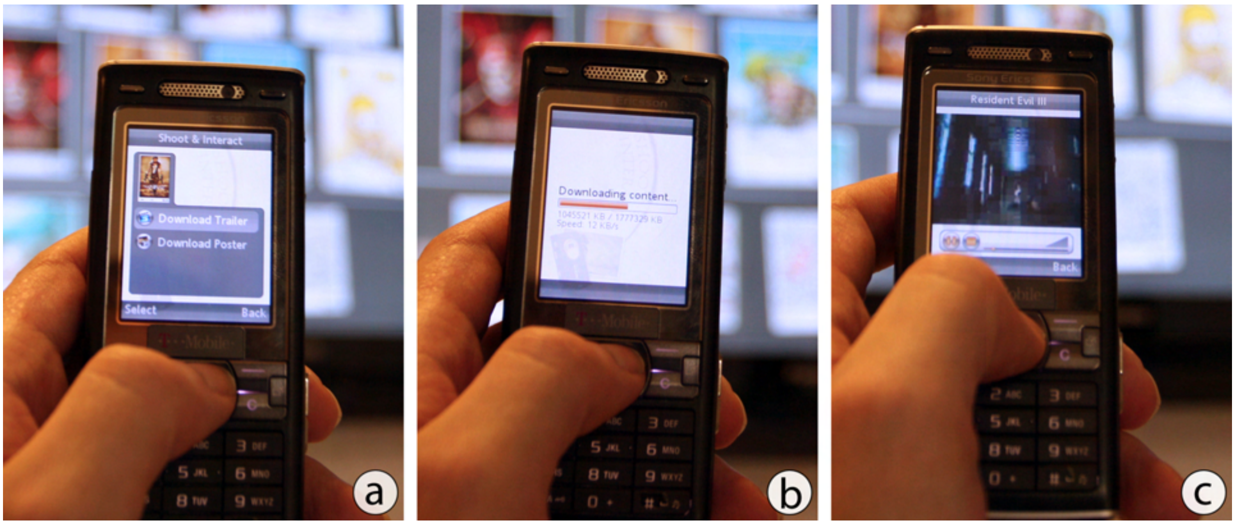


**Figure 5.8:** Walkthrough of *AdWall*: When an item has been selected successfully, users can select from a list of options associated to the content (a). After selecting the option ”download trailer”, the content is transferred to the mobile device (b). Users can then view the downloaded trailer associated to the selected movie poster (c).

As the bandwidth of Bluetooth is limited we chose to only transfer the list of options and the thumbnail after the selection has been made. When selecting an option, the additional content needs to be downloaded from the display to be rendered on the mobile device. While the content is being downloaded, the device again shows a screen notifying the user that data transfer is in progress. In early discussions with potential users we found that this is not harmful for the interaction as long as the download time does not exceed one minute. Users are familiar with this concept since they use mobile internet with eventually higher loading times throughout their day. Once the content has arrived fully, the device shows its predefined user interface. This interface is well-known to the device’s owner as they may use it for other media on their device. Instead of fully downloading the content, one may stream the data which is a considerable option for audio and video files. Comparing download times of compressed media files (e.g., 20 seconds for 1 MB) with their length (e.g., 1 minute), however, reveals that downloading is done faster. In the streaming scenario, users have to reside in the vicinity of the display to fully watch the content until it ends. Downloading the content, on the other hand, allows them to watch it (1) later and (2) multiple times. For this reason, we favored downloading content over streaming it.

Naturally, the content could have been directly shown on the large display as well. This then would avoid the waiting times resulting from downloading the media files. There are certain reasons against this procedure: first, as mentioned before, downloading the content allows users to watch it later and (if they want) repeatedly without being in the vicinity of the large display that hosts the information. And second, multiple users may produce race conditions. For example, the large display may show an image representation of a video clip. If this clip would now be played on the large display, others joining the environment later would have missed the beginning and have to wait until the original user is done watching it. For this reason, we preferred the large display as *point of accessing* media over a *point of consuming* media.

### 5.3.2 Applications and Implementation

To demonstrate the use of this interaction technique, we implemented two possible applications called *AdWall* and *PhotoWall*. In the first application, the public display showed covers of movies and music albums that are available at a nearby store. To break the rigid relationship of a public display only being an information broadcaster, users could select the image representation of an item using the aforementioned system. Once they selected an item, they were presented with options depending on the content. For movies, they were able to view a short trailer (about one minute) advertising it. If users selected a music album, the mobile device presented a list of all tracks. Choosing a track then caused the mobile device to play a 30 second sample (in lower quality for copyright protection) of it. This approach is similar to the previews users can access on online stores such as *Amazon* [4]. If they want to listen to another track, they could simply go back without the need to select the item again. With this, users can already have a personal preview of an album or video which may help them to decide whether they are interested. This application only allows selecting content from the remote display but not manipulating it.

In the second application, the public display showed pictures taken by various people similar to online photograph services such as *flickr*. In contrast to the first application, users are able to select a picture and read or write comments to it. Written comments are then immediately accessible to other users in the same way. As the public display shows a large number of pictures at the same time, these can be downloaded and displayed on the mobile device for a more detailed view. Selecting a picture results in two options: either viewing the picture or receiving the latest comments. If users select to download the picture, the image is transferred and stored on the mobile device. Subsequently, the image viewer allows users to get a closer look at it using familiar interactions built into the mobile device. If they want to obtain the comments written for the selected picture, a list of comments is being shown. Each comment contains the date and time, the author as well as the text itself. Users can read these comments and further write their

[4] *Amazon.com Inc.*: http://www.amazon.com/

own. The interface of writing a comment is similar to the one used for writing text messages (e.g., SMS[5]). After writing a comment, users can send the comment back to the display allowing everyone else to read it. Although the comments are not directly displayed on the external screen, users are able to manipulate the content that is being sent to others.

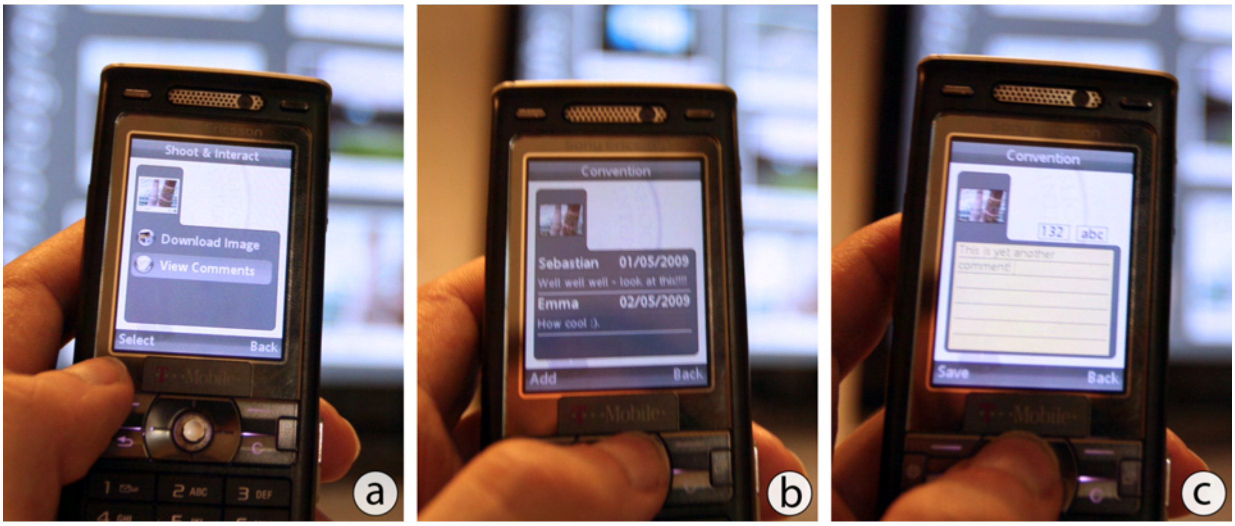


**Figure 5.9:** Walkthrough of *PhotoWall*: After selecting an item, users can choose from a list of options (a). After selecting the option ”view comments”, previously written comments are shown (b). Users can then write their own comment in a text-based interface (c).

We implemented our prototype on a Bluetooth-enabled mobile phone (Sony Ericsson K800i) featuring a 3 megapixel camera. The display has a resolution of 240 × 320 pixels and a diagonal of 2 inches. The pictures taken by the built-in camera had a resolution of 480 × 640 pixels. We chose to use this size to allow a fast transmission while still maintaining details within the picture. The mobile device was further capable of rendering MP3 audio (i.e., music samples) and 3GP video data (i.e., video trailer). We simulated a public display using a 50” plasma screen with 1366 × 768 pixels (see figure 5.10). This is the same display used in the studies described in previous chapters. All software components on the mobile device were written in JavaME (CLDC 1.1, and MIDP 2.0). The components running on the public display are implemented in C# for displaying content and Java for Bluetooth connectivity. The computer driving the large display featured a 3.0 GHz CPU and 2 GB of main memory to allow for fast image processing.

### 5.3.3 Performance and Qualitative Evaluation

In a study we tested the performance of the presented system regarding overall recognition time and error rate. We further wanted to see whether participants understand the selection. We wanted to gain insights into the acceptance and usability of such systems. However, we did not compare

[5] SMS stands for *short message service*

the system against existing solutions due to the increased selection time caused by the image transmission and analysis. In terms of performance, we present the outcomes only for selecting remote content. The interaction and manipulation with it served as basis for qualitative feedback.

During our test, we showed 12 different images on the display (see figure 5.10). The visual representation of these images differed with at least 20%. This difference is measured by subtracting one image from another and subsequently building the ratio of different pixels to overall pixels. Pixels are considered different if they differ in terms of their grayscale value with more then 25 units (out of 255). Although the pictures showed different objects, they may be equal from the computer's point of view in terms of structure and color. This increases the risk of detecting the wrong image, but also gives insights into the performance of our algorithm. Each picture was placed with its neighbors in a way that theoretically each item can be identified uniquely. We logged the pure processing time (i.e., the time needed for both image and result transmissions were excluded) as well as the error rates. Selection errors can have two characteristics: first, the system is not able to detect anything asking the user to select the item again. And second, the algorithm detects the wrong item requiring the user to go back and select the item again. We did not distinguish between these errors as they both lead to the same result of having the user to re-select the intended content.

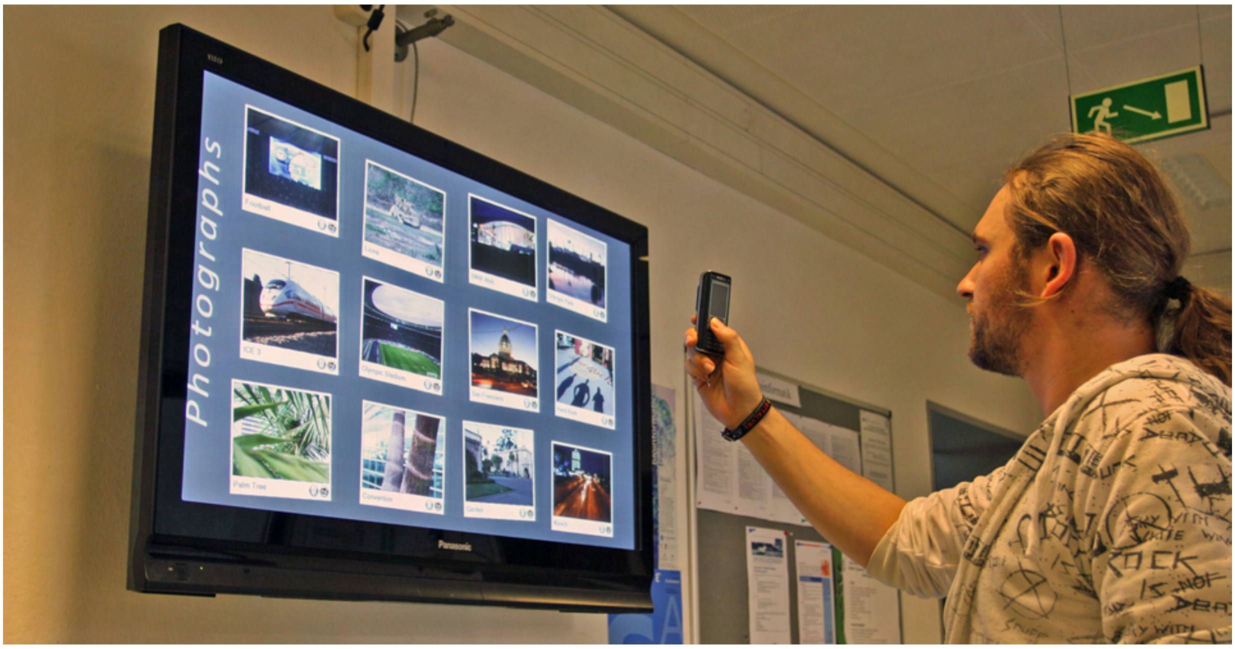


**Figure 5.10:** A person using our prototype on a large display.

We recruited 28 participants (14 were female) ranging in age from 18 to 47 years (average age was 25.2). 25 participants own a mobile phone, but only 15 of them use the built-in camera. During the test, we asked participants to capture each picture in a randomized order. If the picture was the correct one, they could go back and select the next one. If the system was not able to detect a picture (or it detected the wrong one), we increased the error count and asked the participants to select it again. The processing times of failed selections were not further considered when the system was not able to identify anything. This is because a missed detection of an item could lead to extremely short detection times as the contour detection might have failed already. This would cause the system to stop even before any image comparison mechanisms (which we consider the bottleneck) would take place. Missed detections of the mentioned nature would result in an overall lower processing (and shorter selection) time. After correctly selecting each image, participants had to fill a post-questionnaire also stating their opinion about the system.

In terms of performance we found that our system was reasonably well suited for short interaction sessions such as selecting content. The average detection time of selected content was about 180 ms. The error rate was about 4% (i.e., only 4 out of 100 taken pictures resulted in a wrong detection). Most of the errors (about 90%) resulted in not detecting the item at all. This was mainly caused by the slight movement of the mobile device while selecting the content (i.e., pressing the ”select” button). The taken image was then affected by motion blur making the detection of contours complicated due to the missing contrast. Delivering wrong results (i.e., the system favors another item), on the other hand, was mainly caused by bad lighting conditions such as flickering occurring in fluorescent lamps. While having a noticeably low error rate, we acknowledge that increasing the number of pictures on a display (and potentially lowering the pair-wise difference of them) would result in both higher processing times and higher error rates. For numerous items as well as real-time interactions, the approach needs to be revised.

In post-questionnaires users ranked the overall use of the system as well as its ease-of-use. In addition, we encouraged them to give further feedback about our prototype. On a five-point Likert-scale (1 equals ”I dislike the system” and 5 equals ”I like the system”) the average rating was 4.1 indicating that the system was appreciated among our participants. When asking whether they would use such a system in public spaces, we found that 82% (exactly the same as participants that liked the overall idea) would like to use it. During the test, the Bluetooth connection time was a large portion of the overall selection time (i.e., *connecting*, *selecting*, *sending*, and *receiving*). The average time for a successful selection was 9.2 seconds although the actual content recognition only took 180 milliseconds. While we thought that this time would decrease the ratings, participants stated that a selection time of 10 seconds still meets the criteria for interacting during a short period of time. This is also in line with Rukzio et al. [RSH04]. The most interesting statement regarding our system was that participants felt comfortable in using the prototype regarding privacy. They stated that although everybody sees *that* a person is accessing information, no one can actually see *which* information has been captured. Our system and the underlying concept of interacting *through the display* may be a first step towards more private interactions on external displays by allowing users to select distant content on their local display.

### 5.3.4 Limitations

Although ranked well by the participants of the study, our prototype still has several limitations that need to be addressed in order to increase its usability. Embedding the connection process into the startup procedure of the application already gave the impression of a connection-less procedure. The time needed for discovering a service (and the associated display) still requires several seconds requiring the user to wait. Especially in fast-paced scenarios such as having a minute before a train arrives, this may be an extremely limiting factor. Furthermore, while being invisible to the user, the connection between the mobile device and a single display still exists on the technical level. To keep the connection transparent to the user, the strict connection to a single display is not suitable. To overcome this limitation, the aforementioned services may reside on a centralized component within a public space acting as single instance serving multiple displays.

The association of a display already at startup further limits users if the wish to select content from a different display. To do so, they have to close the application and then start it again. This does not guarantee that they connect to the correct display. This type of interaction can be important if cross-display operations are needed. For example, a user selects one item from one display and wants to paste it on another. Disconnecting from one screen and connecting to another can only be done if (1) the displays are uniquely identifiable and (2) the user closes and starts the application again once the destination display has been reached. The previously mentioned procedure of having a centralized service would again overcome this. In the following section, we describe how our prototype is changed to enable such infrastructures.

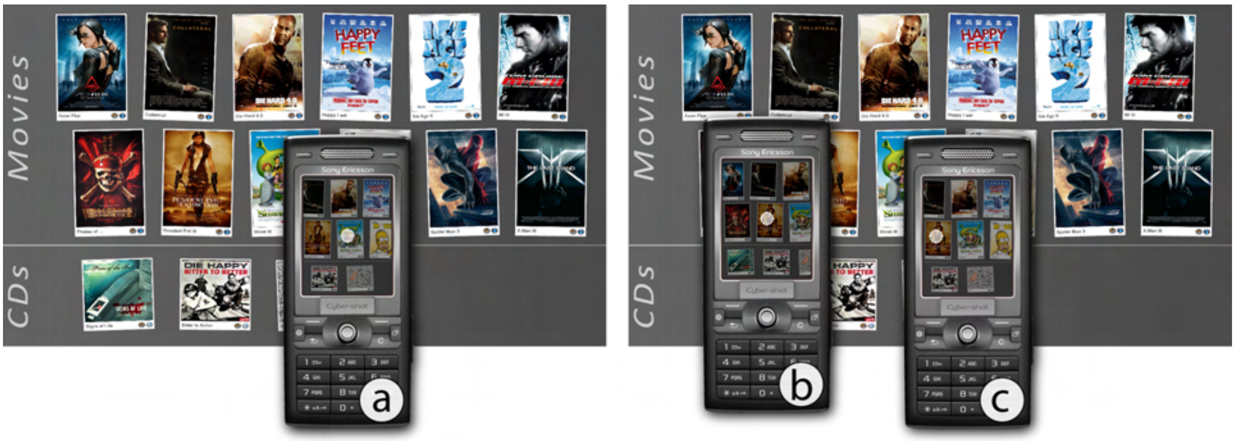


**Figure 5.11:** Comparison of moving the device versus moving the crosshair: the user aims at a movie (a) and wants to select another. The user has to either move the device to get the new content underneath the crosshair (b) or shift the crosshair to the desired content (c).

Another limitation presented in our prototype the fixed selection area in terms of a crosshair overlaid on the live video image. Users need to move the device even though content is already visible in the viewfinder. One solution to this problem is to allow users to move the crosshair using the mobile device's joystick. This would then increase the overall selection time. In turn it decreases the device's movement which allows bystanders to observe the interaction (see figure 5.11). As modern mobile devices nowadays are equipped with touch screens, the cumbersome procedure of shifting the crosshair to another position can be avoided by using direct touch input. Users could tap on the item shown in the viewfinder. The system then translates the coordinates of the touch point into the remote display's coordinate system. The presented image processing and transformation approach already supports this (see chapters 7 and 6).

# 5.4 Allowing Multiple Displays in the Environment

While our prototype works well on single displays with a well-defined relationship in terms of wireless connections, some tasks may require accessing multiple displays in the same environment. Such displays can either coexist without the need for cross-display operations. A good example for such a scenario is a subway station equipped with several individual displays showing different content. A more extreme case is New York's Times Square with a large number of heterogeneous (i.e., different sizes and purposes) displays. Besides being able to access one out of several displays, users may also wish to transfer content from one to another. These cross-display operations need to be handled as well if multiple displays are present in the environment. As discussed in the previous section, the infrastructure needs to be changed by having a centralized component as *environment manager*. In this section, we first discuss the necessary changes that have to be made to our architecture in order to meet these tasks. We then show how multiple displays can be differentiated in general. We then describe the necessary changes that have to be made to our architecture in order to meet these tasks.

## 5.4.1 Managing the Environment

In environments with several screens, each display is usually driven by its own computer (or processing unit). From the technical point of view, a connection has to be established between the mobile device and the target display. Several solutions exist depending on the task. If the environment only allows for accessing information on a certain display, the user can take a picture through the mobile device which is then sent as multicast package to displays connected to the same network. Displays can then evaluate whether the picture represents a portion of their content or not. If the picture matches a display's content, the screen can respond to the multicast. Due to potentially wrong detections, multiple displays may answer positively to a single multicast. The mobile device then does not know which (or if even one) of these displays is the currently targeted one. While being a promising solution, a decentralized approach may fail if the displays act as autarkic devices not communicating to each other.

To allow the idea of having an entire environment acting as a single device and display respectively, a centralized managing component can be installed. This approach is similar to the one described in chapter 4. This *environment manager* is the gateway for traffic between the users' mobile devices and any present display (and its computer respectively). Pictures taken by any mobile device can be sent to the environment manager. The question is then at which component (i.e., each display or the environment manager itself) the pictures are analyzed. On the one hand, each display can determine on its own whether a picture matches its content. This leads to the same problems such as multiple positive responses by several displays. Besides being a gateway, on the other hand, the environment manager can also evaluate the images coming from mobile devices (see figure 5.12). This decreases the computational resources needed on each display by having a high-performance machine as analyzing instance. The displays then require less processing power as they simply display content.

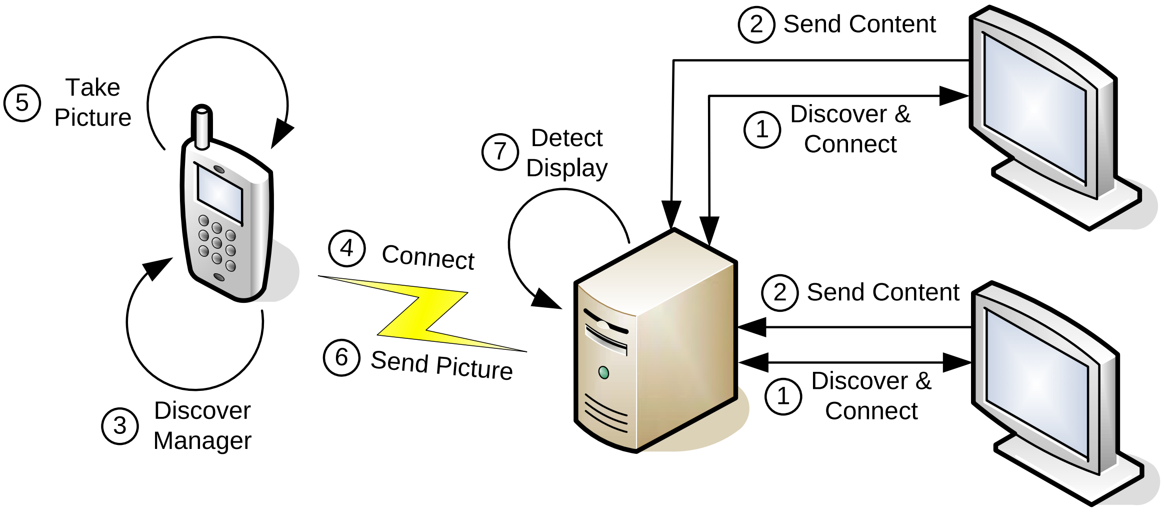


**Figure 5.12:** Architecture for environments with multiple displays: On startup, each display discovers the *environment manager* and connects to it (1). They also send their content for image analysis (2). Mobile devices also discover the environment manager (3) and connect to it (4). When users take a picture (5), the image is sent to the managing instance (6). There it is analyzed based on the content of each and every display (7).

To enable the image processing on this centralized server, the managing component has to know each display in the environment. This means, that the content and its constellation on each display needs to be present at all times. On startup, displays are discovering the environment manager in their public space. Subsequently, they connect to it and deliver content information. A simple screenshot is sufficient as the same image processing methods can be used to separate content from background. A more reliable solution is to transfer the content combined with geometrical information directly. By knowing the content, the environment manager is now able to associate incoming pictures from mobile devices to displays in the area. Furthermore, it always results in an unambiguous result: either *one* display has been found (this has not necessarily to be the correct one) or *no* screen has been detected. Obtaining the content of each display only once (during startup) may not be sufficient for longer time periods. For example, if the display changes its content over time (e.g., an animation is shown or users influence the screen's content), it is not possible for the environment manager to either (1) identify the display at all or (2) calculating the correct transformation between the display and a mobile device. Updates have to be communicated to the environment manager whenever they occur. In theory, this can again be done by sending a screenshot. Only sending position updates for items already known to the environment manager may decrease the necessary bandwidth. This approach is theoretically also possible in distributed scenarios without having a centralized instance. However, for testing purposes we chose to rely on the environment manager.

If users want to interact through their mobile devices, the underlying application discovers the environment manager in the same network and connects to it. Whenever users take a picture indicating a selection, the mobile device sends this picture to the environment manager which then evaluates the image and identifies the display and its portion respectively (see section 5.4.2). In theory, the environment manager could send the connection address of the corresponding display to the mobile device for a direct connection. As this would require the mobile device to reconnect to the environment manager for each selection, we decided to keep a permanent connection between the mobile device and the managing component. If each display registers with full information instead of sending a screenshot only, the managing component becomes a machine with multiple displays attached. From the users' point of view, each display in the environment presents a portion of the information in the area than acting as a singleton.

## 5.4.2 Distinguishing Content and Displays in the Environment

Our first implemented scenario only allowed content to be *selected* instead of being manipulated in place. Even if the same content resides on several displays (e.g., the same album cover on multiple advertising displays), the exact location and its host is not important. Accessing content does not necessarily mean to have a direct connection to the targeted display. It rather assumes to have a connection to the centralized instance which knows the list of options associated to each content item. Therefore, matching content items individually may be sufficient in such scenarios. Taking neighboring relationships into account is then only needed if ambiguous results exist based on the image processing and comparison. This approach only works if content is selected directly without any option to paste items or manipulate items in place.

If the location of the content is important, the correct target display needs to be identified based on the captured image (see figure 5.13). This seems to be a trivial task if each display has a unique set of items (i.e., no item is shown on more than one display). The neighboring information again can be used to verify ambiguous results. This optimal case most likely does not exist in real settings. For example, multiple displays in a subway station could advertise for the same store which increases the risk of two items of a kind being shown on two different screens. Thus, the items in the neighborhood of the reference item need to be taken into consideration. This approach assumes that two displays are different by means of their item distribution on their canvas. Hence, two displays can have the exact same items but are still distinguishable if the items are placed in a different structure. These structures must be further independent from each other, meaning that one cannot created from the other by translating, rotating or scaling the entire canvas. If this is not the case, the system will not be able to determine the target display. One solution may be to take other contextual information (e.g., the display's surrounding) into account. This is possible if the picture taken by the mobile device contains portions outside the display. We admit that this is rather unlikely for very large displays (or at least only occurring far away from the display's center) presenting a true limitation of our technique. This general issue of optical tracking solutions can only be solved by other tracking mechanisms.

Users may also want to address a location on the display's canvas which is not inside the boundaries of a content item. This is the case, for example, when users want to paste a previously selected item (or an item stored on their mobile device) onto a certain position. Our technique allows selecting such positions based on the visual and geometrical appearance of a display's content. The content of the display implicitly becomes the visual marker. The aforementioned problem of having two displays with the exact same representation would not allow for such an operation. We again have to assume that two displays are either visually or geometrically different. In the best case, they differ in both. Similar to previous tasks, the ambiguity for selecting the targeted display based on the content can also be overcome by presenting a list of possible candidates (e.g., images of the displays). While this seems to work for discrete and sequential operations as presented in this chapter, immediate content-specific feedback cannot be given using this approach (see chapter 6).

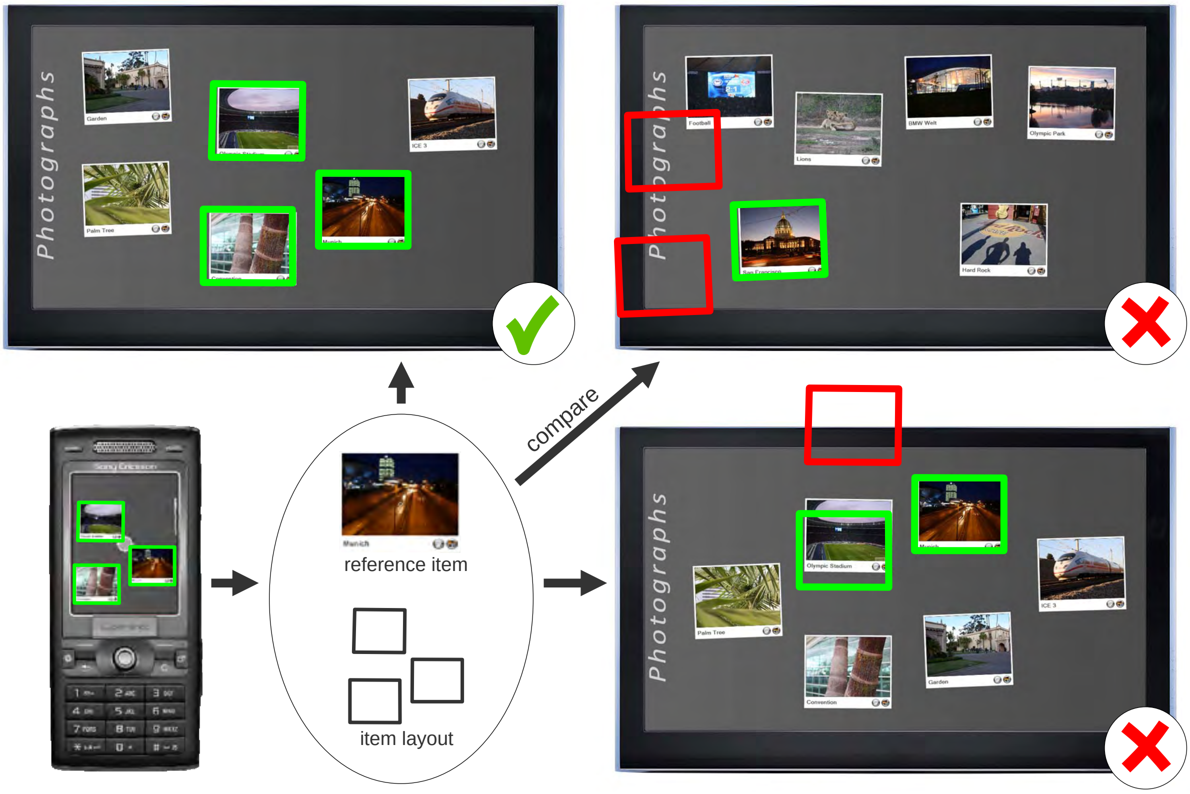


**Figure 5.13:** Distinguishing multiple displays in the environment. The analysis of the taken picture reveals the reference item and the item layout. These can now be compared against all available displays. The top left display turns out to be the correct one. Top right shows a display with entirely different content (i.e., both layout and items). The closest item on this screen can naturally be matched geometrically. Lower right denotes a display showing the same content as the correct one. As the geometrical layout is different, the display cannot be correct. The green rectangles illustrate correct matches, the red ones depict incorrect ones.

# 5.5 Summary and Discussion

In this chapter we have presented the first stage of our new model of interacting *through the display*. We described how a display and its targeted sub-region respectively can be identified purely based on the visual and geometrical representation in the captured image. We further showed two prototypes making use of this algorithm. With these prototypes we were able to confirm the validity of our image processing approach. We further showed how the concept can be transferred to environments hosting multiple displays. Therefore, we introduced a managing component that knows all displays and their content in the environment and further acts as gateway for mobile devices. Only if two displays have equal visual and geometrical content, the distinction of them is impossible using our approach unless the user makes a decision (e.g., selecting the correct display out of a list of ambiguous results). From the users' point of view, the connection happens in an *non-modal* way and is thus combined with the content or region selection. The technical foundation still requires a connection to the environment manager. We believe that this can be done using discoverable services similar to DNS[6] in combination with location information.

In a first user study we found that participants understood the concept and were able to use it. Furthermore, we gained interesting insights regarding the interaction itself. Two noticeable statements were: first, the long time needed for sending the image and receiving the results is still acceptable. And second, users perceived a higher level of privacy while interacting with the system as everybody could see *that* they capture information, but no one was able to see *which* information they were targeting at. The system considers *privacy* to some extent which may be important in certain scenarios. Furthermore, the detection rate was encouraging even with similar items shown on one display. We found that the geometrical properties were a good addition to the pure visual identifiers. The similar content gets, the more error-prone the technique may be. The rather low error rates combined with the fast image processing techniques make this approach a suitable candidate for *through the display* interaction. However, when *continuous* interaction is required, the image processing time of this *discrete* approach needs to be reduced further at least for subsequent frames.

According to the design space denoted in section 1.2, our prototypes allow *absolute* and *direct* interaction in a *discrete* fashion. The distance is still *limited* due to wide-angle lenses (and a greater field of view respectively) used in today's mobile devices. When incorporating the managing component, the environment is *flexible* as the interaction is dependent neither on the exact locations of displays nor on special input technologies. Furthermore, the system contains a *non-modal* connection mechanism from the user's point of view. Considering simple selection tasks (such as in *AdWall*) performed by multiple users, no race conditions can appear as users may download media simultaneously. This system is then usable by a large number of users.

We also found several limitations which we will address in the remainder of this work: first, using Bluetooth as communication technology limits the implicitness of the interaction. One solution is to use a faster and more flexible technology such as wireless LAN or even 3G networks (which is nearly always accessible providing sufficient bandwidth). On the other hand, this counteracts

[6] DNS stands for *Domain Name System*

on Bluetooth as a provider for location information (see section 4.1). Second, taking and sending pictures explicitly does not allow for fluent and continuous operations. This may be especially needed if users want to drag items on a single display or from one display to another. The image processing mechanism, however, needs to be extended to support real-time interactions (i.e., less than 100 ms per frame). Furthermore, continuous operations require a permanent video image. The mobility of the input device plays a more important role compared to discrete approaches. And third, placing the crosshair in the center of the display causes a device movement for each selection although the item of interest may already be visible in the viewfinder. Allowing users to shift the crosshair or even to use, for example, direct touch will help to overcome this issue. By addressing these issues, we believe that our model serves as a good candidate for interacting on external displays in the future.

Chapter 6

# *Tap & Drop*: Interacting with Local and Remote Content

*An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.*

**– John W. Tukey –**

The prototype presented in the previous chapter allows for discrete interaction on external displays by taking a picture of the desired region. However, the phases of *content selection* and *content manipulation* are consecutive. This in turn does not allow for real-time interactions such as obtaining personalized feedback for a certain remote item while already pointing at it. In the advertising scenario (see chapter 5), for example, users had to take a picture of the desired content which then resulted in a list of options on the mobile device. In this chapter we[1] discuss how the entire process of (1) capturing the image, (2) sending it to the environment, and (3) receiving the list of options can be reduced to a single step.When users point at the item of interest, the system immediately shows the feedback associated to the content (e.g., a list of further options). To overcome the second limitation – namely having the item of interest to be in the center underneath a crosshair – we chose to use a touch-enabled mobile device. In doing so, users can (1) coarsely position the mobile device using their non-dominant hand (i.e., until the item is shown in the viewfinder) and (2) select the item and one of the options respectively using a finger of the dominant hand. Besides showing individual feedback only, the mobile display can render personal content items. Similar to the approach presented in chapter 3, our prototype allows for

[1] The work presented in this section has been published in several scientific papers [BBB$^{+}$10, BB10]. The scientific plural refers to all authors of these publications – namely Sebastian Boring, Dominikus Baur, Andreas Butz, Sean Gustafson, and Patrick Baudisch.

transferring items back and forth using a single tap. To allow for such kinds of interactions, we present an adaptation of our tracking solution in order to determine the mobile device's position and orientation with respect to a display in the environment. With further refinements our tracking solution allows for real-time interaction on various displays.

The remainder of this chapter is structured as follows: We first discuss a bimanual approach for accessing displays and their items in a more efficient way (see section 6.1). We then describe the necessary changes to the previous setup – namely real-time display identification and touch point transformations (see section 6.2). Subsequently, we focus on variations of local content being superimposed on remote content (see section 6.3) and present necessary changes to the environment manager to allow for such interactions (see section 6.4). We describe a proof-of-concept prototype that allows users exchanging data between their mobile device and the public screen (see section 6.5). Finally, we conclude this chapter by summarizing the approach and discussing the different variations of employing the mobile display (see section 6.6).

## 6.1 Bimanual Touch-based Interaction

In chapter 5 we discussed the limitations of the centered crosshair for content acquisition. The crosshair's placement reduces the input area to a small area. This then requires that users always have to center the content they wish to access although the item may already be seen on the edge of the mobile display and is visually reachable. We identified several solutions to this problem. On the one hand, users can shift the crosshair using the mobile device's joystick. However, in chapter 4 we discussed moving a personal pointer on the external display by using the same input strategy. We found that it to be the worst technique compared to any other continuous interaction (i.e., moving or tilting the device for pointer control). Although the canvas on which the pointer and crosshair respectively reside is much smaller compared to the external display, we think that this option would rather increase the task time than decrease it. The only benefit users may gain from using this technique is that bystanders are not exactly aware of the addressed content. Others in the surrounding see the region users point at, but they do not exactly know the crosshair's location on the mobile display. Hence, they can only estimate the content item.

A more promising approach is to use touch-enabled devices such as Apple's iPhone or Nokia's N97. Users can directly interact with content using their bare fingers. Instead of interacting on local content through touch, users may also use their fingers to interact in live video. The need for the cumbersome shift of the crosshair disappears. If users see an item, they can directly access it with a finger as soon as it is visible on the local display. The touch location can then be transformed into the target display's coordinate space as described in chapter 5. As all precise operations, interacting with fingers in an accurate fashion on touch screens is usually carried out with the user's dominant hand. The other hand (i.e., the non-dominant hand) holds and stabilizes the device at the same time (see figure 6.1). Thus, both hands work in concert. The non-dominant hand holds the device and coarsely orients it, while the dominant hand interacts within the reference frame established by the non-dominant hand. This turns the mobile device into a *toolglass*-like device [BSP$^+$93]. Furthermore, it allows users to bridge large distances by moving the entire device (i.e., positioning the reference frame with the non-dominant hand) as well as fine-grained manipulation by performing touch input with the dominant hand [EB09].

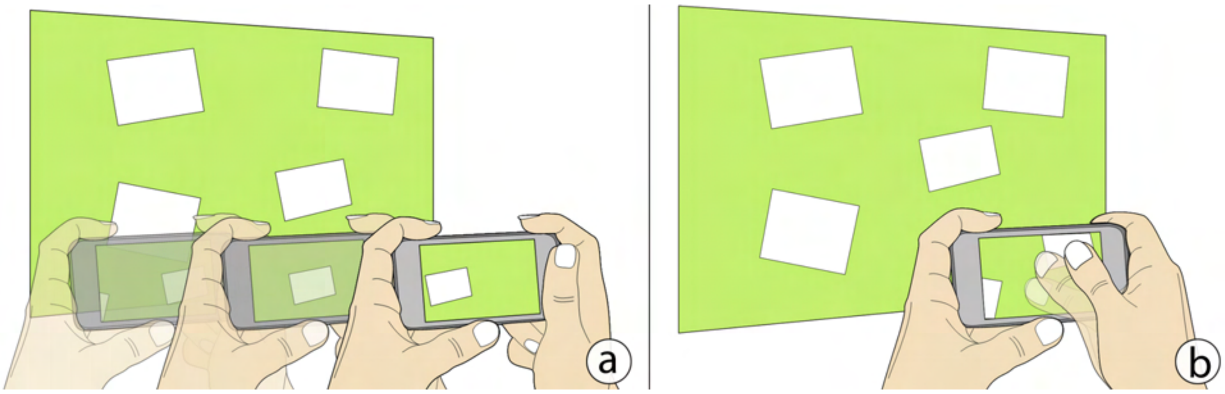


**Figure 6.1:** Coarse and fine-grained bimanual interaction. (a) Moving the device spans the reference frame. (b) A finger can then interact in this reference frame.

Bimanual interactions can be carried out in two ways: first, both hands can work independently. This is the case when accessing multiple items in order to pile them at a predefined location. In other cases, the hands depend on each other. The prime example is to write on a sheet of paper. Guiard found that the non-dominant hand permanently repositions the paper spanning the reference frame for the dominant hand [Gui87]. The area within which participants wrote was much smaller compared to the size of the paper page. Our interaction style falls into the second category (i.e., one hand influences the other one). In a comparative study, Kabbash et al. found that these *dependent* techniques outperform the *independent* ones [KBS94] since they use less motor operations. Guimbretière et al. confirmed these results and found merging command selection and direct manipulation to be the most important factor [GMW05]. Buxton et al. further found that single-handed input as used chapter 5 performs worse compared to bimanual input for selection, positioning, and navigation tasks [BM86]. Latulipe et al. extended this set of operations with manipulation tasks for multi-parameter functions such as image corrections [LKC05]. Hilliges et al. present a bimanual approach for tabletops [HBB07]. Similar to our system, the non-dominant hand spans the reference frame (here: a period of time) while the dominant hand can precisely interact with content within the reference frame. They found that such a design provides benefits for browsing, organizing and sharing digital media collections. Considering these results, we can assume that our bimanual, touch-based approach will outperform single-handed interaction.

## 6.2 Enabling Real-time Interactions

Although already a promising candidate for interacting with external displays, *Shoot & Copy* (see chapter 5) had two limitations. The user's *action* (i.e., selecting an item by taking a picture) and the system's *reaction* (i.e., showing a list of options regarding the selected content) happened in a sequential way. However, to ensure direct feedback on the mobile device without the need of explicitly selecting an item first, the spatial relationship between the external screen and the mobile device needs to be sensed permanently. The second limitation is the need for centering the item of interest in the mobile device's viewfinder (unless the crosshair can be shifted explicitly). To overcome this, we chose to use a touch-enabled mobile device allowing for a more flexible input technique. In this section, we discuss the necessary real-time tracking (see section 6.2.1) followed by a description of bringing touch to displays in the environment (see section 6.2.2).

## 6.2.1 Real-time Identification of Target Displays

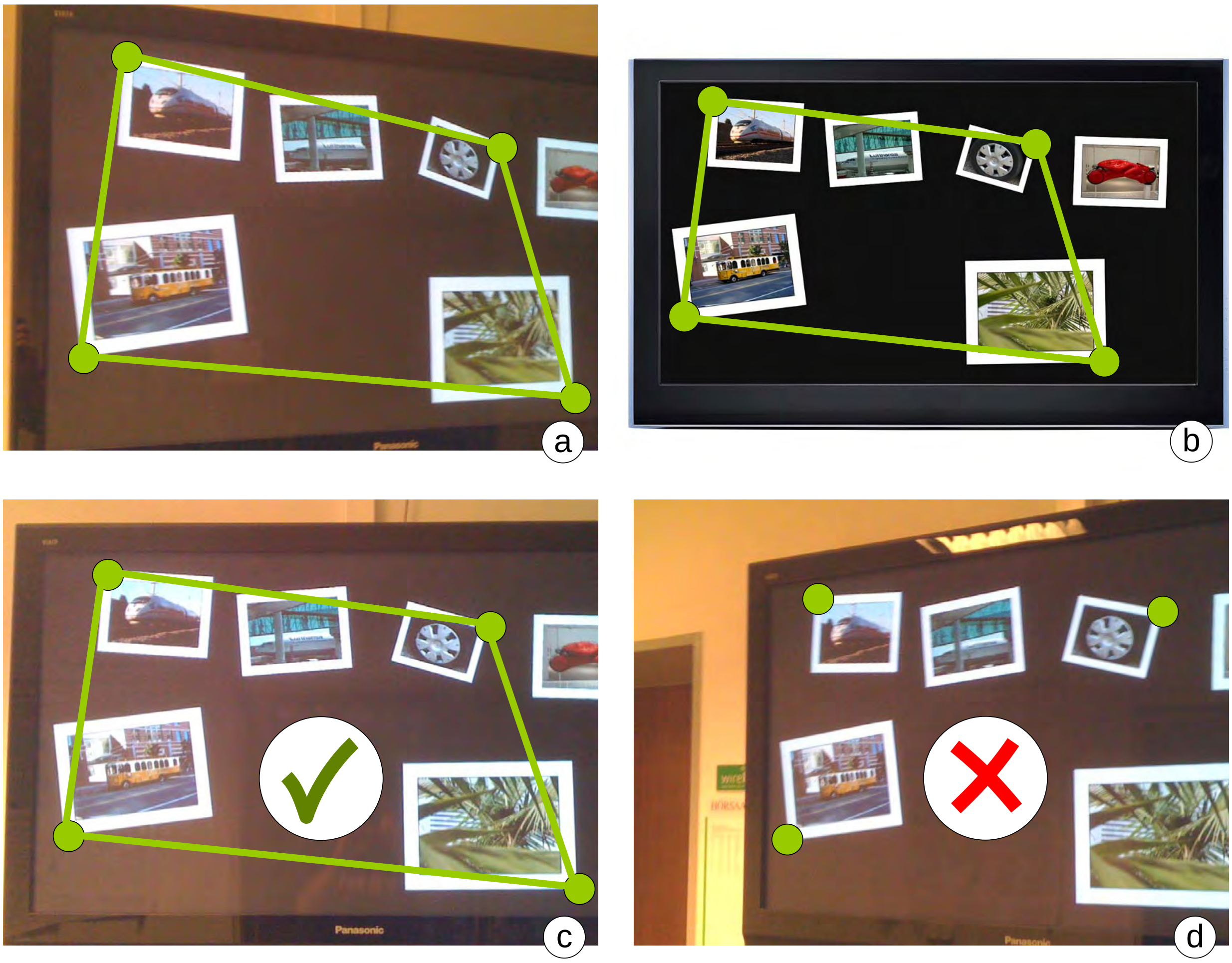


**Figure 6.2:** Feature point tracing to allow for real-time image processing: (a) The first frame is processed fully to calculate the correct transformation. (b) The corner points for the final transformation on the target display. (b) These points are found in the next frame. (c) One of the four feature points cannot be detected.

In theory, the detection process described in chapter 5 can be used for real-time identification and tracking as well. This would result in a higher processor load which is especially harmful when multiple users interact at the same. The recognition time of 180 ms (or eventually more depending on the amount of displays and content items) does not allow for true real-time interaction. According to Azuma, augmented realty systems are considered real-time if the delay is 100 ms or less [Azu97]. With this in mind, the presented approach – already being too slow for real-time interaction – does not scale and hence limits the *flexibility* of the environment. Nevertheless, this registration is needed to allow for real-time feedback *through the display*. Thus, we needed to optimize the algorithm's performance. One solution is to use information gathered in the previous frame which in turn may decrease the steps needed during a single detection cycle.

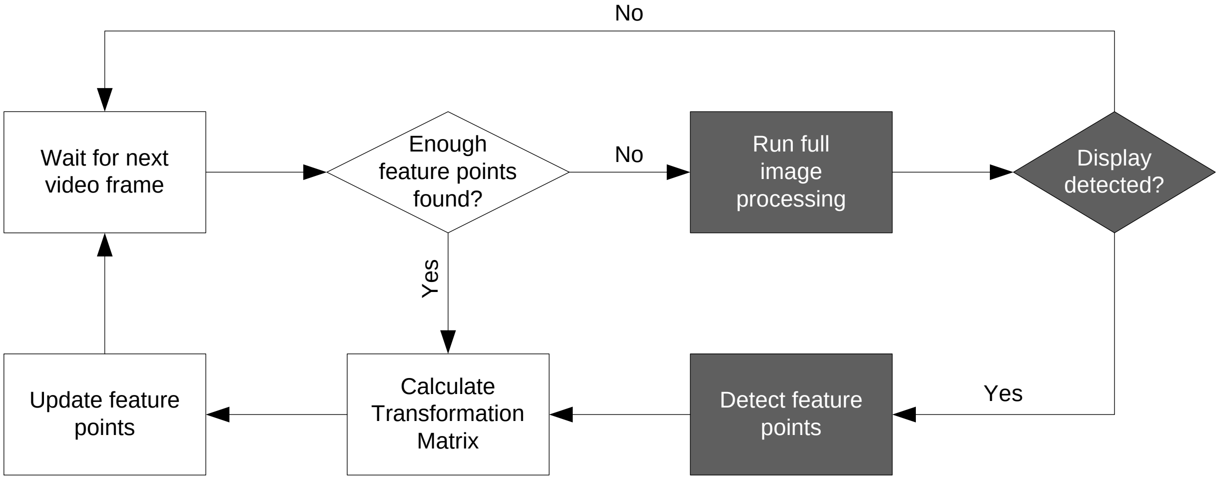

No
Wait for next video frame
Enough feature points found?
No
Run full image processing
Display detected?
Yes
Update feature points
Calculate Transformation Matrix
Detect feature points
Yes

**Figure 6.3:** The image recognition process for real-time feedback. The shaded areas denote the processes with high computational requirements. These can be left out as long as feature points are available and can further be detected in the new frame.

During the detection process of a display (and its content), we obtain the corner points of each identified polygon. Four of these points (describing the largest possible quadrangle) are used to calculate the final transformation. It is sufficient to detect these points only. As our camera (i.e., the camera of the mobile device) does not remain still during the interaction, these points have to be identified repeatedly to permanently allow a correct transformation between both image planes. We can assume that the mobile device will not move far between two subsequent frames. Hence, there is a chance that the points detected in one frame will still be visible in the next frame although slightly shifted. If these points can be found in subsequent frames, the entire image processing can be reduced which in turn saves time and processor load. Especially removing the *Hough Transformation* would decrease the detection time significantly. Since the system still has the points' coordinates in the coordinate system of the external display, it can calculate the transformation as described earlier. This approach requires both high accuracy in tracing the points as well as a clear association between points in the previous frame and those in the current one. If a point in the current frame is associated with the wrong one in the previous frame, the calculated transformation will be incorrect leading to unexpected responses. The system also needs to have a fallback solution if one of the points cannot be found.

Similarly to the optical flow analysis used in chapter 4 for the motion detection of the mobile device, we employ this method to find points in successive frames [BJB09]. In the mentioned system, we chose to track so-called *feature points* that have been detected in a single frame and traced in subsequent ones [ST94]. These points are significantly different from others in the image and can therefore be identified easier in successive images. We cannot guarantee that the identified feature points are corner points in the image. Most likely, these points lie inside of images as the frequency (i.e., their set of surrounding features) is extremely high compared to others. As we have to match points detected in the video image with points on the external display, we have to know the exact location of each point in screen coordinates. This is necessary to calculate the transformation once the points have been detected. The feature points need to be predefined by the system to ensure that only points with known coordinates are tracked. Using the *Lucas-Kanade Optical Flow Method*, we use the detected corner points (from a frame analyzed with full image processing) as feature points [LK81]. Their implementation uses a *coarse-to-fine* approach in scale-space (i.e., a pyramid). This means, that first the larger surroundings of pixels are matched followed by (if still appropriate) a matching of closely neighboring pixels. As our feature points are directly on the corner of a polygon, this method allows for good results as it takes the item's visual content into account (see figure 6.2).

By using this method, we can reduce the processing time significantly at least for subsequent frames. This method allows the detection of these points in less than 20 milliseconds. Including the subsequent matrix calculation, the entire process for each frame with predefined points (i.e., identified in the first frame) lasts 25 milliseconds at most. We found that the approach is suitable for real-time interaction. As mentioned before the success of this method depends on whether these points can be found in the frame. There are two main reasons that limit the chance of detecting these points: first, the video frame contains motion blur which may cause the method to fail. And second, the device moved far enough in a way that not all points are still within the video frame. In both cases, the system is unable to find *all* points. One obvious solution to these problems is to rerun the frame fully as described in chapter 5. This approach causes a slight delay during the interaction. A second alternative is to use four other points in the image. These points then do not describe the largest possible quadrangle which in turn decreases the accuracy.

Our solution (as shown in figure 6.3) uses the first approach of running a full image analysis when the system does detect all points in a subsequent frame. To limit the processing time especially for environments with multiple screens, we first use the display and its content respectively that has been detected during the last successful cycle. If matching is still not possible, the system compares the given frame with all other displays unless it finds a target display. If no display has been found, the system assumes that the mobile device is currently not pointed towards a screen. If a target display has been found the system stores the detected corner points and processes each subsequent frame using *optical flow analysis* until a full rerun is needed (i.e., not all four points were detected in a frame). With this, the average detection time per frame is reduced significantly. This, of course, depends on the movement of the device as well as the quality of the images produced by the device's camera.

### 6.2.2 Bringing Touch to Displays in the Environment

Since we need to process images in real-time, we chose to switch from Bluetooth to the faster and more reliable wireless LAN 802.11g. While the connection procedure is still similar to the Bluetooth version (i.e., discover the environment manager followed by a connection), it is much faster. Thus, this step can be integrated into the application launch as it does not consume much time. Discovering the environment manager is done using IP multicast. The response of the environment manager then includes its IP address and port. Displays in the environment connect using the exact same procedure. The communication with the environment manager is then done via TCP to allow for a reliable point-to-point connection. Similarly to the first prototype, the underlying connection is already established before the user interacts with the system. As the process is hidden in the application launch, users are not aware of being connected to the system already. A different approach would be to use the images coming from the live video as discovery messages and subsequently establish a point-to-point connection to the display. This in turn would decrease the detection performance significantly.

Once the connection is established, the mobile device starts to stream the live video content frame-by-frame to the environment manager. The centralized instance then analyzes the image using the previously described procedure (i.e., full image processing versus point tracing). If a display has been found, visual feedback is given to the user in the form of luminous virtual LEDs. The necessary information of transforming input points from the local device into the coordinate system of the current target display (i.e., the transformation matrix) is known by the environment manager throughout at all times. From the technical point of view, the managing instance tracks the mobile device based on the delivered camera stream which does not require additional equipment in the environment. When a touch event occurs, the centralized instance transforms it with the current transformation and sends it to the target display. The input is then performed on the external display. As users see the content through their viewfinder, they perceive the content to be manipulated directly on their mobile device. Users can interact with displays in the environment using touch input as if they were touching them directly.

Assuming a one-to-one relationship between users and their mobile devices, we can further associate touches with persons. Especially for research regarding interactive surfaces, this seems to be a rather limiting factor. However, there are situations which require this association. For example, some items may only be accessed by certain users but should not perform any action if touched by others. Nearly all systems (with the exception of the DiamondTouch [DL01]) suffer from the missing association between finger and user. Users located around such an interactive surface have equal rights. Our prototype easily allows this by associating all touch events coming from a single mobile device to one person only. The concept of ownership (and the resulting access rights respectively) of items can now be used. This means that users can only interact with an item (e.g., moving it from the target display to the mobile device) if they have the access rights to do so. Naturally, different levels of access rights can also be used.

# 6.3 Superimposing Local on Remote Content

When personal content is superimposed on external content shown in the viewfinder, the role of the mobile device is important. On the one hand, the mobile device can act as an input device only as described before. The superimposed content then presents temporary feedback for the corresponding user only. On the other hand, the mobile device may show its own content in addition to remote content shown in the viewfinder. This then results in two independent content layers shown on the mobile device once it is targeted at an external display. In this section, we present both strategies to understand how they influence our architecture.

## 6.3.1 Two Independent Content Layers

Besides showing content from external displays in the device's viewfinder when it is targeted at a screen, the mobile display can also show its own content. In contrast to remote information, the personal content is shown permanently and remains fixed with respect to the mobile device's boundaries. Naturally, the local content does not rely on the device being pointed at certain displays. Users can see the data whenever to want and further are able to interact with it. Underneath this fixed content layer, the viewfinder (e.g., the live camera preview) can be shown to allow for the previously presented interaction. In order to ensure that users are able to see the viewfinder, the personal content layer needs to show the items only without any background image (e.g., a fully transparent background). In this way, the device shows two layers – the *local layer* showing the personal items as well as the *remote layer* shown in the viewfinder (see figure 6.4). Once the mobile device is targeted at an external display, the remote layer shows content as well.

The amount of items shown in the local layer influences the visibility of the remote layer: the more personal items are shown, the less portions of the remote layer can be seen. However, the visibility of the remote layer is the crucial part of *through the display* interaction. Several solutions exist that still allow the visibility of remote displays: first, the local view could be switched off temporarily once a remote display has been detected in the viewfinder. While this solution would allow for full visibility of the external display in the live video image, it limits potentially needed cross-display interactions which require both layers to be visible. Another option is to move local items towards the edges of the mobile device's canvas in order to free screen real-estate in the device's center. The empty display space can now be used to show the remote layer. Although this region would be fully viewable, the region is rather small for interaction. Furthermore, the interactivity on the local layer is also limited due to either dense item layouts or smaller item sizes. Both solutions are therefore not ideal candidates that allow for (1) great visibility of the remote layer as well as (2) high interactivity on the local one. The order of layers can also be changed. Ramos et al. present the *Tumbler*, a technique that allows users to reorder layers in a three-dimensional stacked view [RRC$^+$06]. While this system allows for changing a layer's $z$-value, it does not allow for showing both layers at the same time which may be needed for cross-layer operations. They further present the *Splatter*, which distributes stacked items around the occlusion point. They found that the latter technique is faster when accessing occluded content. The occlusion is fixed (i.e., neither the top nor the bottom content moves). Our model in turn allows the background layer and its content respectively to be movable. Hence, the static *Splat* technique is hard to realize.

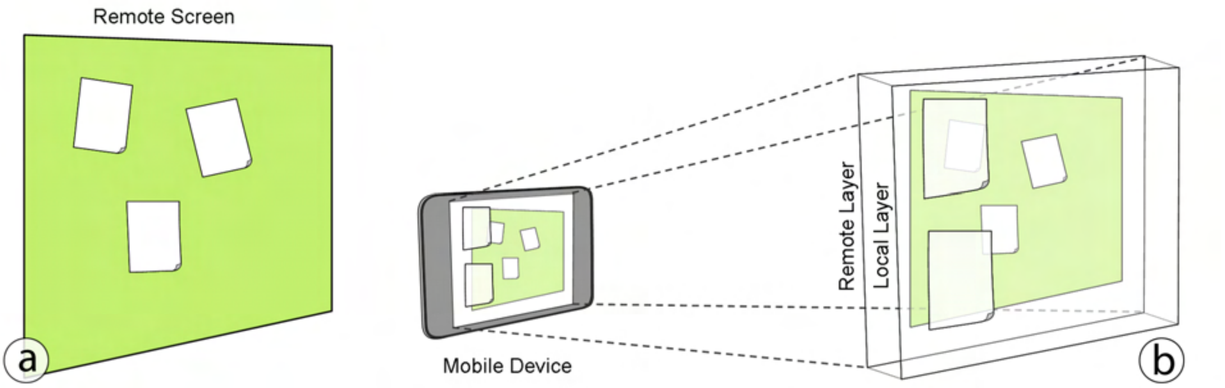


**Figure 6.4:** Two independent content layers: The external display (a) is shown in the background layer with the mobile device's content being superimposed on it (b).

An approach to overcome these limitations is to use alpha transparencies. This principle has been used in regular desktop applications. Ishak et al. demonstrated how semi-transparent windows (or more precisely: the free-space shown in a window) allow users to view and also access information shown in the window located underneath another one [IF04]. This approach seems to be very efficient if large regions of a window are unused. For example, a file browser window may have such spaces where no files are shown. In our approach, we incorporate this idea by the entire background being transparent. We did not apply the context-aware transparency to the items shown on the local layer as their size only allows for a substantially small see-through region. In contrast, we chose to switch to a semi-transparent behavior for all items (i.e., all items are shown with alpha transparency) whenever an external display is shown in the viewfinder. This alpha value is a critical factor: on the one hand, low alpha values (increased opacity) leads to less visibility of the background layer (i.e., the remote content). On the other hand, high alpha values decrease the visibility of personal items. In our first prototype (see section 6.5), we chose an alpha value of 50% as a good trade-off.

## 6.3.2 Personal Feedback on the Mobile Device

Besides showing local content, the personal screen can be used to display individual feedback linked to public content on the external display during an interaction. For example, a remote item may be augmented with possible options. Similar to the sequential approach presented in chapter 5, the list of possible commands is then shown next to the remote item the user has tapped. Furthermore, individual visual feedback may indicate a successful operation such as selecting an item. Naturally, the remote display can also show all these visual aids. However, feedback on a public display should only be given if it does not affect the users' privacy [BKN$^+$05]. For example, others in the environment should not be aware of (1) the content a user selects or (2) the options a user chooses for a specific item. This also holds true for immediate feedback such as a confirmation for a certain action. Especially if sensitive data is entered (e.g., a password), highlighting the pressed key on the public display is rather unwanted. Such sensitive feedback needs to be given directly on the mobile device's display without unveiling it to others.

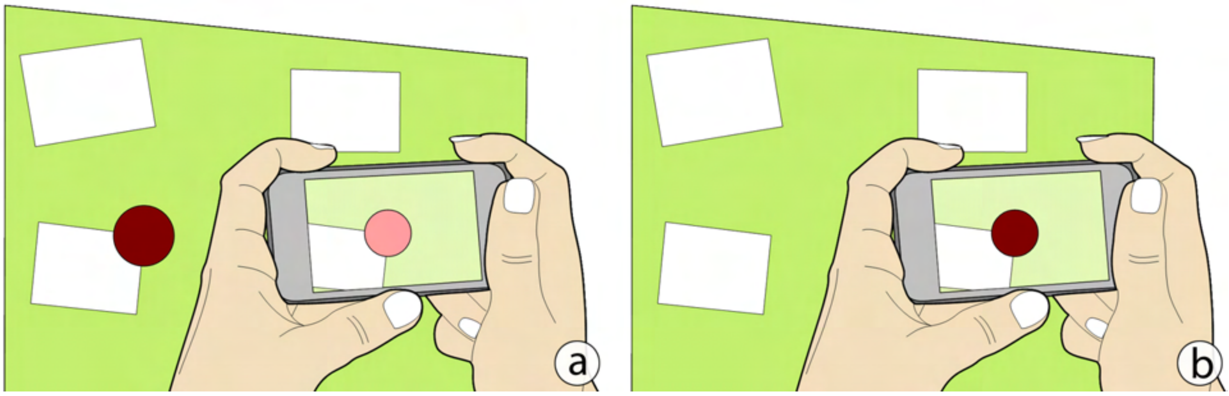


**Figure 6.5:** Placing feedback on the external display or on the mobile device: (a) The feedback is placed on the external screen and therefore also shown on the mobile display. (b) The feedback item is entirely shown on the mobile device only and does not interfere with other interactions on the external screen.

If the type of feedback does not influence on the privacy of the respective user, the feedback can be shown on the public display (see figure 6.5a). However, displayed feedback requires a certain amount of screen real-estate. As discussed in chapter 4, the feedback then needs to be large enough to allow the invoking user to properly interact with it. Our approach allows for obtaining the distance between users and a display. The larger the distance between the user and the external display, the larger the size of a feedback item (e.g., an option) needs to be. Especially when multiple users interact simultaneously, the aforementioned problem multiplies. On the other hand, when users are working collaboratively on an external display, feedback concerning multiple users should be given on the public screen. There are two considerations for feedback: first, personal and individual feedback (e.g., confirmations) is shown on the mobile device (see figure 6.5b). And second, feedback important to the group interacting collaboratively is shown on the public display. With our approach of *through the display* interaction allows for the same visual effect for users regardless of which strategy is used: they always see the feedback on the display of the personal device – either placed there directly or through the live video. As a side effect, the option of ”seeing” feedback on the personal device avoids possible *macro attention shifts* which occurred in the *MobileVue* prototype described in chapter 4.

The location of feedback is another important factor (see figure 6.6). On the one hand, the feedback has a fixed position according to the location of the item it belongs to. We refer to this placement as *content-centric feedback*. On the other hand, it may be fixed with respect to the mobile device which causes it to move with the device. This type of placement can be called *device-centric feedback*. Without any modifications, feedback given on the public display is usually content-centric, whereas feedback shown on the mobile device is device-centric. There may be occasions in which these tight associations are not suitable. The aforementioned lists of options regarding items on the external display can be seen as content-centric feedback. Showing them on the remote screen does not require any changes. If only options for certain items (i.e., the ones the user points at) should be shown, the feedback needs to be given on the personal device which then does not influence others during the interaction. This then requires the feedback on the mobile device to move with the content in the viewfinder accordingly. Another case for content-centric feedback on the mobile device is to indicate whether a user can actually interact with an item. This may not be the case if another user already manipulates it. In tabletop applications this is solved by social protocols and the visibility of someone's actions respectively. When interacting *through the display* users do not see with which content others are currently working. Indicating the item's unavailability by overlaying it with an icon is necessary to avoid user frustration.

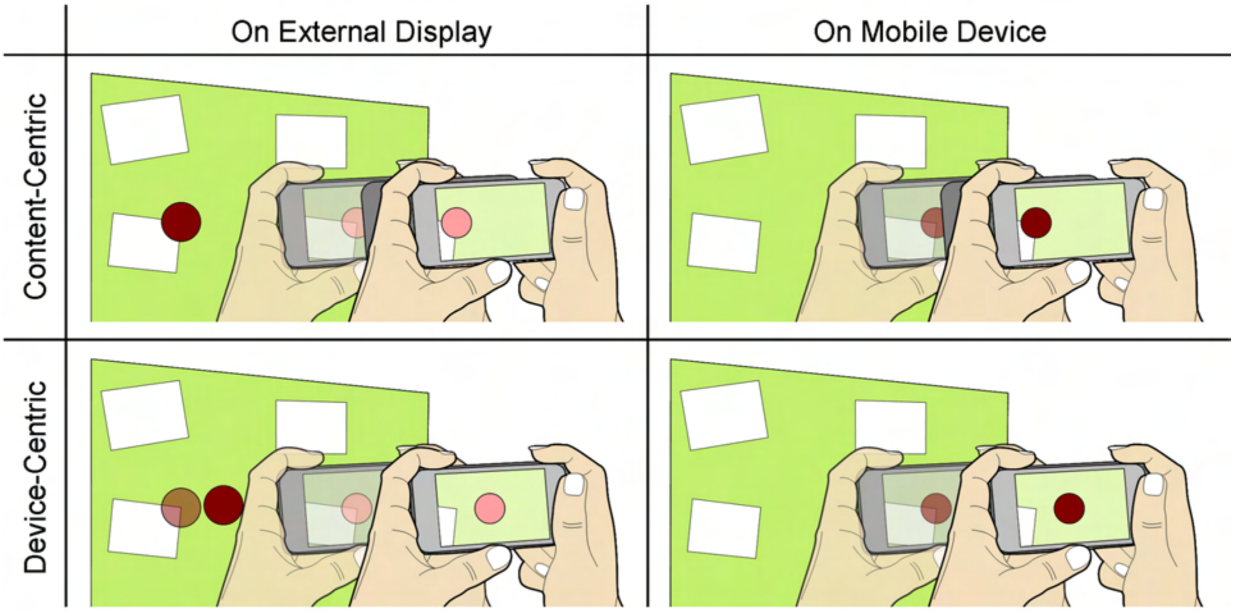


**Figure 6.6:** *Content-centric* versus *device-centric* feedback movement: The left column denotes feedback shown on the external display. In case of *device-centric* movement, the feedback item needs to move on the external screen according to the mobile device. The right column depicts the placement on the mobile device. If feedback moves in a *content-centric* way, the environment manager needs to update its position on the mobile display according to the mobile device's movement.

Similarly to independent content shown on the personal device, the feedback occludes regions of the viewfinder limiting the visibility of the external screen. Naturally, alpha transparencies as described before overcome this limitation. If the feedback is only given temporarily (e.g., as confirmation) and does not hinder the interaction with other items as this is a sequential task. Showing the feedback items with transparencies is then unnecessary. On the other hand, feedback can also be permanent when for example used as a list of additional options bound to a content item. In this case, the feedback occludes certain screen regions. In such cases alpha transparencies should be used as described before. The type of feedback given as well as its visual representation needs to be determined up-front and further associated to the content.

## 6.4 Interacting with Local and Remote Content

When the mobile device's display shows content in addition to the external screen shown in the viewfinder, users need to be able to interact in both layers. In contrast to the model presented in previous chapter, input is not necessarily being transferred to the target display's surface at all times. For example, when users want to interact with local content while an external display is shown in the background, the touch event should be processed locally. Otherwise, the touch event would influence the target display which may not be the desired action. In this section we first present how local and remote input can be distinguished. We then describe the feedback handling in our architecture. As both displays may show their own content independently, we further describe how content can be transferred between both layers.

### 6.4.1 Distinguishing Local and Remote Input

As mentioned before, an input event on the mobile screen can now have two different meanings: first, the user wants to interact with a local item. And second, the user wants to interact with remote content shown in the mobile device's viewfinder. Whenever a tap occurs, the mobile device first needs to analyze whether an input event was performed on a local item or not. If the event can be used on a local item (e.g., the user tapped on an item shown locally), the mobile device directly performs the event without any further actions. If the event cannot be used locally, the mobile device can assume that it is an input event for the remote content layer (see figure 6.7). It sends the input point to the environment manager as described before. The manager can then transform the location of the event into the target display's coordinate system and forward it to the corresponding display. In the case that the point cannot be used on the target display either, the input event will be discarded.

In our model, we want to achieve continuous input. Naturally, users should be able to rearrange local content shown on the mobile device by dragging it from one position to another. This means, that each input event occurring on the local layer needs to be tested in the aforementioned way before it can be processed further. This is important if items listen to and react on *hover* events. For example, a user is dragging an item on the local layer while remote items are shown in the background. When the user then drags the item over a remote item that listens to such events, unwanted feedback may be given. An alternative to perform the test for each input event is to restrict any input to one layer only. To do so, an identifier is assigned to each event that describes the start of an operation (i.e., a *down* event). This unique identifier is then also associated to the layer it occurred which can be detected with the previously described test. Subsequent events with the same identifier then do not need to be tested.

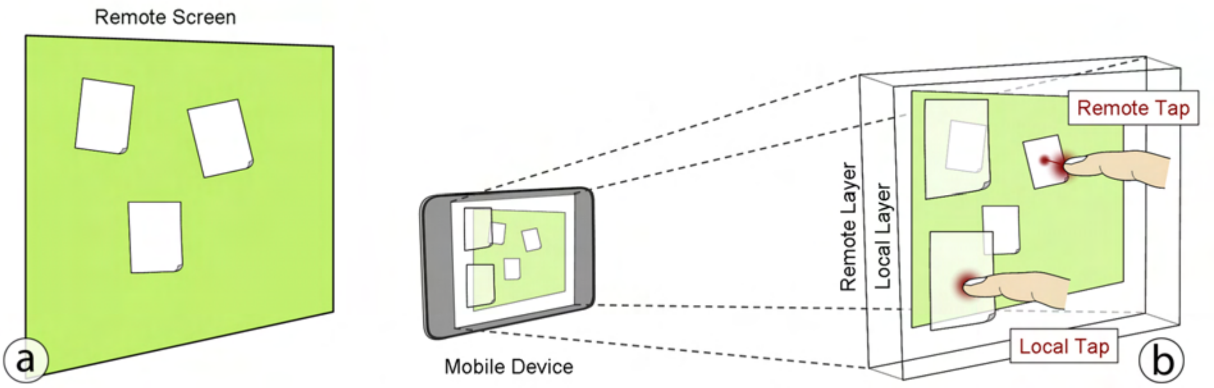


**Figure 6.7:** Distinguishing local and remote input events: When the user points the mobile device at a remote screen it shows both local and remote items (a). The items are shown on two distinct layers with the local one in the foreground and the remote one in the background (b). The received input is first processed on the local layer and then – if no item was hit – sent to the remote layer (i.e., to the remote screen) for further input handling.

The *z*-order of items also needs to be changed by bringing the selected item to the topmost position. With the described input model, this is done on the corresponding layer. The local layer is not affected by this decision as it is the topmost layer. If an item is brought to front on this layer, the item is the topmost one. However, selecting an item on the remote layer only causes it to be the topmost on the external display. This means that it will still be shown underneath local content items when, for example, the device moves. Showing it in front of local items would require transferring it first to the mobile device. Subsequently, the mobile device can render it on top of all other items. When multiple users interact simultaneously this approach does not work. If the item is transferred to the user that selects it, no other user will be able to see it anymore. We decided to keep the output logically separated in two layers (in which items can be rearranged) in the same way as the input seems to be an appropriate choice.

While this approach works fine for manipulating single items, selecting multiple ones by, for example, drawing a rubber band around them is a rather difficult case. When users want to select multiple items on the local layer, they may draw a shape around them indicating the selection of all enclosed items. These shapes usually start at a location at which no item resides. If a remote layer is visible in the viewfinder, our simplified input mechanism then projects the point onto it. Selecting multiple items with a rubber band (and an arbitrary shape respectively) is only possible on the remote layer. This can be sidestepped when no external display is shown in the viewfinder. This may be anyways more comfortable for users as they do not have to hold up their device while they want to precisely select local content. If both layers need to be visible for the selection, this approach fails. An alternative is to allow users to switch the active layer (i.e., the layer on which all input is performed) by pressing a button [LHGL05]. They can then interact freely in the active layer while the secondary layer remains non-interactive. As our envisioned prototypes do not need active and passive layers, we decided to use the first approach.

### 6.4.2 Handling Local and Remote Content

When mobile devices bring their own content, the environment manager does not need to be aware of the content items or their layout in particular. This is due to the fact that each layer handled the input on its own. A dragging operation on the local layer can be performed entirely by the mobile device. Visual feedback on the remote layer is then handled by the respective target display and the environment manager respectively. Nevertheless, if a remote display changes its visual content, users are able to observe these changes in the live video shown on their mobile device. If the environment manager would also handle input for the local layer on the personal device, it would need to be aware of the content and its interaction possibilities as well. This then requires the mobile device to transfer its content directly after the connection. This in turn may consume a larger amount of time necessary for the transmission. Furthermore, the battery life of the mobile device would be affected. The more independent both devices (i.e., the mobile device and a target display), the higher performance can be achieved.

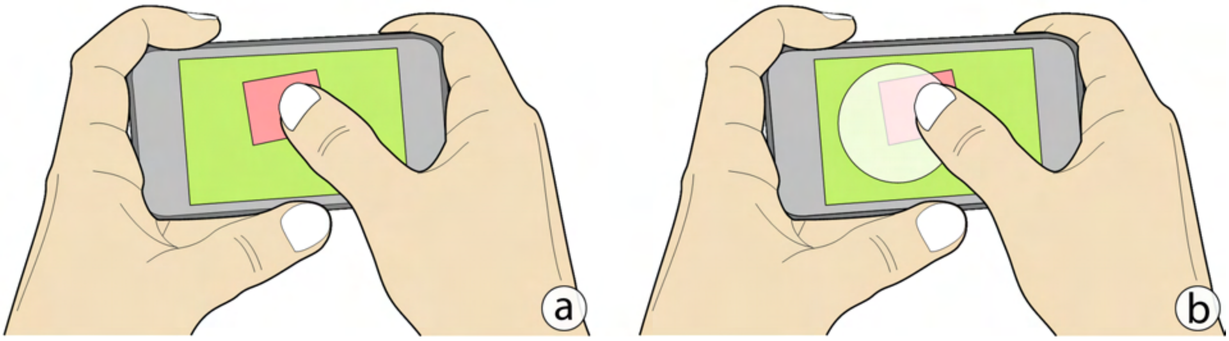


**Figure 6.8:** Distinguishing a local tap from a tap on temporary feedback: (a) The user is touching a local item. (b) The local content is superimposed with temporary feedback causing the same tap to be communicated to the environment manager for further actions.

Similarly to the environment manager not being aware of personal content, the mobile device does not know the remote content either. The reasons are similar to the aforementioned ones: transferring all the content from all possible target displays in the environment would cost time and further consume the battery. However, not knowing the content influences the feedback on the mobile device: first, the environment manager needs to notify the mobile device if feedback needs to be shown. The managing instance also has to send the visual representation. Second, if the temporary feedback is content-centric, the environment manager further needs to update its position on the mobile device whenever a new video frame has arrived. In both cases, the mobile device is responsible for rendering the content on its local layer. And third, the interaction with feedback on the local layer can either be handled by the mobile device or by the environment manager. If the mobile device handles the interaction it needs to know all possible options. In contrast, the environment manager already knows all possible forms of interaction with the specific feedback. Besides testing for local items, the mobile device also tests whether the user hit a feedback item. These temporary items have priority and are being tested first to avoid an event to be performed on a local item shown underneath temporary feedback (see figure 6.8). If a feedback item has been hit, the mobile device notifies the environment manager which then performs the corresponding actions (including potential updates for the mobile device). The flow of input events for different items is denoted in figure 6.9.

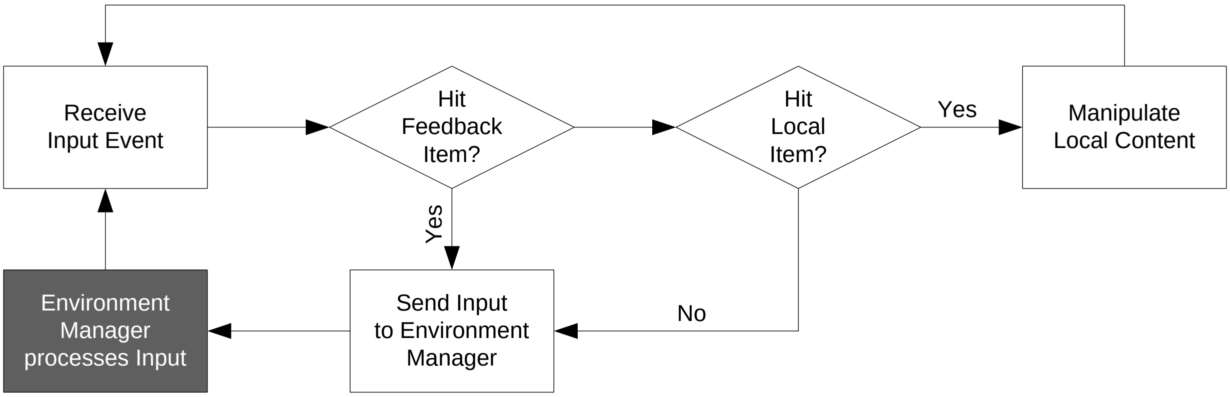


**Figure 6.9:** Input handling for local and remote content: The shaded area denotes the actions taken by the *environment manager*.

In this approach, the mobile device acts as a thin client only carrying its own local content. Each device is responsible for its own input handling as described before. Furthermore, no display needs to know the content from another one (including the mobile device's screen) for the proposed interaction. This allows for an interaction approach using a *non-modal* connection mechanism without the need for downloading the content up-front. Having a local as well as a public layer filled with content may raise the need for transferring items between these two layers. More precisely, users may want to place content on public displays or store public content on their mobile device. In the next section, we will address this scenario.

## 6.4.3 Transferring Content Back and Forth

Local as well as remote content being present may raise the need for transferring content between both displays. One obvious solution to address this is to use a gateway on the mobile display [GFG$^+$09]. Users can then drag local items into the gateway in order to initiate the transfer. This technique, however, has its limitation: the content's placement on the target display cannot be determined. The mobile device's location with respect to the target display needs to be taken into account. For example, users could move the item towards the mobile device's edge. Once the item passes this edge, it would drop to the corresponding position on the target display. This interaction technique only works in one way (i.e., from the mobile device to external displays). Nacenta et al. call such techniques *one-sided* [NGAS09].

In contrast, our system should allow a *two-sided* technique which allows the transfer in both ways with the same interaction style. To do so, we decided to use a single tap to indicate and further initiate the content transfer. If an item resides on the local layer (i.e., a local tap is then performed), the item is moved to the remote layer (see figure 6.10). Vice versa, a remote tap transfers content from the external display to the mobile device (see figure 6.11). Naturally, the speed of the content transfer depends on the size of the content in bytes. For example, moving a video from the external display to the mobile device would require several seconds. We wanted to allow for immediate feedback for users to inform them that a content transfer is in progress. We decided to move the visual representation of the content immediately to the other display. This representation is usually a small image (i.e., a thumbnail) which is transferred nearly in real-time. The actual content (e.g., a video associated to the icon) is then asynchronously transferred in the background. The transfer also needs to be visualized to give an understanding to the user when the transfer is finished completely. This is indicated with a loading bar on the item's lower border.

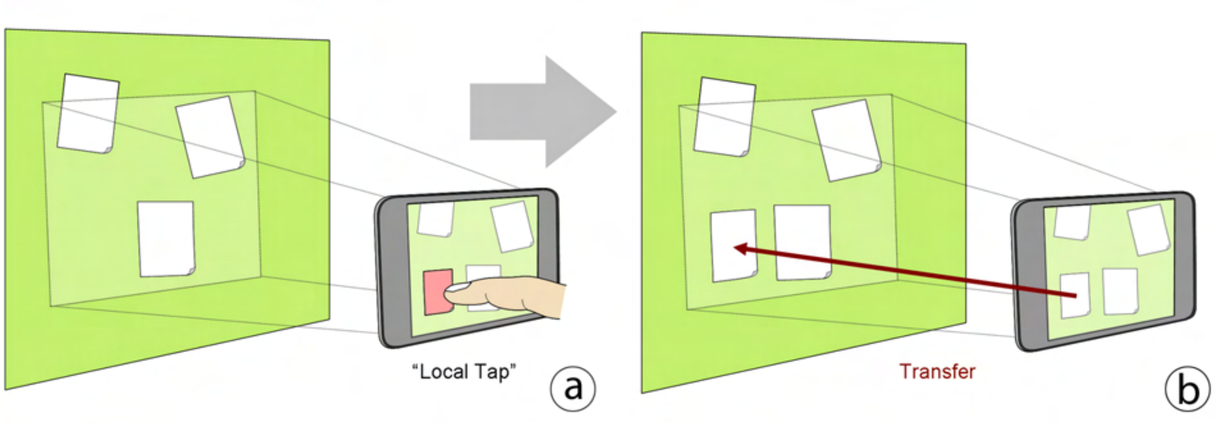


**Figure 6.10:** Transferring content to an external display: When the user taps a local item (a) it is transferred to the external display (b). Red items denote local content.

While transferring content with a single tap is a fast process that works in both ways, this single tap causes another problem. Selecting an item, for example, to open its associated content would also be done by tapping on the item. This, of course, is not a problem if transferring is the only option for content. Furthermore, the selection of local content can be realized with a single tap as soon as no external display is shown in the background. With this, the selection of remote content is possible in a two-stage process: first, users transfer the item to their local display, move their mobile device away from the external display and subsequently select it using a single tap. Another solution is to use marking menus which allow choosing from different options once an item has been selected [KB94]. In case of remote content, the menu could also be shown on the mobile device to reduce visual clutter on the external screen. Certain gestures can also be used to indicate a transfer. This would then work in the same way as using a tap. These gestures can further be applied to both layers which would enable a *two-sided* technique. However, they need to be learned and further memorized by users. In addition, the system has to recognize and distinguish multiple gestures to allow an interaction without errors. As we are interested in the content transfer itself, we chose to use the single tap to demonstrate the content transfer rather than more complex methods in our prototype.

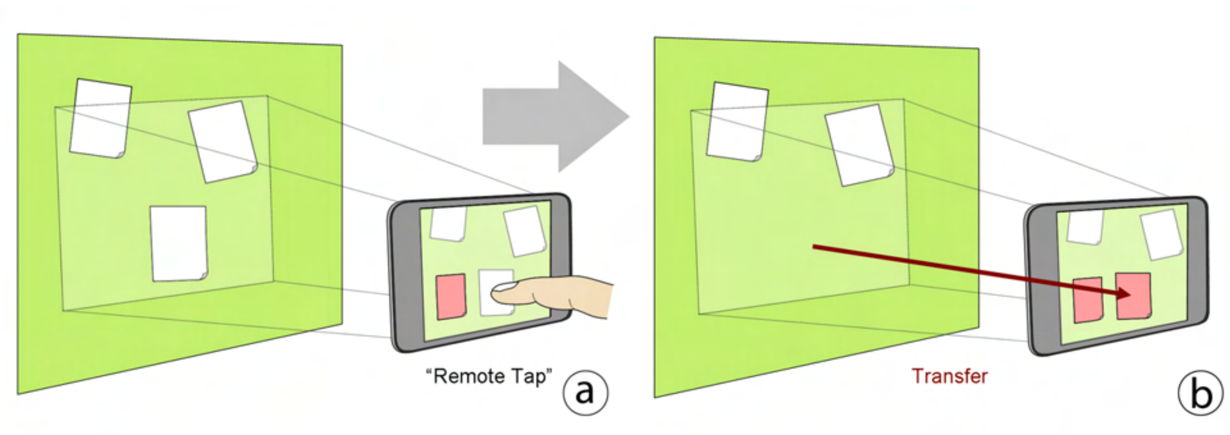


**Figure 6.11:** Transferring content to the mobile device: When the user taps a remote item (a) it is transferred to the mobile device (b). Red items denote local content.

# 6.5 Proof-of-Concept Applications

In order to test our concepts for personal feedback as well as content transfer, we implemented two applications. The first prototype allows users to enter sensitive information (i.e., a personal identification number) through a keypad shown on an external display. Users are able to tap on the number shown in the viewfinder which then results in personal feedback without affecting the public display's content. In the second application, we realized a digital public billboard which can be used to post and retrieve information. The content transfer is initiated using a single tap as described previously. In this prototype, we paid close attention to *moving* versus *copying* content. In the following we first describe the implementation details, followed by our two applications allowing for (1) individual feedback on the personal screen and (2) content transfer between the mobile device and the respective target display.

## 6.5.1 Implementation Details

We implemented our prototype on a mobile device with multi-touch capabilities featuring a 3 megapixel camera (here: Apple iPhone 3G). The mobile display has a resolution of 320 × 480 pixels and a diagonal of 3.5 inches. Capturing the images with the built-in camera functionality resulted in one frame per two seconds which naturally is not sufficient for real-time interaction. In addition the transmission time is noticeably high when images have 3 megapixels. We chose to access the camera data from the live video preview directly. The images coming from this live preview have a resolution of 320 × 426 pixels which allows for faster transmissions when additionally compressed. The entire code running on the mobile device has been written in Objective-C. Some image manipulation functions were further implemented in native C. The external displays as well as the environment manager have been written in C#. While each screen application runs on a computer driving the display, we chose to host the environment manager on a dedicated machine. This is due to the image processing using nearly all computational resources which in turn does not allow for other applications to run smoothly. The communication between the displays and the environment manager has been realized using wired networks. The communication between the mobile device and the environment manager relies on wireless LAN for faster image transmission and greater reliability.

The environment manager runs on a machine featuring a 3.0 GHz Core Duo processing unit. This allows us to analyze incoming frames with 20 frames per second. However, the iPhone used in our prototype has a limited transmission bandwidth that only allows up to 8 uncompressed frames per second. We tested with different compression factors and were able to tune it up to 12 frames per second with a low quality JPEG compressor. This in turn then limited the recognition rate due to extremely visible JPEG artifacts. We decided to have medium JPEG quality (50%) and were able to perform the compression and the image transmission with 10 frames per second. We believe that future mobile devices (including new iPhone revisions) will overcome these limitations by introducing faster transmission mechanisms.

Our prototype has its greatest limitation in the interaction distance. The mobile device needs to see at least one item fully in its viewfinder to detect the content and the associated screen. With a field of view of 45 degrees, the minimum distance is about 1.5 times the item's diagonal (measured in physical units). Having ultra-wide angle lenses, the minimum distance can be reduced further. On the other hand, the maximum distance is ten times the item's diagonal between the mobile device and the item itself. Future devices with higher camera resolutions could increase this distance substantially but need a higher transmission bandwidth. The last limiting factor is the current interaction speed. As the iPhone's camera is particularly susceptible to motion blur, moving the device faster than 50 pixels per frame impacts the recognition rate noticeably. Again, we assume that future devices with better cameras (e.g., better sensors, faster shutters) will address this in part. Overall, these limitations are introduced by the current stage of the prototype but not by the interaction concept itself.

### 6.5.2 Entering Private Information on External Displays

Entering sensitive data such as a personal identification number (PIN) is nowadays accomplished on external devices (e.g., ATMs). The numbered keypads present a security threat as potential attackers may have manipulated them to obtain a user's PIN. Furthermore, the fixed nature of the keypads allows for stationary but hidden cameras to capture the PIN a person enters. While touch screens may overcome the first problem their decreasing accuracy and sensitivity over time forces users to even enter the number in a more explicit (and visible) way. This is why users may not trust external devices at all anymore – especially if they have to be used to perform a very sensitive task such as withdrawing money. Although such tasks require a form of token (e.g., a credit or bank card), copying them is easily done by manipulating the card readers on ATMs. On the other hand, using a more private token which is harder to copy (or mimic) may help to increase the security of such external devices.

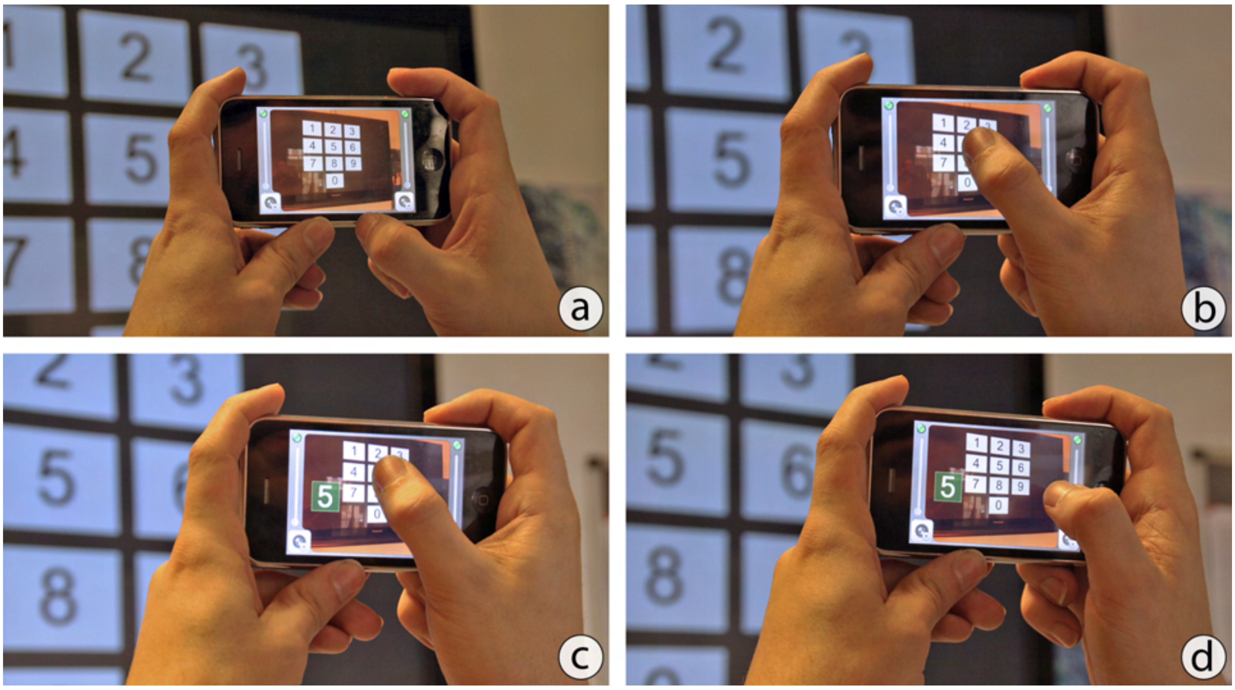


**Figure 6.12:** Showing individual feedback on the mobile device: The user points the device at the target display featuring a virtual keypad (a) and taps on a number (b). The mobile device shows the pressed number only to the respective user (c) for a certain amount of time even when the user is not tapping the number anymore (d).

Several research projects see the users' mobile devices as such tokens. Mobile devices can be used to tell users what to do as the next step while still entering the number on an external keypad. For example, using a short vibration could indicate that users have to enter the next number incorrectly on purpose [DLvZH09]. The mobile device's keypad can also be used for entering the sequence of numbers [DLFBH09]. In both usage scenarios a clear connection has to be established up-front between the mobile device and the target display since the external screen also determines the physical point of releasing the money. This can be done by using markers which users have to take a picture of. These markers then include the connection address. Nevertheless, such combinations separate this type of interaction into two distinct phases.

Instead of interacting with the mobile device's keypad or an external one, our model allows to interact locally on a virtual keypad shown on the target display. Similarly to the aforementioned systems, the external display is still needed as reference point (i.e., the user has to aim the mobile device at it). This especially holds true in environments that contain multiple ATMs such as a foyer of a larger bank. In contrast to the other systems, however, interacting *through the display* now allows to enter sensitive information immediately without the need for a *modal* connection mechanism. An ATM's display still has to contain a kind of marker for identification which may be either the background image or the layout of the numbers. Such unobtrusive markers may be preferred over fiducial ones by providers for to aesthetic reasons.

Once users have pointed their personal device on the respective target display, they can start entering their PIN on the mobile display (see figures 6.12a and 6.12b). After a number has been tapped, the environment manager identifies the number (i.e., the item that has been touched). The centralized instance then notifies the mobile device to show feedback to inform its user which number has been tapped (see figure 6.12c). In this scenario, the feedback is naturally given on the mobile device only to ensure the invisibility of sensitive information to others. The environment manager is aware of the user's interaction. The managing instance is required to be a trusted component. The system further shows the tapped number for a short period of time (here: one second) regardless whether the user still presses the number (see figure 6.12d). The confirmation can be presented to the user in two ways: first, the remote number can be highlighted directly on the mobile device's display. And second, the number may be shown with an offset next to the finger tip to allow for better visibility. In our scenario, we chose to use the latter approach. We admit that this may potentially increase the risk of others observing the PIN. On the other hand, when highlighting the number directly, we found that it was hardly noticed by users. Furthermore, if attackers are able to see the PIN, they still need to get the mobile device since it may act as the token (i.e., its unique ID). Despite these potential limitations, the described application demonstrates the use of personal feedback when interacting with external displays.

### 6.5.3 Sharing Information with other Users

In the past, bill-boards have been used to post information that may be of interest for others such as job offers or general announcements. Similar to the usage model of public displays (i.e., information broadcaster), content on these bill-boards is not meant to be taken away. Users have to write down the important information such as an email address or a telephone number. Sometimes, such contact data is replicated on several paper straps which can be torn off by people interested in the offer. Nevertheless, these straps are limited only allowing a certain number of people to retrieve this information. To overcome this limitation, users may also take a photograph of a note posted on a bill-board. This picture can then later be reviewed and important information is not lost. Nevertheless, while retrieving content can be done in various ways, posting such remains unsolved. Besides these traditional bill-boards, their digital counterparts (i.e., large public displays) suffer from the exact same problems.

In chapter 5 we presented a prototype that allows for retrieving remote content by taking a picture of the content on the external display. Combining it with the touch-based, bimanual approach presented in this chapter, users are able to copy information from the external display onto their mobile device with a single tap. To do so, they have to touch and release a remote item shown in the viewfinder. The item is then transferred to the mobile device in the aforementioned fashion. Assuming that the mobile device also carries local content, the transfer back can be done in the same way (see figure 6.13). However, tapped content may be *moved* (i.e., removed from the originating device) or *copied* (i.e., a full copy is transferred). On the one hand, the technique could only support moving content without the option of fully copying information. When content is moved, the original idea of simulating a bill-board fails since users select information of their interest which is then not accessible for others anymore. The technique has to support copying content as well. On the other hand, permanently copying content may flood an external display with multiple instances of the same data when users constantly transfer the same item back and forth. For this reason, moving and copying needs to be enabled and differentiated by the technique to avoid limitations occurring when it only supports one transfer mechanism.

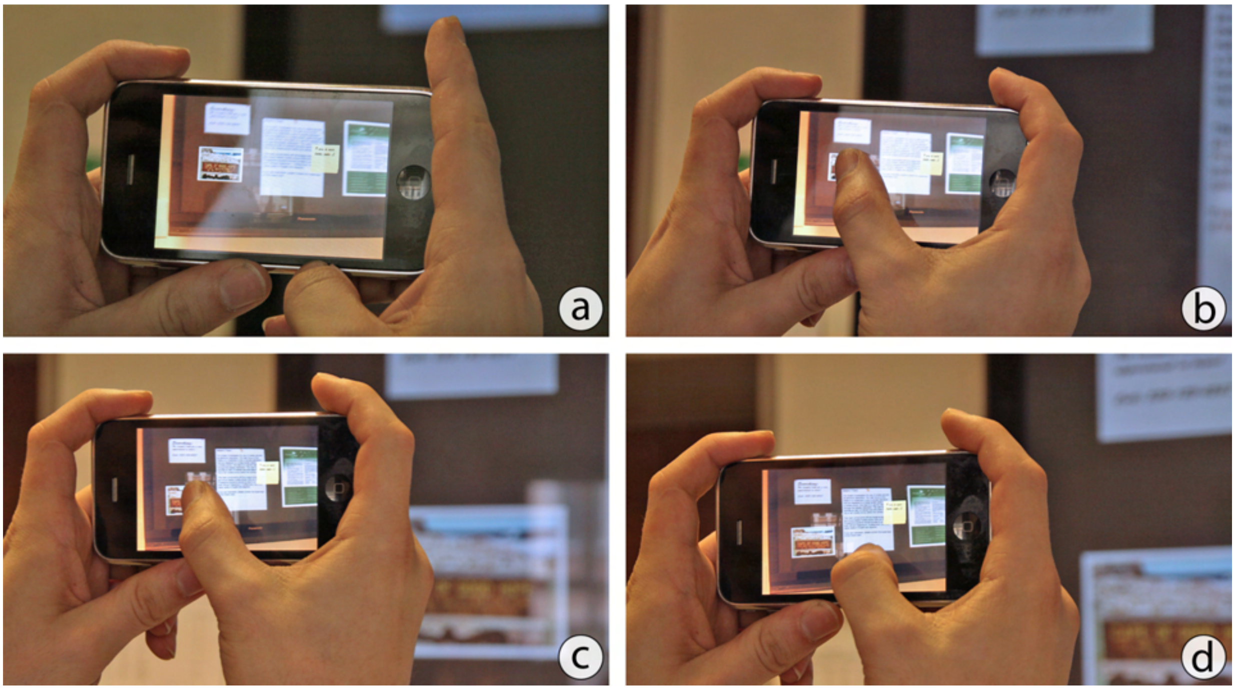


**Figure 6.13:** Transferring content from the mobile device to an external screen: The user points the device at the target display (a) and taps on a local content item (b). The tap initiates the transfer of the selected content (c) which is *moved* onto the target display (d).

The rights of moving and copying content can be described by using the concept of ownership. All items that reside on the mobile device during the application launch are owned by the respective user. If users create an announcement, they can place it on a public display in the aforementioned way. In contrast to personal items, those that are shown on a public screen belong to the public. Thus, users can copy them on their mobile device if they are not the *creator* of the content. The person that created the item and placed it on the external display is the only one that can remove it. Deleting an item from a public display (if applicable) is done by *moving* it back to the owner's mobile device with a tap. When users *copied* an item on their mobile device, they can also place it on another display. This is only possible if the display does not hold a copy of the respective item. If such an item is already present, the content is *deleted* from the user's mobile device only. The only point in which the system cannot determine whether a user is allowed to *move* or *copy* an item is the initial transfer from the owner's mobile device to the external display. In this case, the owner may want only to create a copy on the external screen and put the same item on other displays in the environment. We decided to use the *copy* mode for the owner for the initial placement. We did not intend to explicitly allow for deleting items as we assume this to be performed using functions from the mobile device's operating system.

## 6.6 Summary and Discussion

In this chapter we have presented an extension of our model which allows for real-time interaction in a bimanual fashion. We described how our tracking solution needs to be refined in order to allow for real-time feedback. Our solution is still purely based on the live camera preview. We extended the uni-manual approach featuring a single point only with a bimanual multi-touch approach. In this, the mobile device (i.e., the reference frame) is moved coarsely with the non-dominant hand. The dominant hand then interacts within the defined reference frame. We further showed how the personal and remote display can be used in combination. We described different scenarios each of which makes use of the personal display for (1) individual feedback that is not of interest (or not wanted) for the public and (2) personal content items that may reside on the mobile device. We further showed how our model allows for interacting with local as well as remote content based on two layers. The input is processed on the local layer and, if not already consumed, transferred to the display in the viewfinder. We demonstrated the functionality in two prototypes: In the first application, users can enter a PIN using a numbered keypad shown on the external screen. In the second application, the external display was employed as a public digital bill-board. Users were able to post information as well as retrieve content. The idea was to allow content transfer (i.e., switching from the current layer to another) by tapping on the respective item. We further presented a simple rights management in this prototype.

When the mobile device is used to show individual feedback, the environment manager is responsible for both the visual representation as well as its location on the mobile device's display. The location can be either *content-centric* (i.e., fixed to the remote content's position in the viewfinder) or *device-centric* (i.e., fixed on the mobile device's display). The interaction with temporary feedback, however, needs to be recognized by the mobile device with higher priority. In contrast to temporary feedback, the mobile device is responsible for showing its own content while the environment manager is not aware of it unless the content is transferred. If personal content is displayed on the mobile device, this personal layer needs to have a certain transparency to still allow for the visibility of remote displays shown in the viewfinder. This semi-transparent view only needs to be present when an external display is shown in the background. Otherwise, local content can be shown without alpha-transparency.

The layered approach allows for a distinguishing between local and remote input. The mobile device tests each input point first with potentially shown feedback items. If a hit is detected, the event is transferred to the environment manager which takes further actions. Otherwise, the mobile device tests whether the input occurred on any of the local items. If so, the input event is consumed on the device directly without notifying the centralized manager. Only if no feedback item or local content has been hit, the input point is sent to the environment manager. During all times, subsequent input events occurring from the same source (i.e., the same finger) remain on the layer they have been detected first. This approach, however, only allows directly interacting with the content. The selection of multiple items by using, for example, a rubber-band is either performed on the remote layer (if a display is shown in the background) or on the local layer (if the device is not pointed at a display). Nevertheless, this only is a limitation when both layers are visible while a user wants to select multiple local items.

Combined with chapter 5, we have presented the infrastructure necessary to allow for *absolute* and *direct* interaction on external displays with an *non-modal* connection procedure from the user's point of view. We then utilized the mobile display to show content as well, such as temporary feedback or personal content items. Besides interacting on remote displays, the model also allows to transfer information back and forth between an external display and the mobile device. In all prototypes, the *distance* between the mobile device and the target display is still limited due to the mobile device's wide-angle lens. The further users are away from the display, the noticeable smaller the items are in the viewfinder making it hard to interact with them. In the next chapter, we discuss this problem in more detail and present solutions to it.

# Chapter 7

# *Touch Projector*:

## Continuous Interaction at Variable Distances

> *A hypothesis or theory is clear, decisive, and positive, but it is believed by no one but the man who created it. Experimental findings, on the other hand, are messy, inexact things, which are believed by everyone except the man who did that work.*
>
> **– Harlow Shapley –**

In chapters 5 and 6 we presented the concept of *through the display* interaction based on live video on the mobile device. However, all prototypes only allowed for limited distances between the mobile device and the target display. This is due to the wide-angle lenses used in all modern camera-equipped mobile devices. In general it holds true that the further a user is away from the display, the smaller it appears in the viewfinder of the mobile device. Using wide-angle lenses, the problem increases as a short distance may already be too far for precise interactions since items of interest appear rather small. Due to the *fat finger problem* this problem then further increases [VB07]. If pixel-exact placement is needed on the target display, the user has to see the item (and the display respectively) with a reasonable size in the viewfinder. This *apparent size* of an item in the viewfinder in turn is influenced by both the distance to the display as well as the target display's size. In this chapter we[1] demonstrate how *through the display* interaction can be used on different types of displays at varying distances (see figure 7.1). We further show how continuous cross-display operations can be realized on displays featuring different input

[1] The work presented in this section has been published in a scientific paper [BBB$^+$10]. The scientific plural refers to all authors of these publications – namely Sebastian Boring, Dominikus Baur, Andreas Butz, Sean Gustafson, and Patrick Baudisch.

capabilities. We present a series of improvements to the naïve implementation, including zooming and stabilizing the video image. In a user study we found that highest performance can be achieved when selecting targets and dragging targets between displays with automatic zooming and temporarily freezing the live video image.

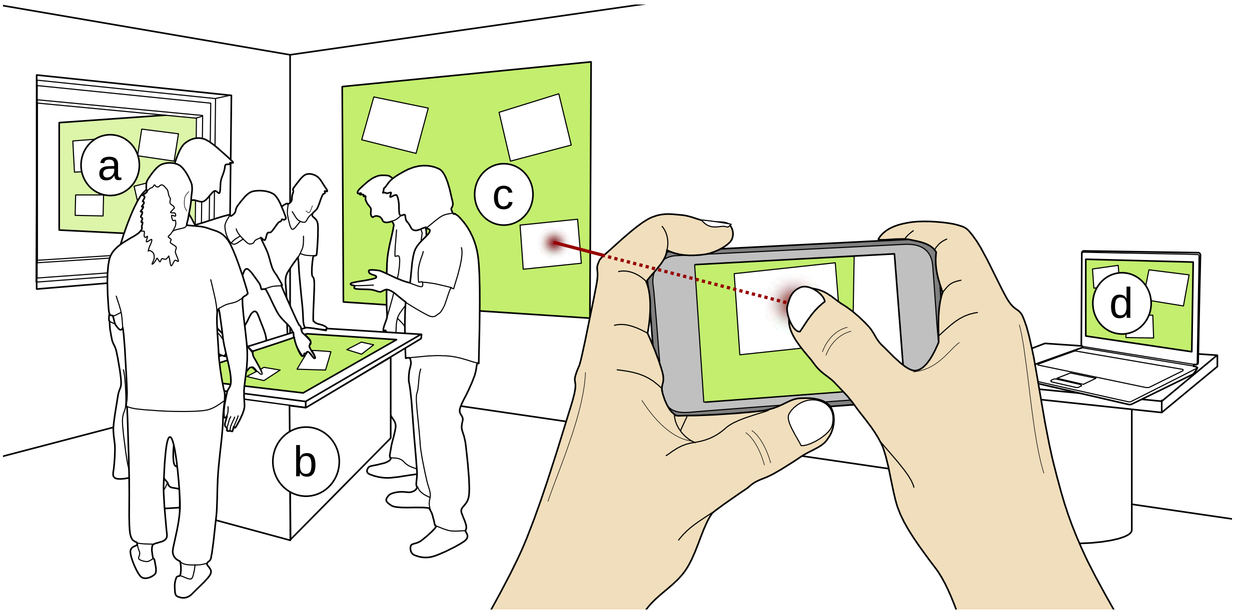


**Figure 7.1:** Reaching distant and unreachable displays, such as (a) displays outside a window or (b) a tabletop crowded with people. The prototype allows users to manipulate devices that are incapable of touch input, such as (c) a wall projection or (d) a laptop. Users manipulate a display's content by touching and dragging objects in live video. The device ”projects” touch input onto the target display, which acts as if it had occurred on itself.

The remainder of this chapter is structured as follows: We present a first (naïve) implementation that allows for continuous input on and across displays in the environment (see section 7.1). Subsequently, we discuss the limitations in terms of the item's *apparent size* shown in the mobile device's viewfinder followed by a presentation of improvements that address this issue (see section 7.2). We then present our thorough evaluation of these improvements (see section 7.3). We conclude this chapter with a summary and discuss the extensions (see section 7.4).

## 7.1 Continuous Input on and Across Displays

In contrast to the previously described prototypes, we decided to realize a system that allows for continuously interacting *through the display*, such as dragging objects on a display. We chose to use continuous interactions since we wanted to investigate potential improvements for exact positioning of items. Similar to earlier systems, users aim their mobile device at the display of interest. The live video then shows the content as if the personal display would be transparent [WFB$^+$07]. When touch input occurs on the mobile device, it is ”projected” onto the exact position on the target display resulting in immediate feedback [Shn83]. The discrete operation of selecting items (i.e., tap them) is realized in exactly the same way as presented in chapter 6. However, we wanted to enable users to drag content on displays and also across screens in the environment in a continuous fashion. In this section we first describe the influence of bimanual input for dragging operations on a display. Subsequently, we discuss the changes necessary to our infrastructure in order to allow for *continuous* cross-display operations. We then present the implementation details and discuss the benefits and limitations of this prototype.

## 7.1.1 Coarse and Precise Dragging Operations

The bimanual nature of our approach allows performing drag operations in three different ways: first, users can directly drag the item shown in the viewfinder by touching it and move their finger (see figure 7.2a). Second, they can touch the item and move the device while keeping the finger steady to bridge larger distances (see figure 7.2b). And third, they can combine both movement techniques to allow coarse and find-grained interaction simultaneously. In the first case (i.e., moving the finger only), it is sufficient to transfer the input whenever a new event occurs on the mobile device. In the second and third case, keeping the finger steady on a position does not lead to new position updates. Whenever a new video frame arrives, the environment manager has to apply the calculated transformation matrix to all active input points. Thus, the managing component has to be aware of all touch points in order to send their transformed representations to the target display. In theory, this is not necessary in the hybrid solution as long as the finger moves on the mobile device. We wanted to allow all solutions and decided to use each video frame for transforming all active touch points. Overall, when dragging an item, our prototype allows for coarsely positioning (i.e., moving the whole device with the non-dominant hand) the reference frame and the item respectively followed by a fine-grained (i.e., moving a finger while keeping the device still) operation to bring the item to its final location.

The advantage of associating each touch with a user is also of great importance in continuous settings. For example, the prominent pinch gesture can be detected by the system in two entirely different scenarios: first, both fingers are from the same person. This indicates a pinch as it is known in today's multi-touch applications. But second, two fingers may come from two different persons. This then would mean that two different users want to drag the same item into different directions. All systems incorporating this simple pinch gesture (with the exception being the DiamondTouch [DL01]) suffer from the missing association between finger and user. Our prototype easily allows this by associating all touch events coming from a single mobile device to one person only. Race conditions as previously described can be resolved on a "first come, first serve" basis. This means that no user can interact with an image as long as another one is manipulating it. However, the person currently being allowed to interact with an item can naturally add a second touch point for scaling purposes as the event comes from the same device.

When users drag an item on the display, the visual content of the external screen is changed. This, however, is the basis of our tracking solution. For this reason, the environment manager needs to be aware of any visual content changes. Each connected display has to send an update whenever a change occurs. These changes can be either of visual (e.g., the contrast of an image changed) or geometrical (e.g., the item is moved to another location) nature. While this seems to be the solution in theory, during the practical realization we found that this is not sufficient. As the camera of the mobile device does not deliver the video images with a high frame rate, the content may already be on another location than captured in a given frame. The calculation of the transformation matrix would lead to unexpected point translations. As items that are currently active (e.g., in a dragging state) may lead to wrong transformations, we excluded them (i.e., each item that has at least one touch point on it) from the analysis. We acknowledge that this does not scale for multiple users when each user manipulates a different item.

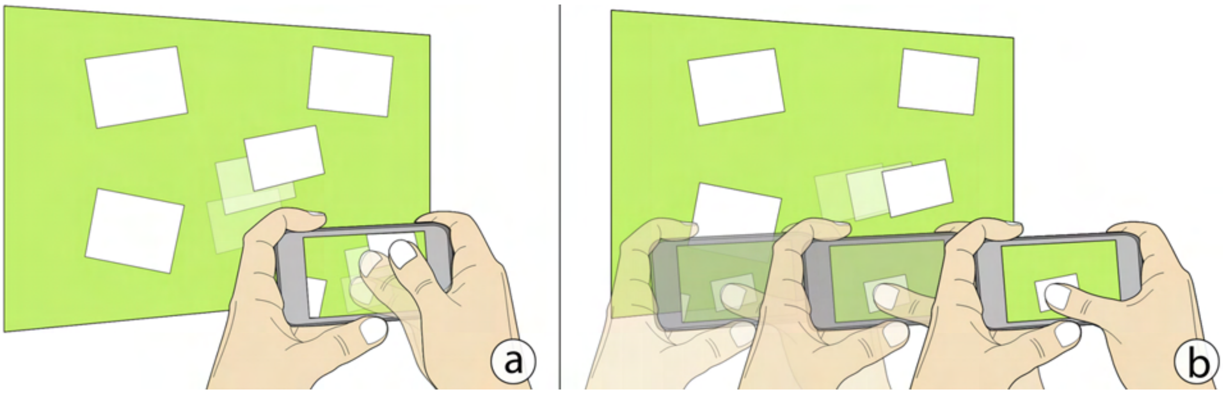


**Figure 7.2:** Moving a finger versus moving the device for dragging content. (a) Moving the finger while keeping the device steady. (b) Performing a touch-and-hold operation and moving the device with the finger remaining at its location.

There is another layout of items that may harm the recognition of our tracking. If two items overlap, the detected blob usually represents the bounding box around both items. While this was no issue in the first prototype (see chapter 5), allowing users to freely drag items on the display may lead to such situations. Similarly to the problem of active items, each occluded item can be excluded from the detection process. This then leads to the same problems as mentioned before. Another solution is to avoid overlapping programmatically. If all items are considered to be in the same plane, a moving item would force others out of its way based on the law of physics [WIH$^+$08]. While this seems to be a promising approach, it fails if two items that are moving collide. If the laws of physics are applied correctly, both items would then behave differently compared to what the users expect. These issues are true limitations up to this point. Nevertheless, better cameras (i.e., higher resolution and faster shutter speeds) as well as future tracking algorithms may overcome these limitations.

## 7.1.2 Dragging Content across Displays

Besides moving items on one display using touch input, users are also able to drag content across displays. To do so, they touch an item on the originating display and start dragging it as described before. Simultaneously, they move the device to the destination display and can then position the item on its final location. As the mobile device tracks its spatial relationship with respect to the displays (and thus partially to the environment), common problems known from relative pointing in display-less space do not exist [NMG08]. Users decide the path they take on their own which does not introduce inconsistencies between the motor space and the feedback. However, while traversing from one display to another, users do not get feedback with the original idea of having the mobile device as a purely transparent one. This limitation can be sidestepped by showing a thumbnail of the dragged item whenever the device is not pointed at an external display (see figure 7.3). As soon as another display is in sight and shown in the viewfinder respectively, the thumbnail is removed and the item is shown on the corresponding display. The mobile device acts as a display in the environment if needed. Overall, users can always track the dragged item – either on a display shown in the viewfinder or on their personal device (see figure 7.4).

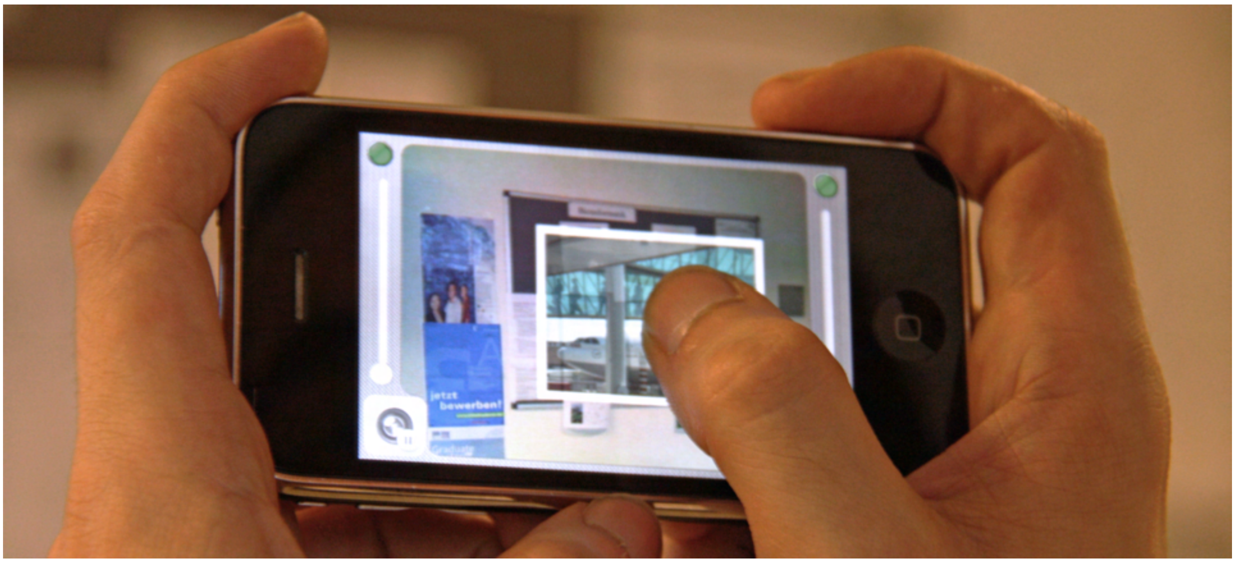


**Figure 7.3:** Visualizing dragged content in display-less space: a thumbnail is shown whenever the mobile device's camera is not pointed at a display in the environment.

To allow for such drag operations, the environment manager needs to know when a touch point that currently drags an item leaves a display. Whenever an event of this type occurs, the manager needs to send the visual representation of the dragged content to the mobile device. At the same time, the content needs to be removed from the display to avoid the content being shown twice. Before transferring the visual representation of an item to the mobile device, it needs to be transformed according to the current device orientation to allow for a seamless transition between external screen and mobile display. This can be done by using the inverse of the current transformation matrix between both devices' image planes. Similarly, when reaching another display, the item's local geometry needs to be translated into the screen's coordinate system. Each corner point is transformed again as described earlier. The content's thumbnail shown on the mobile device can be manipulated while moving the device. For example, users may drag the item on the mobile display towards the device's edges to allow for greater visibility of potential target displays through the viewfinder. To allow the aforementioned translation back into the external display's coordinate system, the environment manager needs to be aware of such manipulations.

We decided to send local touch events to the environment manager allowing the manager to trace the actions while the device is not pointed towards a display. This information allows for a correct transformation of the dragged item once an external screen is in the viewfinder.

Besides handling the visual output, the input also needs to be handled during cross-display operations. In our architecture, touch points are and their corresponding actions respectively are directly performed on the target display. Hence, if a touch point leaves the current target display shown in the viewfinder of the mobile device, the environment manager has to notify the screen as well, so that it can remove it. This point is then temporarily active on the mobile device as long as no display is shown underneath the touch point in the viewfinder. Similarly to the dragged item, the point needs to be removed from the local layer and then added to the respective external screen, when the point enters a display in the live video. Users can then continue to drag directly on the external screen. Figure 7.5 denotes the input and output handling process. Naturally, if a second touch point is added or removed on the mobile device while the input is performed locally (i.e., no display is shown in the viewfinder), the point needs to be added as well when the device reaches another screen. Such events may occur when users want to scale the currently dragged item to fit the destination display (i.e., downsizing it if the screen is smaller than the originating one). In our architecture, the environment manager is aware of input on the mobile device at all times to allow for the aforementioned input handling.

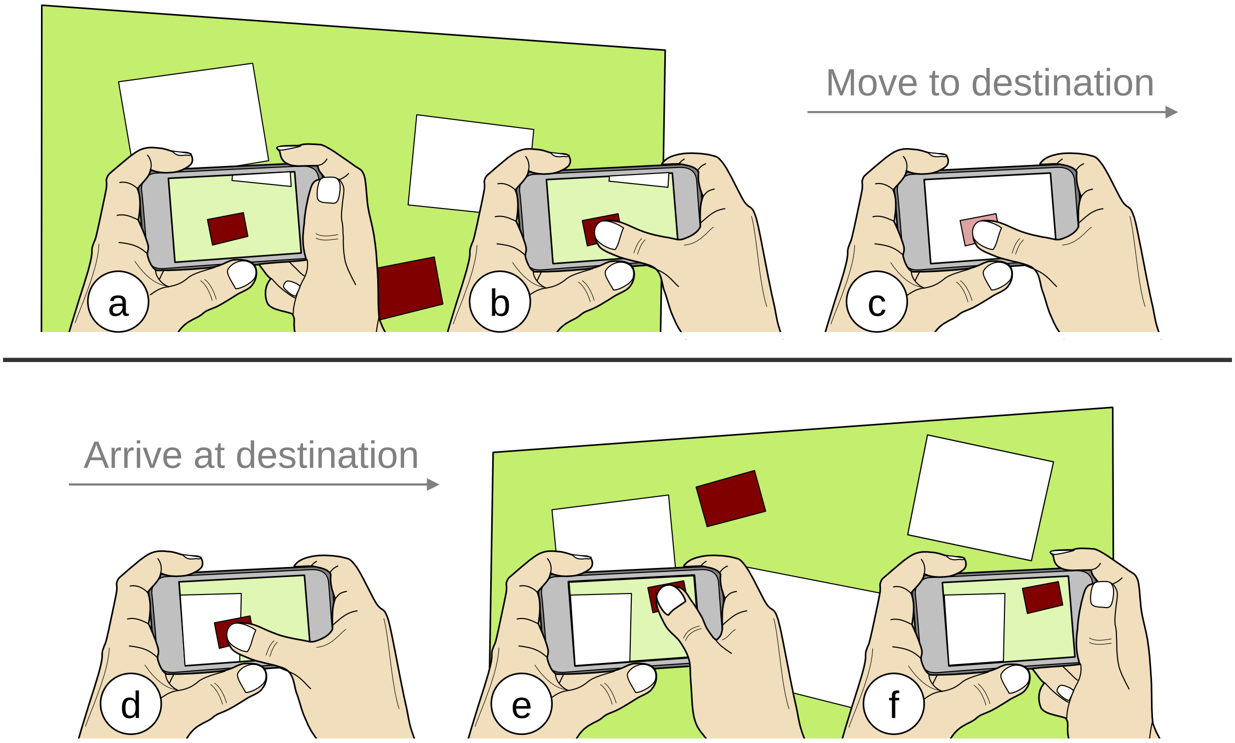


**Figure 7.4:** Dragging content across displays with the original metaphor: The user aims at a display (a) and touches the item of interest (b). When moving the device off-screen, a thumbnail of the item is shown (c). After reaching the destination display (d), the item can be positioned precisely (e). When the finger is released, the item has been transferred (f).

Users can also (accidentally) remove the last touch point when they drag an item in display-less space. The logical operation is then to drop the item at this location. Since we do not track the device with respect to the environment but rather with respect to each display, the item would be lost and is not detectable anymore. In this case, we decided to return the item to its original location before the drag started. To do so, the environment manager stores the necessary information by keeping an exact copy of it. It is then able to return the item (i.e., adding the copy to a display) once it is released in display-less space.

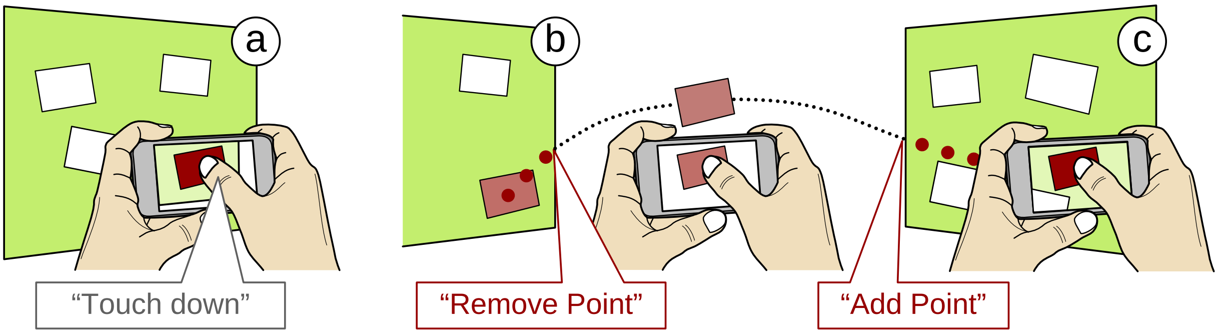


**Figure 7.5:** Touch event handling when interacting across displays: (a) A touch event is projected onto the current screen. (b) Leaving the screen requires a removal of the point. (c) Reaching another screen then requires the manager to add a new touch point to the display.

## 7.1.3 Implementation Details and Performance

In order to test the use of continuous operations and cross-display interactions, we implemented a prototype called *Touch Projector*. The system uses the real-time tracking presented in chapter 6 with the extensions discussed in earlier sections. Furthermore, we decided to use the exact same mobile device (i.e., Apple iPhone 3G) as well as the dedicated machine for hosting the environment manager. We again chose to only use the raw camera data (i.e., 320 x 426 pixels) for the same reasons as described in chapter 6. The code for the mobile device needed slight adjustments in order to allow continuous touch input on and across displays. The external displays in our setup were a laptop (i.e., 1440 × 900 pixels with 15" diagonal), a regular desktop computer (i.e., 1280 × 1024 pixels with 19" diagonal), and a large public display (i.e., 1366 × 768 pixels with 50" diagonal). The implementation of these displays did not change at all since they only had to show content distributed by the environment manager. The content items in our system are represented as pictures (similar to previous prototypes). To enhance the tracking and allow for a more precise interaction we chose to place white borders around each item for increasing the contrast. With this, the maximum distance is now up to 20 times the item's diagonal.

While we were able to allow the recognition of items at larger distances, the previously described limitations from the technical point of view remain. As described before, currently active items (i.e., items that are being manipulated or dragged) cannot be included in the recognition process. The iPhone's susceptibility to motion blur when it is moved faster harms the recognition noticeably. Dragging an item fast from one end of the display onto another my moving the device reduces the stability of the tracking; in the worst case causing the device to loose the target display completely. The environment manager then assumes that the mobile device is currently pointing in display-less space. The item is removed from the current display and a thumbnail is shown and the mobile display. We again suspect that future mobile devices with (1) higher camera resolutions and (2) higher frame rates will overcome this limitation. For now the limitations are only of a technical nature (which most likely will be addressed in the future) and are not introduced by the concept of interacting *through the display* itself.

When dragging content from one display to another, we noticed a recognition delay as soon as the target display has been reached. This is mainly due to our current implementation. When *Touch Projector* ”lost” its target screen, it still uses this display as the primary one for image comparison. The correct display is not the first display in the list of known screens. Although the target display is already visible in the mobile device's viewfinder, it still takes time until the screen is actually recognized. This time depends on (1) the number of displays in the environment and (2) the position of the correct display in the list of all screens. Since the system needs approximately 180 ms to correctly identify each display, the delay is noticeable when more than four displays exist. However, this delay can be reduced with the following steps: first, the system can tell that a display is not the correct one based on image comparison only. The recognition process can already be stopped here resulting in less time. And second, the displays in the list can have a geometrical order (if their locations are known) depending on the distance to the previously detected display. This means that displays that are close to the previous target screen appear more in the beginning in the list of all screens. In nearly all cases this would decrease the delay as one can assume that the next detectable display is within a short distance. This, of course, is not true when the system did not detect a screen for a longer time.

### 7.1.4 Resulting Benefits

The original idea of interacting through video was introduced by Tani et al. in 1992 [TYT$^+$92]. However, their system relies on fixed relationships between the machines to be operated and the camera that produces the live video. Our system in contrast uses the built-in camera of a mobile device which turning it into a see-through device. This then shifts the concept of interaction through video to interaction *through the display*. Compared to the original approach, our prototype offers two main advancements: first, although the camera is mobile, our system allows users to implicitly interact with different displays in the environment and further drag content across displays. And second, the camera's mobility combined with the bimanual approach allows for bridging large distances on and between displays. In the following we will describe these benefits in more detail followed by a review of our system's limitations.

In Tani's setting the environment had to be modeled a priori to allow for correct input transformations between the computer screen and machines in the factory. In contrast, our device tracks itself with respect to displays in the environment purely based on their visual content. This eliminates the need of modeling the environment a priori. These displays can be rearranged in the environment as their exact locations are not important for the success of the interaction. Similar to *Stitching*, our prototype allows for *impromptu* access to displays [HRG$^+$04]. From the user's point of view, the system can start and dismiss connections opportunistically. Besides allowing for immediate access, this feature is especially important for dragging content from one screen to another. *Relative* and *indirect* pointing, for example, requires users to reconnect to the destination display if the environment has not been modeled a priori (see chapter 4). With our *non-modal* connection procedure users can drag content between displays without any additional steps.

The bimanual nature of our system allows both coarse and fine-grained positioning. Using their non-dominant hand, users can coarsely position the device. This defines the reference frame for further manipulation. Within this region, they can use a finger of their dominant hand for precisely manipulating content shown in the viewfinder. This type of interaction allows them to quickly drag items across large distances (by moving the entire device) and accurately position the item at its destination (by moving the finger while the device remains still). Besides employing this technique on a single large display, it can be used for dragging content across displays. Users position their device to access a certain content item. Subsequently, they touch and hold the item and move their mobile device to the destination display. Once arrived there, they can use the finger holding the item to precisely position it. This approach resembles a *dependent* technique in which the one hand (in our prototype the non-dominant one) influences the other [Gui87].

### 7.1.5 Limitations of the Naïve Implementation

While we have transferred the concept of interaction through video, its implementation suffers from several limitations due to the device's mobility. The fixed cameras used in Tani's scenario always produce steady images [TYT$^+$92]. On mobile devices these images are usually more unstable due to even minor device movements caused, for example, by the natural hand tremor. In our case, this influences the positioning of the reference frame which in turn greatly impacts the fine positioning of the dominant hand (see figure 7.6a). Hence, accurate interaction with the current implementation is difficult unless the mobile device is held still completely. However, when users try to hold their device steady in mid-air the resulting postures are rather awkward and most likely are uncomfortable for users when the interaction takes longer.

Another advantage of fixed cameras is that they assume a fixed distance between them and the target displays they are pointed at. When the camera is built into the mobile device, on the other hand, these distances may greatly vary. Mobile interaction through video cannot assume a constant distance especially when multiple users are involved. The further away the target display is, the smaller it appears in the viewfinder (see figure 7.6b). Interacting with small items in the live video image is then affected by the *fat finger problem*. Fixed cameras can overcome larger distance by using zoom to allow for a constant CD ratio regardless of the distance to the actual target. However, zooming into the image increases the aforementioned instability of the video image as even small movements of the device result in high visual movements (i.e., the reference frame moves further than the interaction device). The distance has a great influence on the overall precision that can be achieved with our naïve implementation.

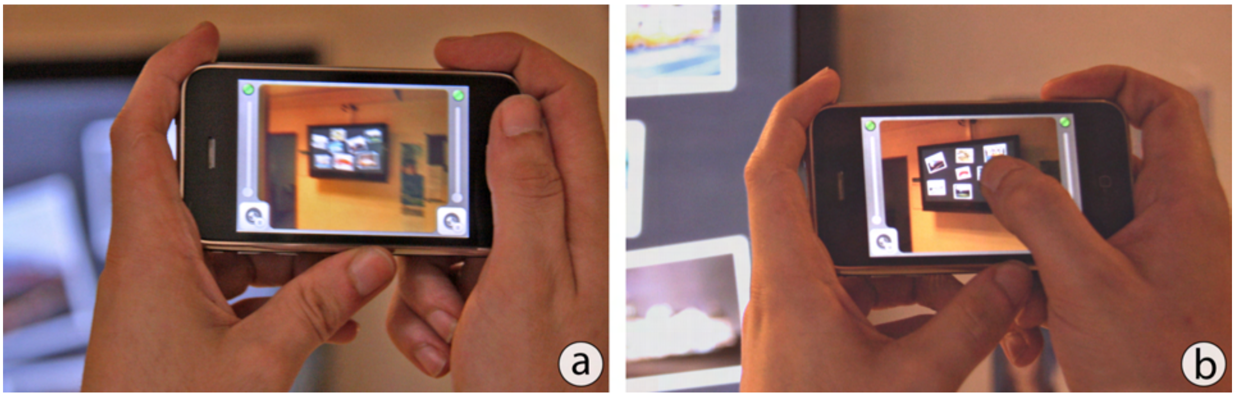


**Figure 7.6:** Two limitations of the naïve implementation: (a) Unstable images caused by the mobility of the camera in combination with slight hand movements. (b) Items are already too small for touch interaction although the display is not far away.

We further found that the item's size has an important role in the interaction. Naturally, large items at higher distances do not decrease the precision as much as small items do. In general one can state that the accuracy of input is proportional to the ratio of the item's physical size on the target display and the distance between the mobile device and the target display. The distance to the display affects the display's size in the mobile device's viewfinder in a linear, but inverted way. This means that doubling the distance to the display causes the screen to be shown with half the size in the viewfinder. The ratio of size-to-distance can be used to calculate the size an item (and a display respectively) has on the mobile device. We refer to this ratio as an item's *apparent size*. In summary this means that even large distances are possible as long as the items are large enough. Large icons on the other hand harm users that are closer to the display since the minimum distance is 1.5 times the item's diagonal (see chapter 6). Enlarging items to allow for higher distances would still result in a limited range of possible distances. For this reason, solutions need to be found that allow for similar apparent sizes regardless of (1) the item's actual size as well as (2) the distance between users and target screen. In the next section, we present a series of modifications to the original metaphor to make it work for variable distances.

## 7.2 Allowing for Variable Distances

As mentioned before, the mobility of the interaction device limits the success of interacting through live video at variable distances. We address these limitations by presenting a series of improvements in this section. We apply the following three improvements: first, we allow users to manually zoom to reach displays further away (see section 7.2.1). Second, we replace the manual zoom with an automatic one to allow for a constant CD ratio throughout the entire interaction process (see section 7.2.2). And third, we allow users to temporarily freeze the camera image for greater stability of the image (see section 7.2.3). In the subsequent section, we then evaluate these techniques based on a selection and a dragging task.

### 7.2.1 Manual Zoom

If users want to precisely interact on an external display, the ratio between the target's size and the viewing distance (i.e., the item's *apparent size*) needs to be reasonably high. As stated in section 7.1.3, our tracking algorithm is comparably robust against smaller target sizes in the viewfinder. Especially for touch interaction, however, the target's apparent size needs to be higher compared to interactions carried out by a pen due to the *fat finger problem* [VB07]. Furthermore, positioning jitter caused by unstable video due to slight hand movements decrease the overall performance of acquiring smaller targets. Interacting *through the display* needs to overcome the limitations introduced by varying distances to displays.

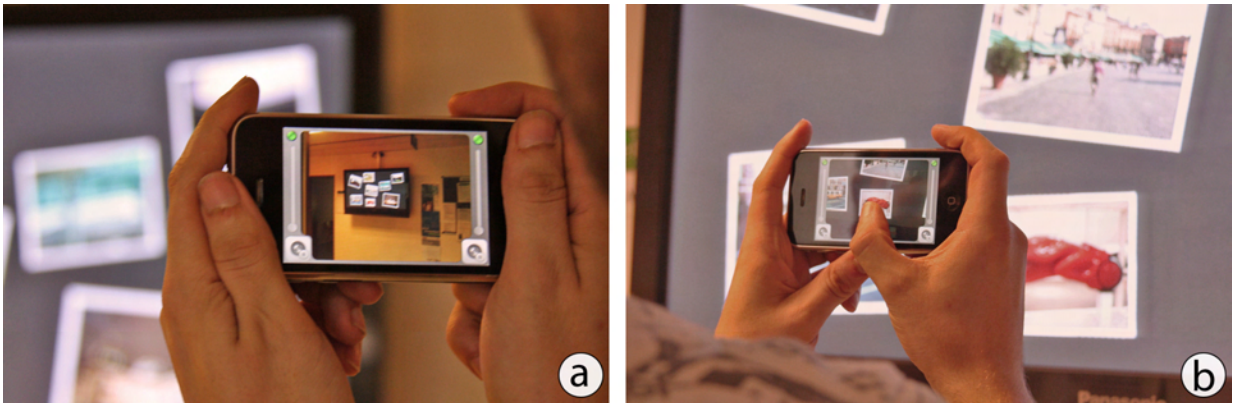


**Figure 7.7:** Walking closer to the display for higher precision. (a) A 50" screen viewed at a distance of 1.6 yards fills only 30% of the mobile display. (b) In order to see a 2" object large enough for precise manipulation, the user has to go very close to the display.

The most obvious way of zooming into a portion of the external display can be done by walking closer to the screen (see figure 7.7). Especially the wide-angle lens built into our mobile device requires users to get very close (i.e., less than a yard) for a reasonably high apparent size. When distant interaction is needed, this solution is not appropriate. Furthermore, displays may be installed in the environment in a way that they have a minimum distance between them and the user. This is, for example, the case, when they are mounted behind subway tracks. Another solution is to take the maximum distance and adjust the target sizes accordingly. For example, in a football stadium, the maximum distance is given by the person seated as far away as possible from the display. This in turn decreases the amount of objects being shown on the external screen and influences on the amount of users interacting simultaneously. Furthermore, users that are closer to the display can then not interact anymore as one item is larger than the entire display of the mobile device. Both solutions resemble a physical (i.e., optical) zoom which resulting in higher image quality but limit the envisioned interaction properties of *multiple users* at *varying distances*. We need to address this problem with another solution.

To preserve varying distances, we wanted to allow users to determine the zoom factor that suits best for them. We added a slider on the mobile device's display which allows for adjusting the zoom in a continuous fashion (see figure 7.8). However, the slider has a maximum value which then restricts the maximum magnification factor. This value can be set according to the highest possible distance and the resulting necessary factor. In a football stadium this would be again the person having the largest distance to the display. The factor can vary between displays and hence needs to be communicated by the environment manager. This means, that whenever a display is recognized, the maximum zoom factor is adjusted. Furthermore, an already adjusted factor needs to be preserved which in turn may cause the slider to move to another position if the range from ”no zoom” to ”maximum zoom” changes. An alternative to using a slider for adjusting the zoom is to use the commonly known pinch-gesture. These gestures are known especially on the iPhone used in our prototype for scaling images. This gesture would then result in two fingers touching the mobile device. Their locations are then ”projected” onto the target display. Hence, these points would be indistinguishable from scaling an image on the external screen.

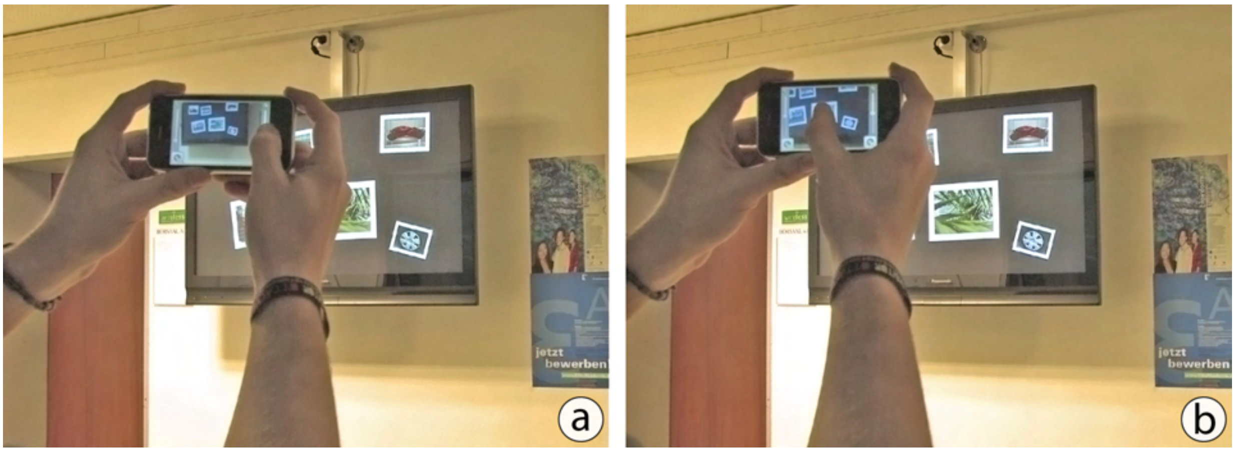


**Figure 7.8:** Using the slider for zooming in manually. (a) Dragging the slider continuously increases or decreases the current zoom factor. (b) Once zoomed in, users can freely interact in the scaled video image.

Similar to most modern mobile devices, our zoom feature scales the image digitally. This generally decreases the image’s quality for higher zoom factors. *Optical* zoom would produce better image quality but adds weight (i.e., the lens) and cost to the device. We decided to use the digital zoom in our concept for verification. Nevertheless, future devices may come with optical zoom at least within a short range (e.g., up to 4× optical zoom). If the concept already works with images that have lower quality, we can assume a similar performance with optical zoom.

### 7.2.2 Automatic Zoom based on Distance

Using the manual zoom seems to be suitable when interacting on one display for a longer time. However, when interacting across displays that differ in their distance (e.g., dragging an item from a projected canvas onto the personal laptop in a meeting room), the zoom first is adjusted for one display and needs to be readjusted once the device reaches another one. This involves the use of a second finger. We found in pilots that users take a finger of the non-dominant hand which already is holding the device. This then decreases the images stability. Hence, another solution needs to be found that facilitates different zoom levels for displays that vary in their distance in an automatic fashion without requiring the user's input.

As stated before, the target's apparent size is a good property in determining the effectiveness of our approach. Similarly, the display's apparent size can be defined by the ratio of the display's size and the viewing distance. Ideally, this ratio and the display's apparent size respectively remain constant throughout the interaction process. A constant ratio would cause the CD ratio in terms of the physical size to be constant as well. More precisely, moving the finger by one pixel will then always result in moving the addressed item by a fixed physical distance (e.g., one-tenth of an inch). While both values - the target's apparent size as well as the display's apparent size - are based on the viewing distance and the respective size of the target or a screen, one must decide which value is taken for measuring the zoom factor. On the one hand, the item's apparent size can be determined with more precision. Keeping the target's apparent size constant would then cause zooming behaviors for different targets on the same display. This may be wanted for precisely manipulating an item but is rather cumbersome to use when dragging an item. In such tasks the overview of the display (e.g., the context of the operation) is important [BGBS02]. On the other hand, the display's apparent size suffers from small calculation errors. Despite these inaccuracies, this value seems to be more effective as it allows one magnification ratio on an entire display based on the viewing distance.

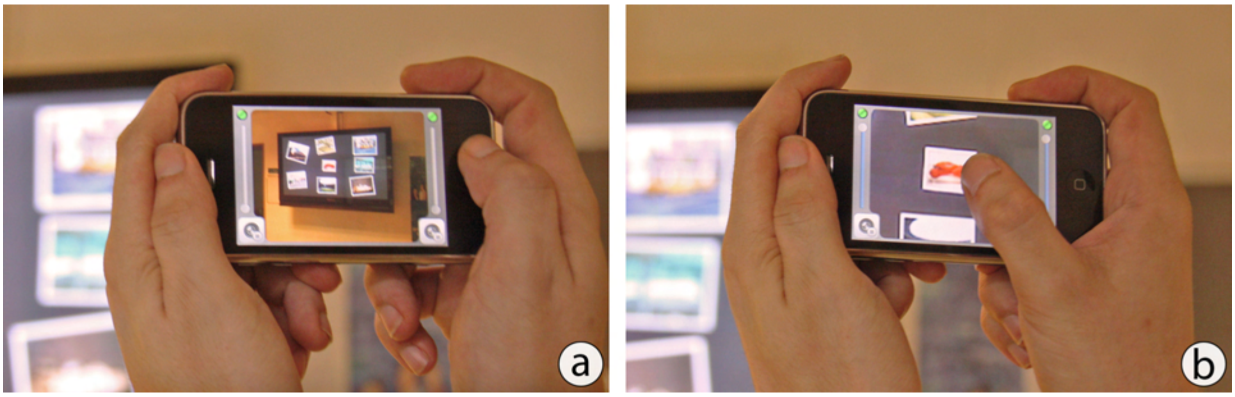


**Figure 7.9:** Zooming automatically based on the display's distance: (a) The display's size in the viewfinder does not allow for precise interaction. (b) To allow a constant CD ratio across all displays, the system zooms into the video image automatically based on the display's distance and size. The zoom factor is updated permanently.

To calculate the currently needed zoom factor, we need the apparent size of the display in the viewfinder if no zoom is applied. This value can be calculated by applying the inverse transformation matrix to all four corner points of the targeted display. With these corner points, the diagonal can be determined which already a good indicator for the display's size is. While the length of this diagonal is now given in pixels, the pixel density of our mobile device can be used to determine its physical length. We also have the physical length of the diagonal of the screen itself. These two values combined with the angle of aperture of the mobile device allows an approximation of the distance between both devices. This distance, the display's physical size, and the predefined CD ratio are now used to calculate the zoom factor needed for the corresponding target display (see figure 7.9). Measuring the distance whenever a frame arrives from the mobile device allows for permanently updating the zoom to keep a single display's size constant. Walking closer to the target screen decreases the zoom factor causing the display to be shown with the same size at all times. Similarly, walking away from the display increases the zoom factor. If the device is not pointed at a screen, the zoom is reset to allow for a greater overview.

### 7.2.3 Stabilizing the Video Image

As mentioned earlier, increased zoom levels amplify even slight tilt operations of the mobile device to a large motion in the camera image. This occurs since the macro CD ratio decreases when the zoom factor increases. At higher distances and increased zoom factors, the loose navigation of the non-dominant hand is too coarse for users to precisely control it. While the bimanual navigation of our prototype partially counters this effect by using the dominant hand for precise interactions, we have to address it especially for large distances by introducing a simplified type of image stabilization: the *freeze* feature. Users can temporarily freeze the live video image by pressing a button (see figure 7.10). This frozen image can then be seen as a static reference frame similar to *WinCuts* in which users can achieve a higher precision [TMC04]. This technique further eliminates the necessity of holding the device still or even pointed at the target display which in turn avoids unnecessary fatigue. To start the live preview, users press the button again.

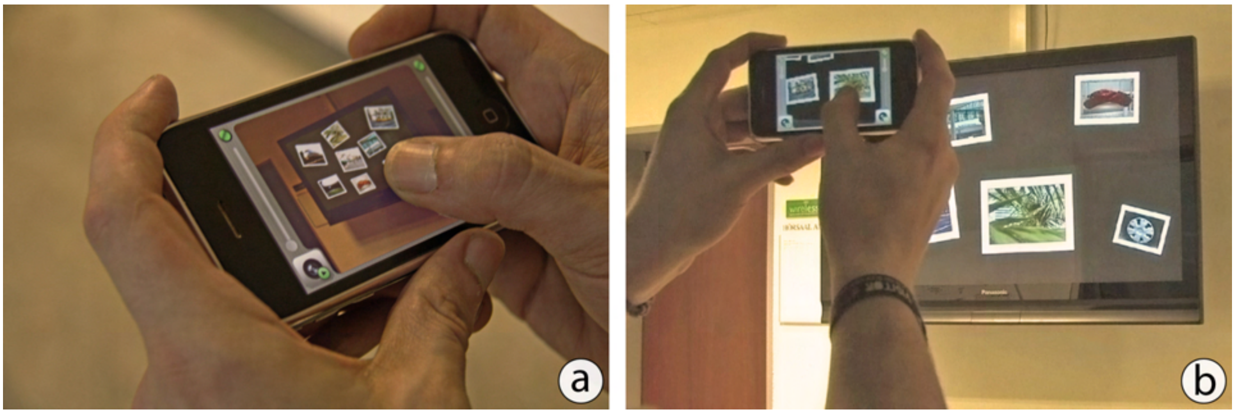


**Figure 7.10:** Freezing the live video image: (a) The live image is frozen without zoom and any further manipulation. Immediate feedback can then only be given by the target display. (b) Freezing can be combined with zoom and further be overlaid with computer-generated graphics for higher image quality and more precise interaction.

Simply freezing the live video preview has two limitations: first, the already mentioned sensitivity of the device's camera is limited making it difficult to take a ”photograph” without motion blur. This especially turns out to be a problem when interacting in rooms that are dimly lit due to projectors that may be present in the environment. And second, the frozen camera image does not allow for live feedback which we see as one of the key aspects of *through the display* interaction (see figure 7.10a). Users may observe changes on the target display, but once the visual content has been changed (e.g., items have been moved to other locations), these changes are not being reflected on the mobile device making it difficult to address shifted content. To overcome this limitation, we decided to overlay the frozen camera image with correctly distorted computer-generated graphics to increase the image's responsiveness (see figure 7.10b). This concept is known from the paradigm of (handheld) augmented reality [WPLS05]. An interesting side-effect is that we can further increase the image's quality if the preview has been zoomed.

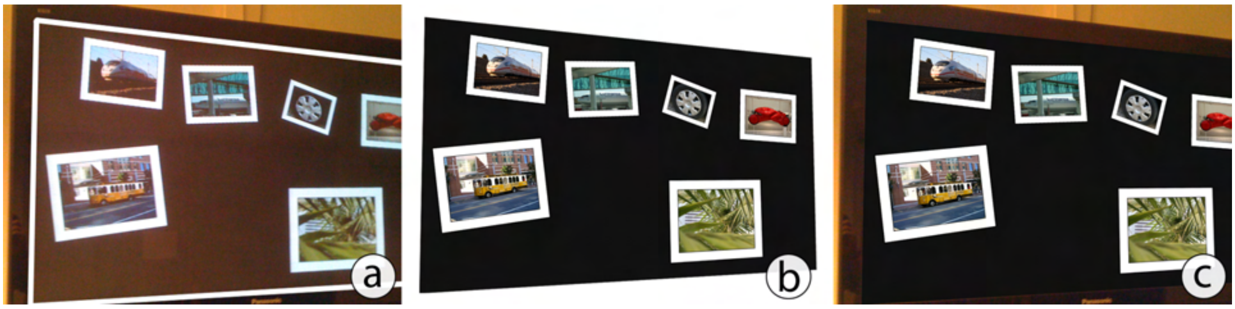


**Figure 7.11:** Overlaying the paused video image for permanent feedback: (a) The paused video image (white borders denote the detected screen). The correctly distorted overlay (b) is then overlaid and permanently updated on the paused image (c).

To create an overlay image with a correct distortion, the inverse of the calculated transformation matrix can again be used. First, we obtain a screen image from the respective target display. The inverse transformation matrix is then used to distort the image pixel by pixel (see figure 7.11). As this operation requires a lot of processing power, we decided to host the image transformation on the environment manager. To save some processing time, only the region that is visible in the mobile device's viewfinder is distorted. In our prototype 320 × 480 pixels = 153,600 pixels need to be transformed. The calculated image is then being transferred to the mobile device where it is overlaid on the frozen camera image. As the mobile device does not send any video frames, we have to use another event for recalculating the image. This is needed when the visual content on the target display changes. Naturally, this occurs when the user with the frozen video image interacts in the computer-generated image. Whenever input is received from the mobile device, the environment manager can recalculate the overlay image. Other users, however, may also interact with the display (either directly or also through a mobile device). The target display needs to inform the environment manager about changes in the image. The managing instance can then use the changed part of the screen image and transform it.

The main advantage of this virtual live preview is that it gives immediate feedback on a temporarily frozen image. It combines the benefits of the live preview (i.e., direct manipulation) with the freeze feature (i.e., more comfortable postures). Thus, the virtual live preview preserves all properties of a physical preview; in particular, it also shows ongoing interactions by other users with the same screen. Combined with a zooming behavior, the virtual image can further increase the quality of the image ultimately simulating optical zoom.

### 7.2.4 Implementing the Improvements

After describing the improvements, we briefly illustrate how these were implemented in our prototype. Higher zoom factors decrease the size of the shown content on the target display. This then influences on the detection algorithm and especially on the distance to the display. We implemented both zoom techniques (e.g., manual and automatic) so that they do not affect the frame being sent to the environment manager. This is done by using two previews overlaid on each other whereas only the top one is being scaled according to the zoom factor. The lower one will have a constant size. This buffer of the lower preview also holds the frame which is being sent to the environment manager. As we increase the canvas of the live preview to simulate zoom, touch points occurring in this view are scaled already. For example, the distance for a point in the non-scaled coordinate system is halved when the zoom factor is doubled. We do not need to change any point at all before sending it to the environment manager.

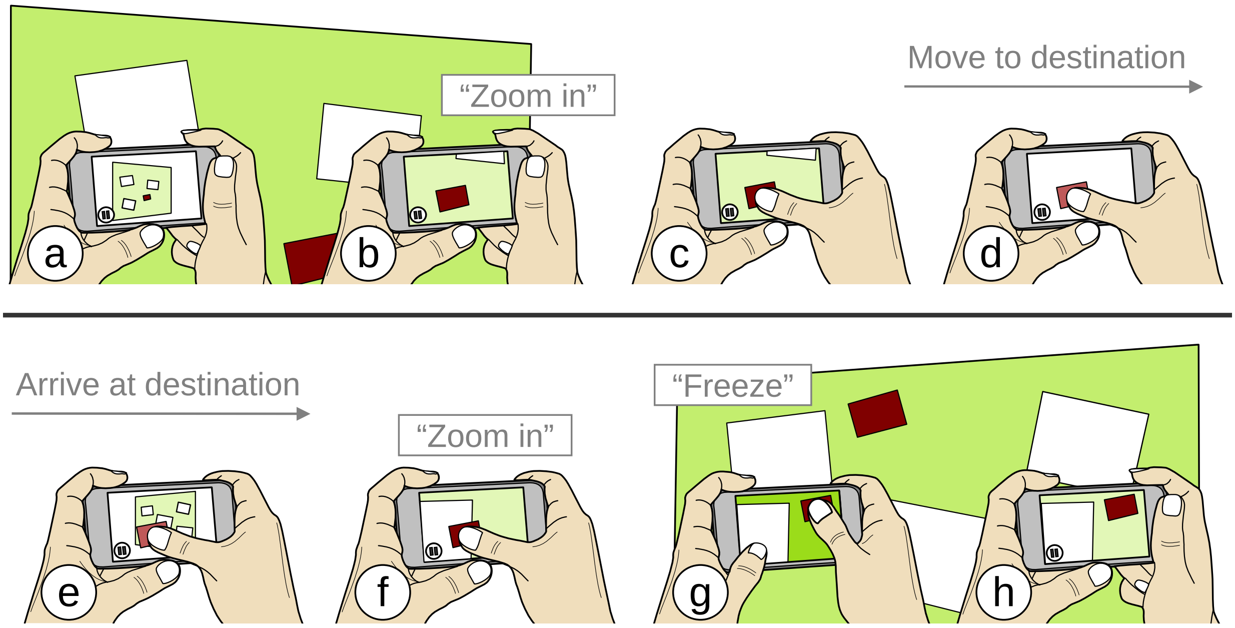


**Figure 7.12:** Cross-display operations with the *updated* prototype: (a) the user aims at a display (b) causing the device to automatically zoom in. (c) This allows the user to touch the red target object and (d) hold it while turning the device towards the other display. (e) Once the device detects the secondary display, (f) it zooms in again. (g) Pushing the freeze button with the thumb of the non-dominant hand causes the live camera image to pause for precise manipulation. (h) Lifting the finger releases the object.

The zoom factor influences on the overlaid images. These images can be either the computer-generated virtual preview or a thumbnail when dragging an item away from one display. Again, the implementation on the mobile device allows adding sub-views to existing ones. For the virtual preview, this means that we can add the image received from the environment manager which is scaled automatically. However, as we want to simulate optical zoom to some extent, we decided against this approach. As we obtain the correctly scaled image from the environment manager, we place this image on top of the frozen camera image without any further transformation. In terms of interaction, we take the touch points in the coordinate system of the camera preview similar to the zoom approach. Thumbnails also need to be scaled to allow a seamless transition from the external display to the mobile device. While these seem to be suitable candidates for being added to the camera view, changing zoom factors (i.e., resetting the zoom when leaving a display) would move them without the user moving the finger. This would result in an unnatural behavior. The thumbnails are also scaled by the environment manager and then placed on a different layer on the mobile device. When the device reaches another display, the resulting zoom does not influence the size of the thumbnail. Figure 7.12 summarizes cross-display operations by using our improvements – namely zooming and temporary freezing the live video.

## 7.3 Evaluation of the Improvements

After presenting our improvements and their implementation, we describe our experimental evaluation of them. We compared each of the three extensions with our first design - interacting through live video without any extension. In our study, participants had to perform two tasks: first, they had to acquire targets on a single display. And second, they had to drag items from one display to another. In both tasks they used the four variations of our prototype (i.e., the naïve technique as well as the three extensions). In this section, we first describe the study setup including a short description of the interfaces, the tasks, and our participants (see section 7.3.1). Subsequently, we present the experimental design, our participants and the hypotheses (see section 7.3.2). We then present the results of the study (see section 7.3.3).

### 7.3.1 Interfaces and Tasks

In our study we had four different interface conditions all of which allowed interaction through video. The *Original* condition enabled users to observe and manipulate content through the pure camera image without any zoom capabilities provided (see figure 7.13a). This condition is our control condition during the test. In the *Manual Zoom* condition, users were able to zoom in using a slider on the mobile device (see figure 7.13b). The maximum digital zoom was restricted to the factor four. In our test, zooming in more was unnecessary as the maximum simulated distances to displays did not require higher zoom factors. Participants could decide on their own whether they zoomed in at all or how far they want to zoom in. In the *Auto Zoom* condition, the mobile device zoomed in automatically based on the distance to the display to maintain a constant CD ratio (see figure 7.13c). In our study, the apparent size of objects shown on the mobile device was constantly 1.2 inches. In the *Freeze* condition, users were able to freeze the image by tapping the *freeze button* (see figure 7.13d). The condition also featured automatic zoom. The frozen image was then overlaid by computer-generated digital image of the external display's portion the mobile device was pointed at. Tapping the button again restarted the live video. Participants were free to choose whether to use freeze.

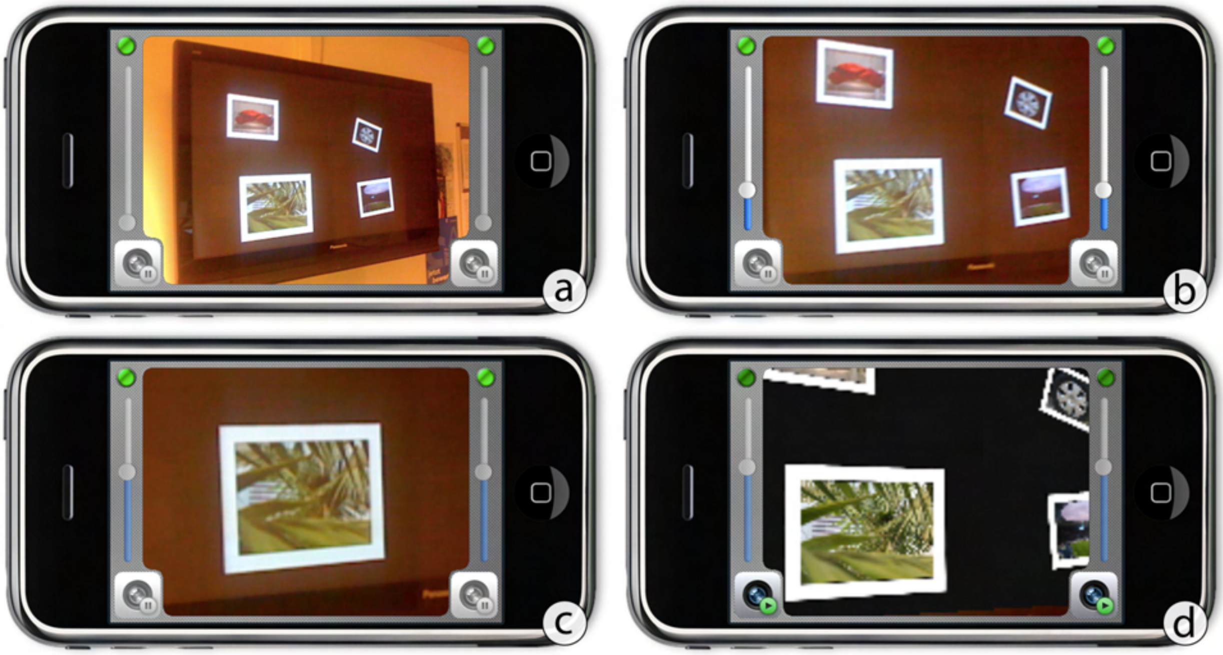


**Figure 7.13:** The four *Touch Projector* interfaces used during the study: (a) Original camera interface. (b) Manual zoom capabilities. (c) Automatic zooming. (d) Freezing the camera image with temporary overlay for precise interaction.

During the study, participants had to perform two different types of tasks. In the *targeting task* participants had to acquire targets on a distant display. As illustrated in figure 7.14, a *start button* was shown on the target screen at the beginning of each trial. After tapping the start button, it disappeared and the target item appeared on the display. Participants were now able to aim at the target display and the item respectively to tap on it. If they missed the target, the error count was increased by one and participants tried to acquire the target again. Once they successfully selected the target item, it disappeared and the start button was shown again to indicate the next trial. We measured the task time from the moment the start button has been tapped until the target item has been selected successfully. For each trial, the target had one of three apparent sizes on the mobile device's screen: 0.3 inches, 0.6 inches, or 1.2 inches. The apparent size was varied to simulate three distances to the target display as our lab did not have sufficient space. Therefore, we kept the distance to the target screen constant and instead varied the target size.

In the *dragging task* participants dragged an object from one display to another. On the originating display, the target's apparent size remained constant at 1.2". As shown in figure 7.14, the setup consisted of two displays. Similarly to the targeting task, the beginning of each trail was indicated by a start button being shown on the originating screen. At the same time, the target's *drop area* (i.e., a frame with the target's apparent size on the screen) was shown on the destination display. After tapping the start button, the trial started by showing the item to be transferred. Participants were then able to aim at the item and acquired the target with a touch-and-hold operation. If they acquired the wrong item, an error was logged and participants had to repeat the trial. Now they could move the mobile device while having the finger still on the display until they reached the destination screen (i.e., it was visible in the mobile display's viewfinder). While not being pointed at a screen, the mobile device showed a thumbnail of the item as described earlier. Once the destination display was visible, the item was transferred from the mobile device to the target display. Participants could now move the item to its exact location indicated by the *drop area*. Lifting the finger initiated the transfer. If the item's center was located in within the boundaries of this area, the trial was completed successfully. If the item's center was outside, participants had the option to correct it. Once a trial was completed, the item disappeared and the start button was shown again on the originating display.

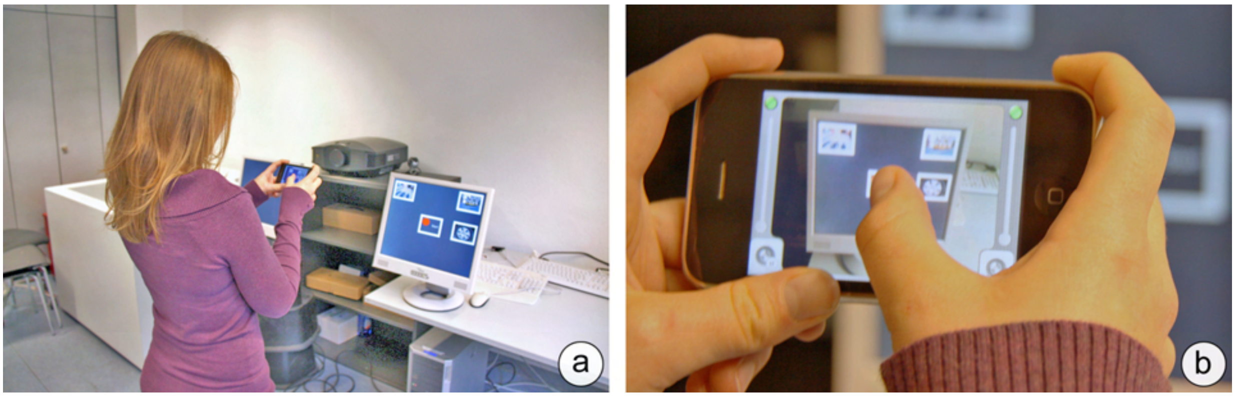


**Figure 7.14:** A participant performing the *targeting* task. She first starts by tapping the start button (a) and then selects the target item (b).

In this task, we measured task time and the item's *docking offset* on the destination display. This offset describes the percentage of the item's area being located outside of the target area. Similar to the targeting task, we varied the apparent size of items on the destination display while the item's apparent size on the originating screen remained constant at 1.2". On the destination display their apparent size again had three possible values: 0.3 inches, 0.6 inches, or 1.2 inches. We further varied the angular distance between both involved external displays. This distance had one of three values: 45° (slightly left of the source display), 90° (directly left of the participant), or 180° (exactly behind the participant). The setup is shown in figure 7.15. In both tasks, we asked our participants to perform the task as quickly as possible while maintaining a low error rate. Furthermore, the simulation of apparent sizes caused the mobile device to zoom in automatically when in the *Auto Zoom* or *Freeze* condition.

## 7.3.2 Experimental Design, Participants and Hypotheses

In our experiment, we used a within-subjects factorial design. In the targeting task, the independent variables were *Technique* (*Original*, *Manual Zoom*, *Auto Zoom*, and *Freeze*) as well as the Apparent Sizes (*0.3 inches*, *0.6 inches*, and *1.2 inches*). In the dragging task, the apparent sizes were only shown on the destination display. We further added the independent variable *Angle* (*45°*, *90°*, and *180°*). The targeting task used a 4 × 3 factorial design whereas the dragging task used a 4 × 3 × 3 factorial design. The dependent variables were task time measured from tapping the start button until successfully completing the trial as well as error rate (i.e., number of taps not hitting the target) for the targeting task and docking offset (i.e., percentage of the item's are not being inside the drop area) for the dragging task.

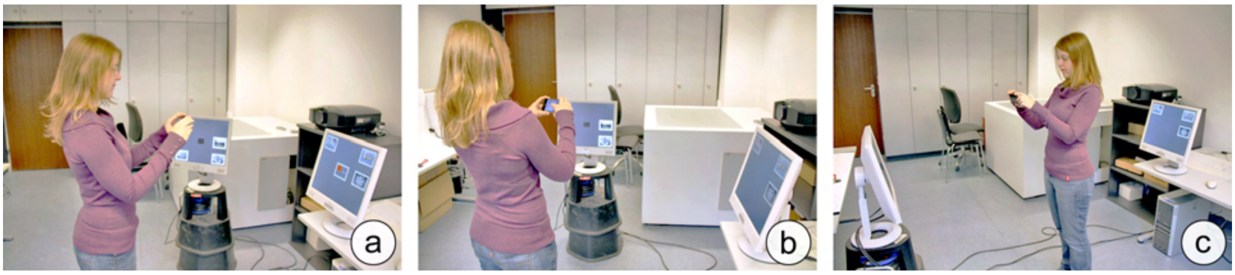


**Figure 7.15:** A participant performing the *dragging* task. We chose three different angles between two external displays: 45° (a), 90° (b), and 180° (c).

The order of *Techniques* was counterbalanced across participants in the targeting task. During the dragging task, we combined *Technique* and *Angle* and counterbalanced the order of these pairs. In both tasks the three *Apparent Sizes* were presented in random order within each block. Each task consisted of one practice block and three timed blocks per *Technique*. Participants received up-front training to get familiar with this interaction style. Excluding the practice blocks, we collected 36 data points per participant for the targeting task and 108 data points per participant for the dragging task. The overall time needed for one participant was 60 minutes or less. During the study, participants spent about 25% of the time with the targeting task and 75% of the time with the dragging task. Although it seems that participants spent less time with acquiring an item, the *targeting* subtask is part of the *dragging* task as well.

For our study, we recruited 12 volunteers (4 female) from our institution, ranging in age from 22 to 30. One of our participants was left-handed. However, this is not a problem with our prototype as all control elements (i.e., the zoom slider as well as the freeze button) have been placed on both sides of the interface. Eleven of our participants had at least some experience with touch-based mobile devices such as the iPhone. None of them was (partially) color-blind while all participants that had visual impairments used vision aids (i.e., glasses or contact lenses). This was important to us since the drop area for the smallest apparent size of 0.3 inches is hard to recognize on the external display when visual impairments are present.

With the understanding of our techniques, we hypothesized that each of the three modifications would lead to an improvement in user performance for small apparent sizes. Our first hypothesis is that zoom-enabled techniques will outperform the *Original* interface in terms of task time as well as error rate (H1). As zooming manually costs time and effort, we further expect that the *Auto Zoom* interface will perform better than the *Manual Zoom* interface in terms of task time while having an equal error rate (H2). As *Freeze* allows very precise interactions, we assume that this interface will outperform all other interfaces regarding the docking offset in the dragging task especially for smaller apparent sizes (H3).

### 7.3.3 Results and Discussion

For each task, we compared separate repeated measures ANOVA tests on mean task completion times as well as errors for each task. In the targeting task, errors were measured as the number of failed attempts to acquire a target. In the dragging task, the error was the previously mentioned docking offset measured in percentage. To determine the nature of detected interaction effects, we performed tests on subsets of the data. All post hoc pair-wise comparisons used Bonferroni corrected confidence intervals to retain comparisons against $\alpha = 0.05$. In this section, we will briefly state the results of both the targeting task as well as the dragging task including a discussion. We then present subjective feedback from our participants received during the study. A more detailed statement of our results can be found in [BBB$^+$10].

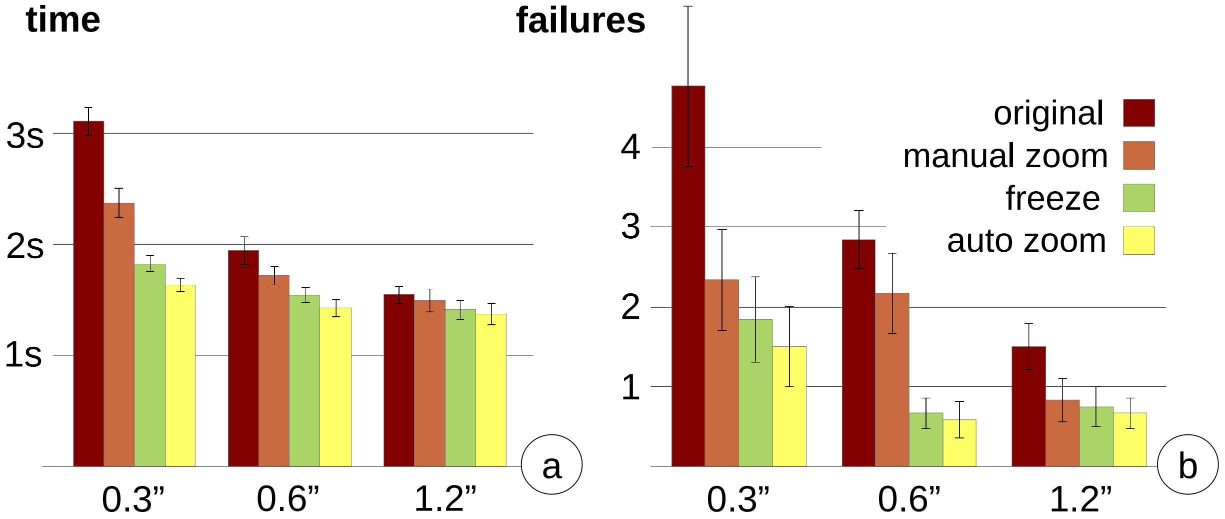


**Figure 7.16:** Results of the *targeting* task: (a) Task completion time and (b) number of failed trials grouped by *Apparent Size*. Error bars indicate ± standard error of the mean.

Regarding the task completion time of the targeting task, the zoom-enabled techniques were faster than the *Original* interface for all *Apparent Sizes*. Especially for smaller target sizes (i.e., 0.3” and 0.6”), the differences were significant. For large apparent sizes all techniques performed equally with only slight advancements of the zoom-enabled techniques. Upon inspecting figure 7.16a, the influence of the apparent size can be observed. The bigger a target’s apparent size is, the smaller the disparity between the different techniques. Overall, the slowest technique was *Original* (M=2198 ms, SD=775 ms), followed by *Manual Zoom* (M=1862 ms, SD=521 ms), *Freeze* (M=1592 ms, SD=299ms), and *Auto Zoom* (M=1476 ms, SD=290ms).

In terms of failed trials, the *Original* interface again performed worst for all *Apparent Sizes*. However, only *Auto Zoom* and *Freeze* differ significantly. The rather low performance of *Manual Zoom* can be explained by the fact that participants did not have to use the feature. They tried acquiring the target first and zoomed in only when they missed it the first time. Especially for medium-sized targets, figure 7.16b shows a significantly lower error rate for both *Auto Zoom* and *Freeze* while *Original* and *Manual Zoom* perform equally bad. This can be again explained by the optional nature of the zoom feature. Overall, *Original* (M=3.0, SD=2.5) had the highest number of failed trials, followed by *Manual Zoom* (M=1.8, SD=1.8), *Freeze* (M=1.1, SD=1.3) and *Auto Zoom* (M=0.9, SD=1.2). As already observed with the task completion time, the difference in failed trials shrinks as the *Apparent Size* increases.

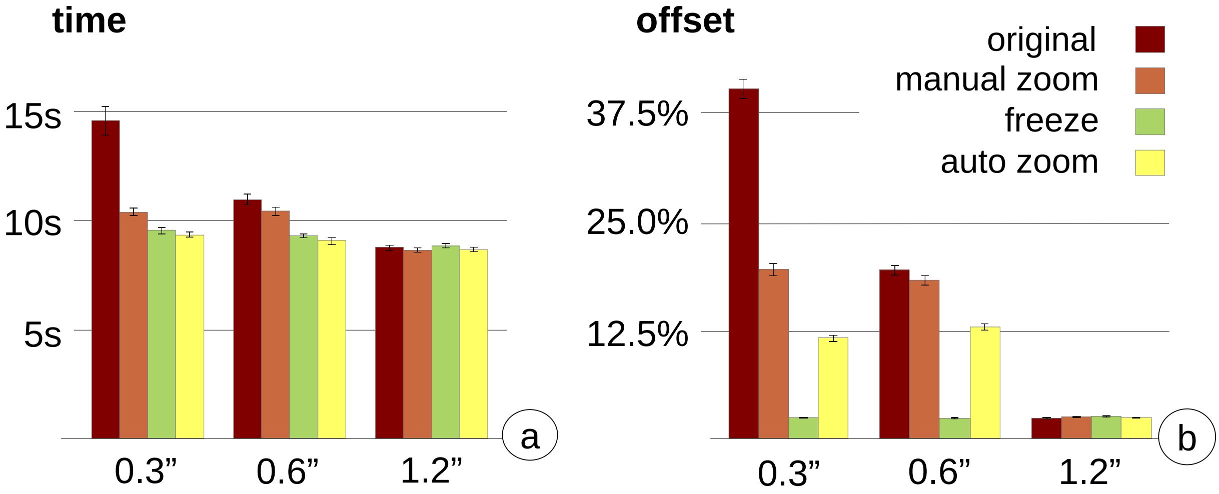


**Figure 7.17:** Results of the *dragging* task: (a) Task completion time and (b) docking offset grouped by *Apparent Size*. Error bars indicate ± standard error of the mean.

The results of the dragging task favored the zoom-enabled techniques even more for small *Apparent Sizes*. While all zoom-enabled techniques performed equally in terms of task completion time, the *Original* interface was only comparable for large *Apparent Sizes* (i.e., 1.2”). The most interesting result is the significantly worse performance of *Freeze* for large *Apparent Sizes* compared to *Manual Zoom* and *Auto Zoom*. This may be explained by the fact that freezing the image increases the task time although the benefit may not be as high as for smaller target sizes. Upon inspecting figure 7.17a, the influence of the *Apparent Size* can be observed. Similarly to the targeting task, increasing the *Apparent Size* decreases the difference among the techniques. As already shown in the targeting task, the *Manual Zoom* technique performed equally worse as the *Original* technique for medium-sized targets. Again this can be explained by participants not using the zoom in the first attempt, but using it afterwards to correct the item placement. Overall, the slowest technique in this task was *Original* (M=11430ms, SD=2785ms), followed by *Manual Zoom* (M=9821ms, SD=1014ms). The fastest techniques were *Freeze* (M=9230ms, SD=493ms) and *Auto Zoom* (M=9024ms, SD=516ms).

For the docking offset we found that *Freeze* was outperforming the other techniques for small and medium-sized targets. For large *Apparent Sizes* all techniques performed equally well. It is also noticeable, that *Freeze* performed similar for each target size. Figure 7.17b further shows that the offset for medium-sized targets is equal for the *Original* technique and *Manual Zoom*, while *Auto Zoom* outperformed both techniques. These two observations explain the significant interaction effect found for *Technique* and *Apparent Size*. Overall, the least accurate of the techniques was *Original* (M=20.6%, SD=15.9%), followed by *Manual Zoom* (M=13.3%, SD=8.0%), *Auto Zoom* (M=8.9%, SD=4.8%) and *Freeze* (M=2.4%, SD=0.2%).

The mechanical nature of our tasks did not leave much space for thinking aloud. However, we did get several comments and suggestions for features in future versions of our prototype. The most prominent feature requested by our participants was adding auditory or haptic feedback (e.g., a short vibration) to indicate when a display has been detected. Another request was to hold the device in a vertical way. The current display detection is not affected by the orientation of the device. Currently, the interface on the mobile device (i.e., the placement of the zoom sliders as well as the freeze buttons) do not adapt to the device's orientation, but will be included in future versions of our prototype. When asked which technique they prefer, our participants stated that they liked *Freeze* the most, simply because the perceived fatigue is the lowest when using this technique. Overall, all our participants seemed to enjoy the interaction.

As hypothesized, all three improvements performed significantly better than the *Original* interface in both tasks except for the largest *Apparent Size*. For the target acquisition, the performance was increased by 49% regarding task time (70% less error-prone) than the *Original* condition. For the smallest *Apparent Size*, the *Auto Zoom* technique was even 90% faster and 68% less error-prone. In general, all zoom-enabled techniques were 34% faster and 59% less error-prone than the *Original* technique. This supports our first hypothesis. In the dragging task, participants overall were 27% faster and 132% more accurate than the *Original* interface when using *Auto Zoom*. In this task, the *Freeze* technique revealed its potential as it was over 10 times more accurate than the *Original* one for small *Apparent Sizes*. This supports our third hypothesis. The *Auto Zoom* also performed significantly better than the *Manual Zoom* in terms of task time and docking offset. This can be explained by participants not zooming in manually (i.e., the technique is equal to the *Original* interface). Nevertheless, this result supports our second hypothesis.

## 7.4 Summary and Discussion

In this chapter we have extended the prototype presented in chapter 6 to allow for continuous operations on and across displays. We demonstrated the implementation, benefits and limitations of a first prototype. We paid close attention to enabling continuous touch input on all displays in the environment and described how our approach can be used for transferring items from one screen to another. This naïve implementation suffered from several limitations: first, when dragging items from one screen to another, the distances between users and both displays may vary. This results in different CD ratios on both screens which in turn causes varying accuracies. And second, the further a user is away from a display, the more likely blurry images occur due to slight hand movements. We addressed these limitations with a series of improvements to make interaction *through the display* usable on mobile devices. We enabled constant *apparent sizes* regardless of the display's distance and its physical size. For large distances we allowed users to temporarily freeze the live video to increase image stability and hence interaction accuracy.

We then compared these improvements and found that zoom-enabled techniques outperform the naïve implementation especially for smaller apparent sizes of targets. The study further revealed that the *Auto Zoom* technique is sufficient if a target only needs to be acquired. When dragging targets to a predefined location, the *Freeze* technique outperformed the other techniques and further showed no difference when used with different apparent sizes. Additionally, this technique was highly favored by our participants. The success of the *Freeze* technique for tasks that need higher precision is easily explainable. Compared to the zoom-enabled techniques, freezing the live video image stabilized it temporarily. Furthermore, users could hold the device in their most comfortable posture without the need of pointing it towards the display. This decreased perceived fatigue especially in arms and shoulders. This means that future versions of such interaction devices should have an automatic zooming as a standard and also present an option such as a button to temporarily freeze the camera image.

According to the design space depicted in section 1.2, the prototype presented in this chapter allows *absolute* and *direct* interaction. In contrast to the systems presented in chapter 5 and chapter 6, the prototype described in this chapter now allows interacting *continuously* on and across remote displays. Our improvements further enable the interaction at *variable distances* by using zooming and temporarily freezing the live camera image. This was only partially valid for the first prototype. Although the system still connects a priori to the actual interaction, the user perceives the connection process as being *non-modal*. The remaining characteristic of the prototype shown in chapter 5 – namely the *flexibility* of the environment – remains unchanged. It is worth to mention that the association between touch points and users improve multi-user handling. Naturally, if users drag items on the respective display, the privacy cannot be preserved as bystanders are aware of the users' actions. As simply selecting an item is still possible with our system, it is of a semi-private nature.

Combined with chapters 5 and 6, we have presented the infrastructure necessary to allow for absolute and direct interaction on external displays. We utilized the mobile display to show content as well, such as temporary feedback or personal content items. Besides interacting on remote displays, the model also allows to transfer information back and forth between an external display and the mobile device. In this chapter, we additionally extended the infrastructure to address the issues caused by the mobility of the interaction device.

# IV

# Conclusions and Future Work

Chapter 8

# Conclusions and Future Work

*Science is always wrong. It never solves*
*a problem without creating ten more.*

**– George Bernard Shaw –**

Our overall goal was to define an interaction model that allows for interacting with external displays using a mobile device. In a first attempt we built two prototypes to investigate the effect of the remote display's distance and reachability. The first prototype (i.e., both displays are available) shows that *absolute* and *direct* interaction is suitable when both displays are available. The second prototype used *relative* and *indirect* input mechanisms when the display is (partly) unreachable. However, in studies we found that (although already widely used) this combination of input characteristics has its limitations compared to *absolute* and *direct* pointing: first, users have to explicitly connect to a display a priori. And second, personal pointers shown on the external display restrict multi-user functionalities. Based on these findings, we showed how to bring the *absolute* and *direct* approach to distant displays as well. We first presented *Shoot & Copy*, a system that allows users to take a picture of the content they are interested in. To allow more flexibility in content selection, we built *Tap & Drop*. This system resembles a bimanual approach which allows both coarse and fine-grained positioning. We also added real-time feedback to avoid sequential operations (e.g., content selection followed by content manipulation). In both prototypes the input itself remained discrete. In our subsequent prototype *Touch Projector* we added continuous operations to allow users to drag items on and across displays. In contrast to previous prototypes, we addressed issues related to the mobility of the device to allow for variable distances between the mobile device and the target display. To do so, we added two functionalities to the mobile device – namely zooming into the live video as well as temporarily freezing the video preview enriched with computer-generated graphics. Overall, we created a model that allows for interacting with external displays regardless of their distance by manipulating remote content shown on both the local and the external screen.

The remainder of this final chapter is structured as follows: We summarize the research conducted in this thesis (see section 8.1). We discuss our advancements in the field of interacting with external displays using mobile devices. Therefore, we take a look at the problems presented in chapter 1 and evaluate how and whether our model counteracts them. Furthermore, we discuss how the individual prototypes as well as the overall model contribute to this field of research (see section 8.2). Nevertheless, the proposed model only serves as a basis for future studies. We present areas for future work that has been opened up by this work (see section 8.3)

## 8.1 Thesis Summary

In the introduction of this work (see chapter 1), we first characterized different display types according to physical factors such as display size, the space they are located in and their reachability. We further categorized displays in terms of their interactivity, the interaction phases they support and the type of information that can be used on them. All these categories, however, are linked and influence on each other. For example, a very large display is usually located in a public space, rather unreachable and does not allow for direct interaction. The classification reveals several combinations of characteristics that may restrict interacting with them. By using personal, mobile devices as interaction devices, these limitations may be reduced or even bypassed. There are numerous ways of this combination to allow for interacting as well as manipulating content on the remote screen. Furthermore, the mobile device first needs to be connected (wirelessly) to the target display which often remains a separate task in current research. In chapter 2, we first identified the three phases of an interaction session (based on phases found in traditional GUIs [Fit96]) – namely *connecting* to a display, *acquiring* content on the display, and *manipulating* it. According to these phases, we reviewed existing approaches in. Furthermore, we paid close attention to existing work in the field of *interaction through live video* as well as solutions for *cross-display content transfer*. After carefully analyzing these approaches, we sorted them into the taxonomy and identified that the *distance* to the external display is critical factor. We further found that the *flexibility* of the environment is influenced by the distance as well. We extended the design space to take these properties into consideration.

In part II of this work we first investigated the distance between the mobile device and the target display (see chapter 3). We presented a setting within which the mobile display is directly superimposed on a larger, public screen (see figure 8.1a). We chose to use a tabletop as external display to avoid the necessity of permanently holding the mobile device. *LucidDisplay* allows users to place their mobile device (i.e., tablet PC) on the external display. We found that physically coupling these two devices resembles an implicit connection from the user's point of view. In this setting, the mobile device can take several roles: first, it may act as a see-through device similar to the concept of a *magic lens* [BSP$^+$93]. Second, the mobile device can show its own content. In this scenario, content transfer can only happen through *Drag'n'Drop* across the mobile device's bezels. And third, the mobile screen shows both its own content as well as remote content located underneath. Content can be transferred between both layers using a single tap. In a study using both applications, we found that (although the external display was reachable) users prefer moving the device over moving the virtual content on the tabletop in order to get a certain region into the focus. Furthermore, moving content from one display to another, a single tap was preferred over *Drag'n'Drop* across the mobile device's bezels.

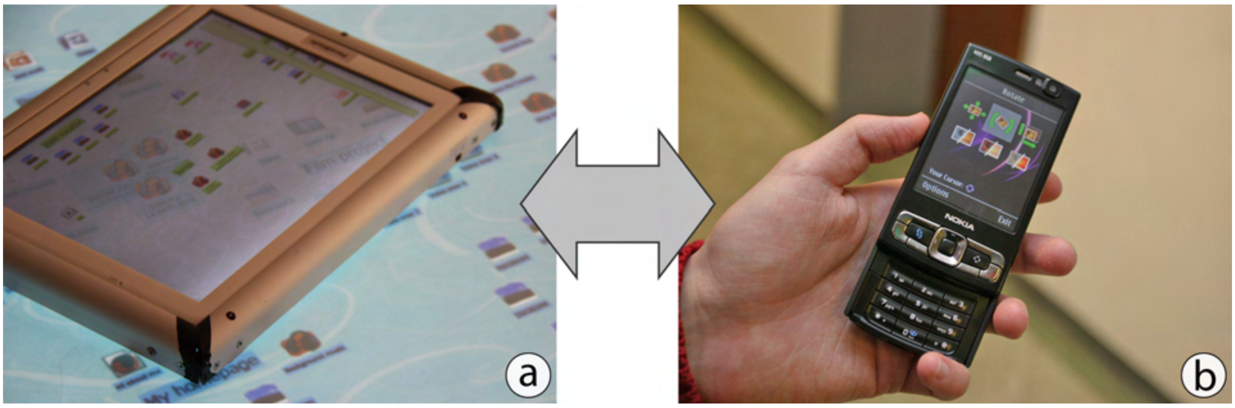


**Figure 8.1:** *Close* versus *variable* distance: (a) *LucidDisplay*, a prototype working at *limited* distances only. (b) *MobileVue*, a system allowing for interacting at *variable* distances.

In chapter 4 we investigated how users can interact with a distant and (possibly) unreachable display using their mobile device (see figure 8.1b). In contrast to the absolute and direct approach, *MobileVue* uses a relative and indirect pointing mechanism. The mobile device acts as a personal mouse that allows users to control a personal pointer to manipulate content on the external display. As the connection procedure cannot be done implicitly as in *LucidDisplay*, we also incorporated this phase into our prototype. We first reviewed possibilities of narrowing down the set of suitable displays. This set was then presented in a two-dimensional layout allowing users to select the best suitable display (i.e., temporal and spatial availability) for them. Once users have selected the display they want to interact with, they were able to control a personal pointer on it using the rich sensing capabilities of their mobile device. We conducted a study comparing different input mechanisms and found that controlling a pointer through linear motion is the fastest solution. Here we found that placing the tool palette on the mobile device is faster (and less error-prone) than rendering it on the external display since the virtual pointer does not need to be moved at all. However, when analyzing our prototype we found several limitations: first, relative and indirect pointing requires an *modal* connection procedure by the user. Second, using an indirect pointing mechanism limits the number of users that potentially interact simultaneously. And third, cross-display interactions using relative pointing require either a model of the environment a priori (stationary displays only) or real-time tracking of all displays in the environment. Hence, we subsequently investigated absolute and direct pointing as alternative.

In part III of this work we first presented how absolute, direct and discrete interaction can be used on distant displays (see figure 8.2). We developed *Shoot & Copy*, a prototype that allows retrieving content from a nearby display by taking a picture of the content's visual representation (see chapter 5). Users aim their mobile device's camera at the respective item which is then shown in the viewfinder. When they selected it, the application on the mobile device took a picture and sent it to the display. We further extended this prototype to allow for multiple displays by using a centralized instance called *environment manager*. This component knows all displays and their content in the environment. The mobile device now sends the image to this managing instance which identifies both the target display as well as the addressed content. The connection between the mobile device and the environment manager is handled in the background during the application launch. From the user's point of view, we have merged the connection and the content selection phase. In a first user study we found that participants understood the concept. We further received two interesting insights regarding the interaction: first, the noticeable long time needed for sending the captured image and receiving the results is still acceptable. And second, users perceived a higher level of privacy as everybody could see *that* they address information, but no one was able to see *which* information they were interacting with.

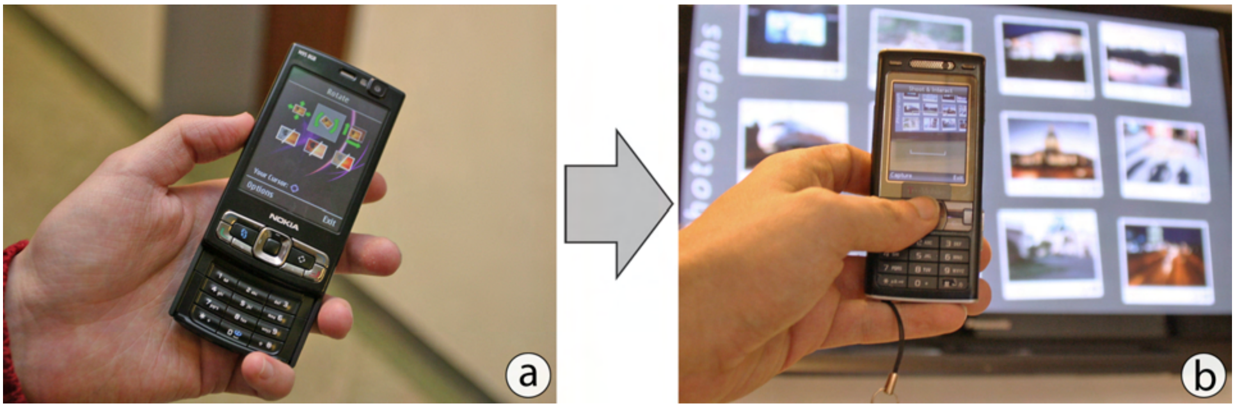


**Figure 8.2:** *Relative* versus *absolute* pointing: (a) A prototype for *relative* input showing an individual menu on the mobile device. (b) Our first system that allows for *absolute* pointing on external displays using touch input *through the display*.

In *Shoot & Copy* the mobile device's center always has to be pointed at the item causing more movements by the user when addressing multiple ones. To allow for a more flexible selection we built *Tap & Drop*, a system that uses a bimanual approach (see chapter 6): The non-dominant hand is used to coarsely position the reference frame (i.e., the portion that is shown in the viewfinder). A finger of the dominant hand can then precisely interact within this reference frame (see figure 8.3). To decrease the overall interaction time we further added continuous feedback on the mobile device's display. This means that users do not have to select an item in order to obtain a list of possible options. By using real-time tracking and identification we are able to show temporary feedback immediately when the item is showing in the viewfinder. In a proof-of-concept application, however, an item had to be tapped which then resulted in temporary feedback on the mobile device. Besides temporary feedback, the mobile display can be used to host personal content. At the same time, the personal content was shown in a semi-transparent fashion allowing users to see both personal and remote content. Similar to the approach presented in chapter 3, we demonstrated how such content can then be transferred back and forth.

Although *Tap & Drop* already allowed for continuous feedback, the input was still discrete (i.e., a single tap for selecting or transferring content). The logical next step for our model was to allow for *continuous* interaction within the viewfinder of the mobile device (see chapter 7). In

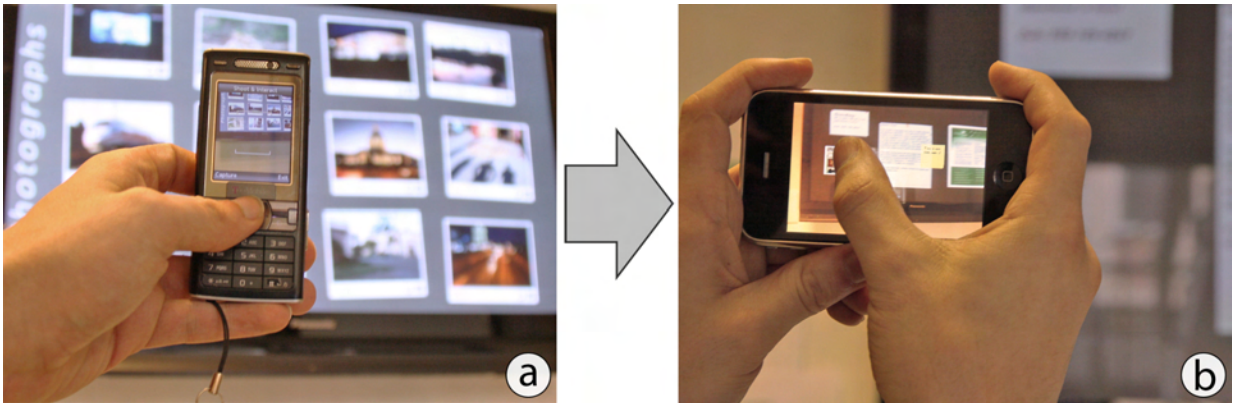


**Figure 8.3:** Turning *uni-manual* interaction into a *bimanual* one: (a) *Shoot & Copy*, a prototype using *uni-manual* input. (b) *Tap & Drop*, a system using *bimanual* input.

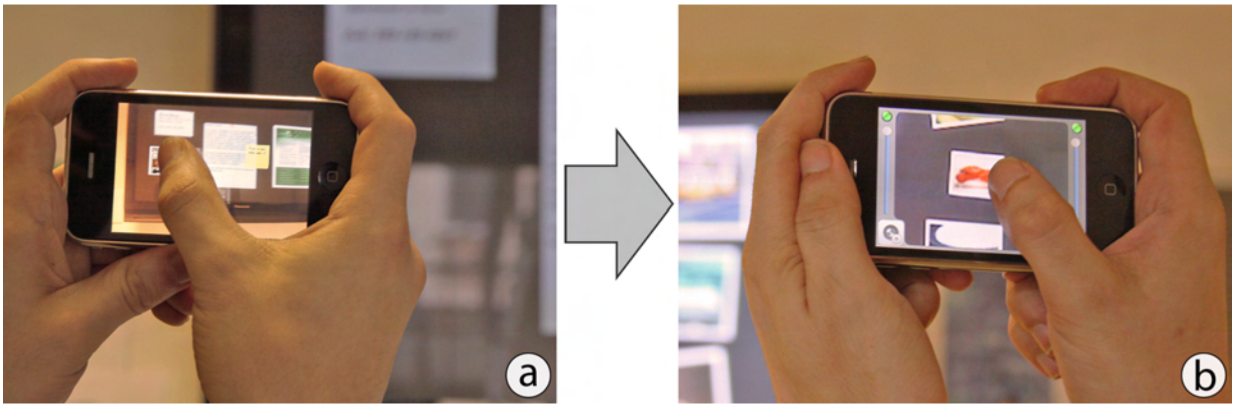


**Figure 8.4:** Turning *discrete* interaction into a *continuous* one at *variable* distances: (a) *Tap & Drop*, a prototype using *discrete* interactions. (b) *Touch Projector*, a system that allows for *continuous* input at *variable* distances.

*Touch Projector*, users are able to move remote items shown in the device's viewfinder as if they would touch the remote display directly (see figure 8.4). Furthermore, users could drag items from one display to another even if they have different input capabilities. Similar to the prototypes presented in chapters 5 and 6, this prototype only allows for precise interactions if the item's *apparent size* (i.e., the ratio of its physical size and its distance to the mobile device) is reasonably high. This mostly is not the case since distances may vary greatly. Furthermore, increasing the item's physical size is not a solution as it only shifts the range of distances within which users can accurately interact with it. We created three extensions to address this: first, we allowed users to manually zoom into the camera image. Second, we replaced the manual zoom with an automatic zoom feature to keep the CD ratio constant regardless of the display's distance. And third, the camera image can be frozen temporarily to stabilize it for precise interactions. In a study we found that automatic zooming (for target acquisition) and freezing the camera image (for precise manipulation) significantly performed better than the original idea. Combined with chapters 5 and 6 this constitutes our model of interacting *through the display*.

## 8.2 Contributions and Conclusions

After summarizing this work in general, we give a more detailed overview of the contributions in this section. Therefore, we revisit the four contributions stated in the beginning of this thesis (see chapter 1): first, we extended the design space of interacting with external displays through mobile phones by adding the *flexibility* of the environment and the *distance* between the user and the target display. Second, we showed how *implicit* connections (i.e., connecting to a display is embedded in the interaction) are possible in environments hosting numerous displays. Third, we allow for interacting with and, most importantly, across displays within *flexible* environments (i.e., the display topology is not important). And fourth, we presented different techniques to allow for *variable* distances between the user and the respective target display. In the following, we describe in more detail how each contribution has been achieved in this work.

**Extension of the Design Space:** The existing design space of mobile device input is only covering interactions with a single display [BRSB08]. By analyzing the different phases of an interaction session (i.e., *connecting*, *pointing*, and *manipulating*), we identified further characteristics for each phase. These criteria consider the environment within which the mobile device is used in. Selecting and connecting to a display can be performed either *implicitly* during the interaction (as described in chapter 3) or *explicitly* initiated by the user (as discussed in chapter 4). The *modality* is the first factor that has been added to the design space. While the pointing task is well described for a single display in the existing design space, cross-display operations are not covered. For certain input mechanisms such as relative and indirect pointing the topology of the environment needs to be known if cross-display operations are possible. This in turn is rather difficult if mobile devices are potential target displays as long as no real-time tracking is used. The *flexibility* of the environment in terms of display arrangements is another important factor. During the manipulation phase the input might be needed with higher precision. However, the distance to the target display may influence on the accuracy. For this reason, we further added the *variability* of the distance between the user and the target display. By adding these three dimensions to the already existing taxonomy of mobile device input, we aimed to get a better understanding of the actual design space. This enriched classification revealed that interacting with external displays at *variable* distances with an *implicit* connection procedure in *flexible* environments seems to be an unexplored field.

**Non-Modal Connections to a Display:** Most interaction techniques between a mobile device and an external display do not consider the fact that a connection has to be established a priori. Certain settings allow for *non-modal* connections from the user's point of view. As presented in chapter 3, placing a mobile device on a larger interactive surface resembles a conceptual link between both devices. On a technical level, this might also initiate the connection. For example, when the external display recognizes a device it broadcasts the detected pattern and waits for the response of the corresponding mobile device [WS07]. When the mobile device is used to control a remote pointer on distant displays in a relative and indirect fashion, however, non-modal connections are hard to realize (see chapter 4). In such settings, users have to select the target display on the mobile device in a modal way. This especially harms cross-display operations (e.g., transferring content from one display to another). If modal connection procedures are required, users have to disconnect from one display (i.e., the origin of the content) and connect to another (i.e., the intended destination of the content) during the interaction. To avoid the limitations of modal connections, our proposed model of interacting *through the display* allows for non-modal connections for distant screens in a similar way as presented for close displays. By using the mobile device's built-in camera, users can point at the display of interest which is then shown in the viewfinder. When they want to interact with content shown in the viewfinder by, for example, taking a picture with the *Shoot & Copy* prototype, the content is selected without the need of previously connecting to the display (see chapter 5). From the user's point of view, our model of interacting *through the display* implements a *non-modal* connection procedure.

**Flexible Environments:** Today's environments (both public and private) may contain several displays that can act as target screen for the user. As long as stationary displays are used (e.g., a tabletop as presented in chapter 3), the arrangement of these screens usually does not change often. However, mobile devices such as laptops or tablet PCs can also act as target display for users. Their mobility then increases the changes in the display arrangement. As long as users only interact on one display, the *modal* connection procedure presented in chapter 4 is sufficient if the amount of displays in the environment remains rather low (e.g., less than ten screens). Furthermore, relative and indirect pointing seemed to be a good candidate to interact with content on one display. When users want to perform cross-display operations, the virtual remote pointer needs to move from one display to another with a correct perspective transformation. This transition from one display to another then requires the spatial display layout of the environment to be known. To allow for ideal transitions, users (and their head in particular) need to be tracked permanently. A three-dimensional model of the environment containing the spatial information of all displays is needed. Absolute and direct pointing techniques do not need such a model but may limit the flexibility in terms of the display's input capabilities. *Pick & Drop* as an example requires both the originating and the target display to understand pen input [Rek97]. Our model of interacting *through the display* does not rely on input capabilities of involved displays. By allowing the mobile device to track itself with respect to displays in the environment in real-time, the display arrangement is not important at all times (see chapter 6). If users want to perform cross-display operations, they can simply drag content from one display to another regardless of the spatial relationship between these two screens. This is possible since users point in an absolute and direct fashion by aiming at the display of interest. It is not important how displays are arranged or how they change during the interaction as only a line-of-sight is necessary. For this reason, our model allows for cross-display operations in *flexible* environments by using the aforementioned *non-modal* connection procedure.

**Interacting at Variable Distances:** The more displays an environment contains, the more the distances between users and these displays vary. For example, one display may be close to the user, while another one is further away. The distance greatly influences the accuracy of our proposed model of interacting *through the display*. This can be explained with the ratio of the display's size and its distance which we refer to as the display's *apparent size* in the mobile device's viewfinder. This size then needs to be reasonably high to allow for precise interactions. Since camera-equipped mobile devices usually feature wide-angle lenses, the apparent size decreases rapidly even if users are not far away from the display. All of the prototypes using *absolute* and *direct* pointing investigated in chapter 2 as well as our prototypes *Shoot & Copy* and *Tap & Drop* only allow for a limited distance. Prototypes using relative and indirect pointing (see chapter 4) on the other hand may allow for variable distances. As long as users want to interact *through the display* with a single screen only, they can of course walk closer to it. This is not always possible as several screens are placed in a way that users have to keep a certain distance (e.g., behind subway tracks). As we wanted to allow for variable distances (i.e., close to as well as further away from the target display), we implemented several improvements to the original idea of interacting *through the display* (see chapter 7). Naturally, users are able to zoom into the live camera image. The more a user zoomed in, however, the more unstable the live video got due to the natural hand tremor. We further had to stabilize the video image and decided for a simple solution: temporarily freezing live video. We further superimposed a computer-generated virtual image of the screen's content on the frozen camera image to allow for live updates. These improvements significantly outperformed the original implementation of our model now allowing for *variable* interaction distances.

In summary, we created a model that allows for establishing connections in a *non-modal* way in *flexible* environments. We further showed how interacting *through the display* can be used at *variable* distances. Based on the original taxonomy of input devices, our model uses *absolute* and *direct* pointing with both *discrete* and *continuous* feedback [CMR90]. Throughout this work, we identified the different strengths and weaknesses and decided on this combination for our model. We found that our added dimensions of *modality* of the connection, *flexibility* of the environment, and *variability* of the distance, however, cannot be achieved by multiple combinations of these properties. In the following we describe how the the original properties influence on our extensions within the taxonomy.

The first property was *absolute* versus *relative* input. In our early prototypes we found that relative pointing always requires a pointer whereas absolute mechanisms do not rely on such. While relative pointing usually is *indirect*, absolute pointing can be carried out in either an *indirect* or a *direct* fashion. The use of indirect mechanisms again requires some kind of pointer to be shown. In case of relative and indirect techniques, this pointer further needs to be of a virtual nature. On the other hand, Absolute and indirect input such as laser pointers create their pointer on their own through a laser beam. While this would allow for cross-display operations more easily compared to relative and indirect mechanisms, it requires each display to recognize the laser pointer. This then limits the flexibility of the environment. It is apparent that *absolute* and *direct* techniques do not have these limitations but usually require the user to reach the display directly. Several prototypes have been proposed to overcome this and allow for absolute and direct pointing with some distance between the mobile device and the external display [BRS05, PJO09]. These approaches had two major limitations: first, the user needed to be quite close to the display (i.e., due to the wide-angle lens). And second, the prototypes still require some kind of pointer although absolute and direct pointing is used. These pointers are primarily used to obtain the spatial transformation between the mobile device and the target display. For the third property (i.e., *discrete* versus *continuous* input) we chose to implement both and found that discrete input is sufficient for selection tasks. However, continuous feedback allows both selecting targets and dragging them on and across external displays.

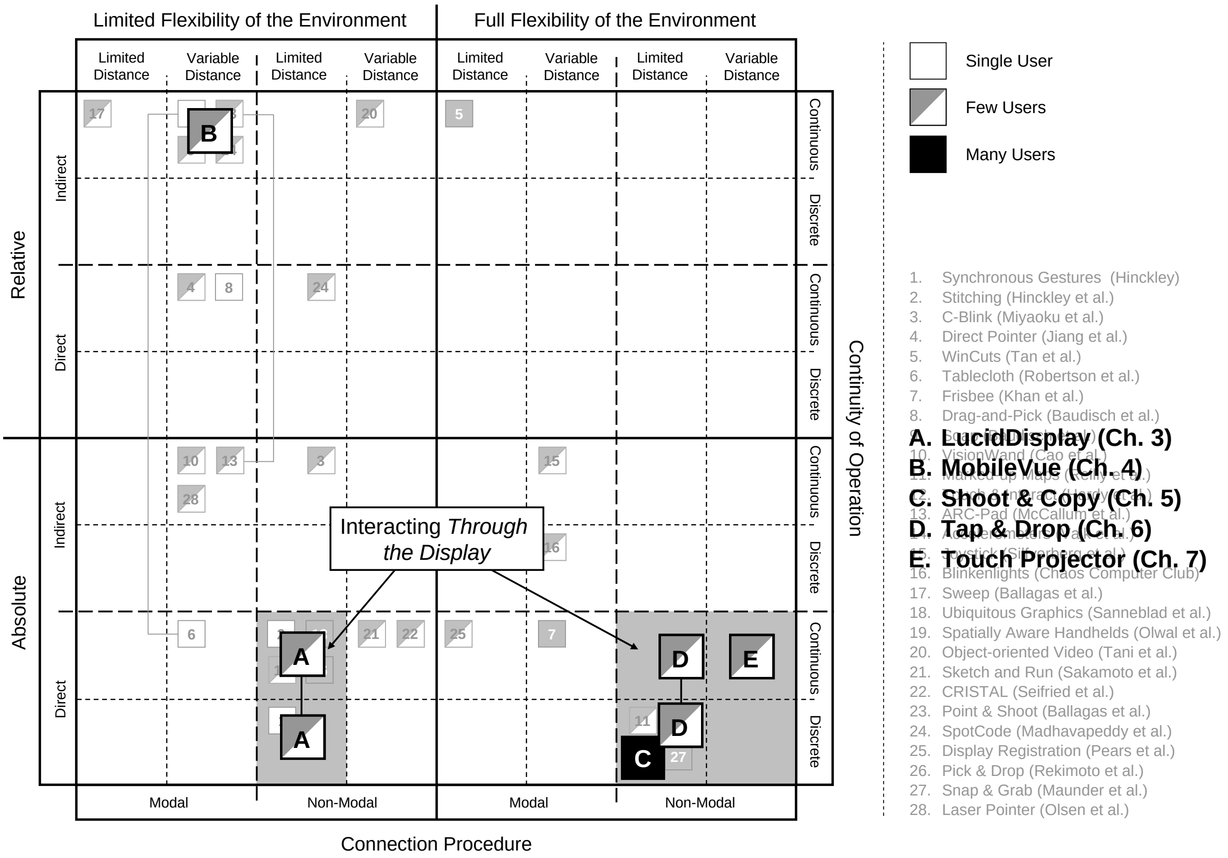

Limited Flexibility of the Environment
Full Flexibility of the Environment
Limited Distance
Variable Distance
Limited Distance
Variable Distance
Limited Distance
Variable Distance
Limited Distance
Variable Distance
Relative
Absolute
Indirect
Direct
Indirect
Direct
Continuous
Discrete
Continuity of Operation
Modal
Non-Modal
Modal
Non-Modal
Connection Procedure
Interacting *Through the Display*
A
B
C
D
E
Single User
Few Users
Many Users
1. Synchronous Gestures (Hinckley)
2. Stitching (Hinckley et al.)
3. C-Blink (Miyaoku et al.)
4. Direct Pointer (Jiang et al.)
5. WinCuts (Tan et al.)
6. Tablecloth (Robertson et al.)
7. Frisbee (Khan et al.)
8. Drag-and-Pick (Baudisch et al.)
9. Soap (Baudisch et al.)
10. VisionWand (Cao et al.)
11. Marked-up Maps (Reilly et al.)
12. Touch & Interact (Hardy et al.)
13. ARC-Pad (McCallum et al.)
14. Accelerometers (Vajk et al.)
15. Joystick (Silfverberg et al.)
16. Blinkenlights (Chaos Computer Club)
17. Sweep (Ballagas et al.)
18. Ubiquitous Graphics (Sanneblad et al.)
19. Spatially Aware Handhelds (Olwal et al.)
20. Object-oriented Video (Tani et al.)
21. Sketch and Run (Sakamoto et al.)
22. CRISTAL (Seifried et al.)
23. Point & Shoot (Ballagas et al.)
24. SpotCode (Madhavapeddy et al.)
25. Display Registration (Pears et al.)
26. Pick & Drop (Rekimoto et al.)
27. Snap & Grab (Maunder et al.)
28. Laser Pointer (Olsen et al.)
A. LucidDisplay (Ch. 3)
B. MobileVue (Ch. 4)
C. Shoot & Copy (Ch. 5)
D. Tap & Drop (Ch. 6)
E. Touch Projector (Ch. 7)

**Figure 8.5:** The taxonomy of mobile device input filled with our presented prototypes: The shaded area denotes the design space of interacting *through the display*.

Overall, the model presented in part III of this work resembles an absolute and direct approach that still fulfills all the new criteria added to the design space. Figure 8.5 denotes how our prototypes fit into the taxonomy presented in the introduction. It becomes apparent that the presented approaches (except the one described in chapter 4). Although users can inter interact at variable distances with our model, the mobile device still needs to maintain a minimum distance to the external display. This is primarily caused by the use of live video as tracking source. If the mobile device is too close to the target display, the video image is blurred more and more. Even if the system would still be able to track the device based on this video image, users would not be able to see details in the viewfinder making it hard to interact with the shown content (see figure 8.6). To overcome this, the mobile device needs to show computer-generated graphics permanently similar to *LucidDisplay* (see chapter 3). While this may require a different tracking solution, it still shows that interacting *through the display* (either in live video or in computer-generated graphics) finally allows for interacting at *all* distances. Nevertheless, the maximum distance is still defined by the user's ability to identify content visually.

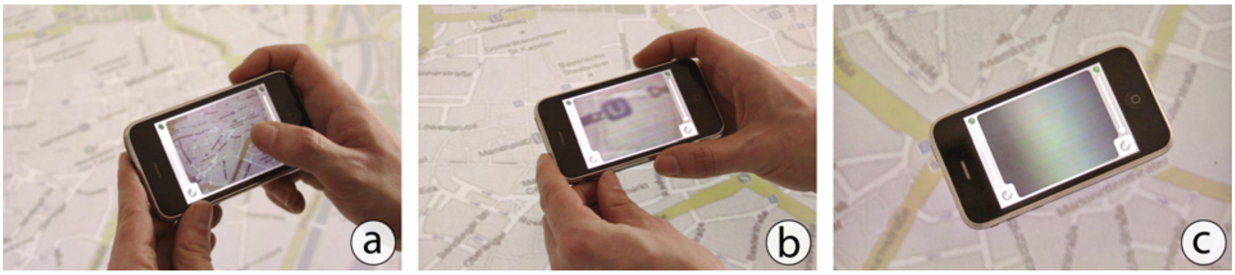


**Figure 8.6:** Problems of live video with directly superimposed displays: (a) At distances of several inches users can interact in a sharp video image. (b) Getting closer blurs the video image. (c) Placing the camera on the external screen makes the video unusable.

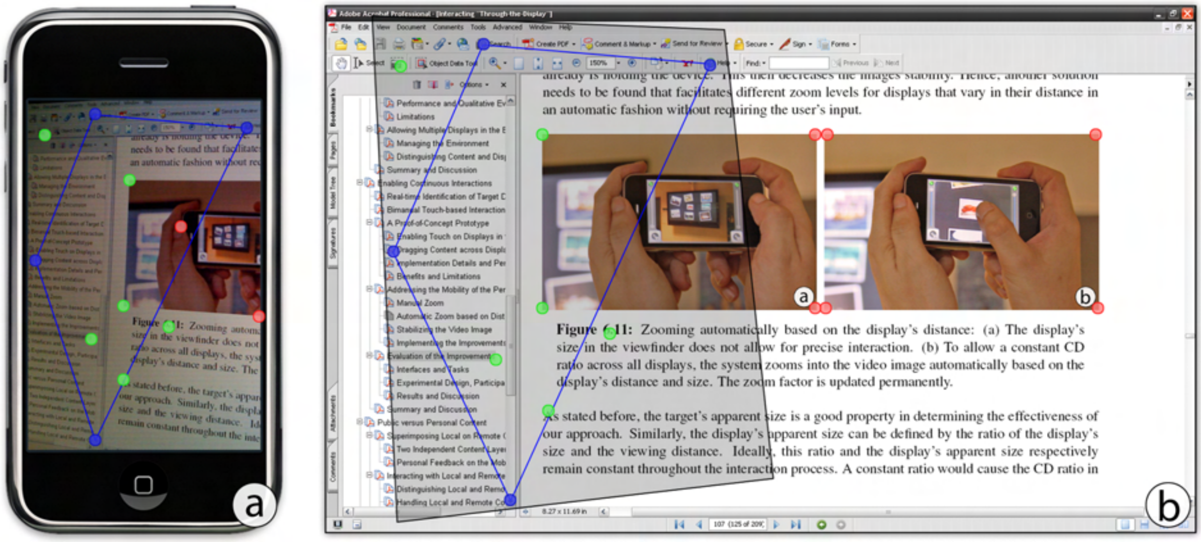
## 8.3 Future Work

In this thesis we have presented the first model of interacting *through the display* with several prototypes. Based on our experiences in designing, implementing and studying these systems, we were able to identify (1) areas for improvement as well as (2) future research directions. These possible improvements and future directions include both technical aspects as well as interaction techniques. In this section we highlight three of them: first, we present a new approach for tracking the mobile device to allow for a more generalized framework (see section 8.3.1). Second, we look at visual feedback during multi-selection tasks when the mobile device is used as input device only (see section 8.3.2). And third, we present another use case in which the depth information obtained by our model may be beneficial (see section 8.3.3).

### 8.3.1 Tracking the Mobile Devices

In chapter 5 we have presented the tracking (i.e., obtaining the spatial relationship between both displays) of the mobile device purely based on visual and geometrical properties of the target screen. We refined this in chapter 6 to allow for real-time detection. While this tracking solution does not use fiducial markers, it still relies on certain display configurations. For example, items shown on the display should not overlap with others as each item serves as marker. Our solution turns the entire display into a multi-marker (without being too apparent to the user). From the technical point of view, however, the problem of tracking markers did not really change at all. Furthermore, our solution only allows identifying displays whose content has (1) rectangular shapes and (2) a clear distinction between content and background. We assume that the second case will be common on digital displays as the human vision also relies on contrasts. The first restriction is a true limitation of our system which needs to be addressed in the future. In the following we present a first idea of a new tracking solution.

Instead of identifying the visual content directly, the system could use feature information contained in each frame. In chapter 6 we already use this approach for subsequent frames after successfully processing one frame fully. We found that the feature point extraction and compar-



already is holding the device. This then decreases the images stability. Hence, another solution needs to be found that facilitates different zoom levels for displays that vary in their distance in an automatic fashion without requiring the user's input.

Figure 6.11: Zooming automatically based on the display's distance: (a) The display's size in the viewfinder does not allow for precise interaction. (b) To allow a constant CD ratio across all displays, the system zooms into the video image automatically based on the display's distance and size. The zoom factor is updated permanently.

As stated before, the target's apparent size is a good property in determining the effectiveness of our approach. Similarly, the display's apparent size can be defined by the ratio of the display's size and the viewing distance. Ideally, this ratio and the display's apparent size respectively remain constant throughout the interaction process. A constant ratio would cause the CD ratio in

**Figure 8.7:** Tracking purely based on feature point extraction: (a) The device can extract feature points in each frame on its own. (b) These are then compared with the ones extracted from the screen image. Green (red) points denote feature points that were (not) used. The blue points illustrate the largest possible quadrangle used to compute the final transformation. The shaded area on the screenshot is the one covered by the mobile device.

ison lead to a significant amount of time being saved. Processing a full frame needed about 180 ms while subsequent frames only needed 25 ms at most. To allow for true real-time interaction, it is desirable to have 100 ms or less in total [Azu97]. Instead of fully processing the first frame, we can also extract feature points based on the *Lucas-Kanade Optical Flow Method* [LK81]. The environment manager can then compare these feature points with the ones found for visual content of nearby displays. Similar to our approach, the largest quadrangle is computed and the four corresponding points are used to create the final transformation matrix. For subsequent frames, these points can then be used in the same way as described in chapter 6. In contrast to our approach, however, these feature points are not bound to rectangular shapes anymore but can be used on arbitrary screen content (see figure 8.7). They simply represent points that are significantly different from others. The difference measurement is the color difference of neighboring pixels for both close and far ones of the originating pixel. This approach is similar (to some extent) to the SIFT[1] algorithm and its faster SURF[2] implementation [Low99, BETVG08].

This approach further avoids another limitation of the our approaches. In our prototypes we permanently stream the video image on a frame-by-frame basis. Although we decided to use wireless LAN, it sill decreases the remaining bandwidth for other information such as input points or thumbnails. In contrast, this idea may be implemented in a way that live video does not need to be streamed to the environment manager. With the current processing power of mobile devices, the feature point extraction can already be performed on the device itself. The only informa-

[1] SIFT stands for *Scale-invariant Feature Transform*

[2] SURF stands for *Speeded Up Robust Features*

tion that is then transferred is the set of feature points. The environment manager nevertheless compares these points and calculates the transformation matrix in the aforementioned fashion. One requirement remains making the algorithm's usage cumbersome (or even impossible) at this time. Live video on mobile devices usually has a lower quality compared to a picture taken by the same camera. The feature points differ from the perfect points detected in the screen image of any display. One solution to address this is to use better cameras that allow for higher resolution in the live video image. On the other hand, this increases the necessary processing power significantly and would result in slower image processing overall.

### 8.3.2 Feedback during Multi-Selection Tasks

The prototypes presented in part III allow the ”projection” of any input point on the target display. While this appears to be a good solution in addressing content on remote screens, having a visual representation of the contact point on the external display may not always be the best choice. For example, when users want to draw a free-form selection on the target display, the mobile device's movement may influence on the line's accuracy. In the original setup, the free-form's line is shown on the target display and in the viewfinder of the mobile device respectively. Both moving the input point (e.g., the finger) as well as moving the mobile device would result in changes of the line itself which may be unwanted. The solutions addressing the mobility of the interaction device (i.e., especially the *freeze* mode) in chapter 7 would reduce this issue. However, the discussion in chapter 6 about employing the local display begin a drawing canvas as well then introduces new possibilities for this kind of scenario.

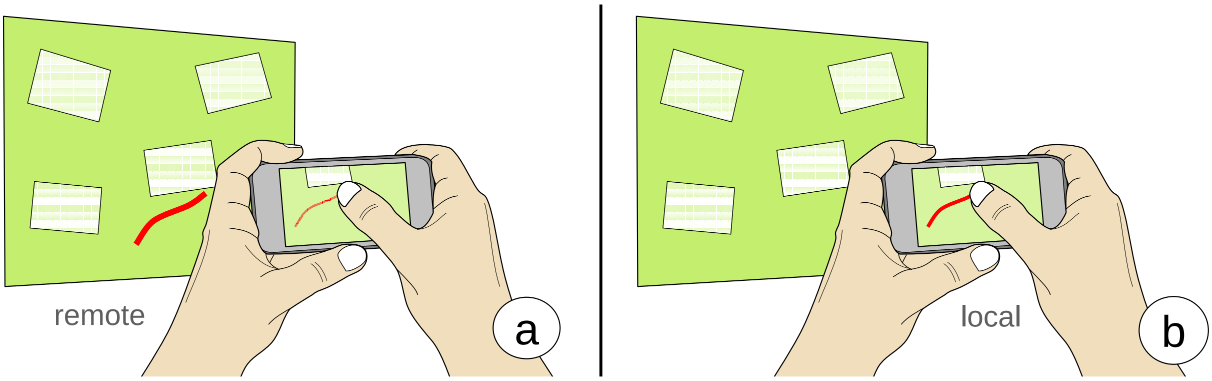


**Figure 8.8:** Rendering a line drawn either *locally* or *remotely*: (a) The line is rendered on the target display but also shown in the mobile device's viewfinder. (b) The line is drawn directly on the mobile device's screen without being shown remotely.

When thinking about the placement and behavior of such a line, two considerations again have to be made: first, the line can be shown on the mobile device or the target display (see figure 8.8). And second, it can be *content-centric* (i.e., fixed with respect to the target display) or *device-centric* (i.e., fixed with respect to the mobile device). Naturally, rendering the line on the mobile device will lead to a more accurate one as it is independent of the device's movement. However, the content users want to select slightly moves in the viewfinder. Users may then adapt to the content's new location by correcting the line. Furthermore, having the line in a *device-centric* way, selections across two display surfaces are only possible when both screens are shown in the viewfinder simultaneously. In other words, moving the mobile device from one display to another does then not allow for selecting content on both displays at the same time. This does not depend on the physical location of the line (i.e., on the mobile device or the target display) as long as the line is in *device-centric*. On the other hand, *content-centric* lines allow this type of selection but in turn decrease the accuracy as they are influenced by the device movement.

Besides the rendering of dynamically created content, the input processing is an open question. If the aforementioned line, for example, is rendered on the target display, input events are naturally performed remotely. When the line is shown on the mobile device, the input can be processed in two ways: first, the events are still sent to the target display through the environment manager in real-time. And second, the input is kept on the mobile device until the selection process is finished. In the latter case, the mobile device then sends the entire shape of the selection form to the target display selecting all items that are contained in it. While this seems to be the more stable solution, the first case also has one advantage. When input is processed directly on the target display, selected content (i.e., content that is currently within the shape) can be highlighted to indicate the activation. Highlighting the content can then again be done on the mobile device if multiple users interact at the same time. We currently implement the prototype and investigate the different placements as well as the centrality of the temporary content.

### 8.3.3 Other Usage for Depth Information

In the presented prototypes we only used the depth information to allow for automatic zooming (see chapter 7). Other scenarios in which the content contains depth information can be envisioned as well. One prominent example is the concept of piles on a digital tabletop [MSW92]. Such piles represent multiple items placed on top of each other. While the pile's conceptual structure could be transferred to the digital domain, the representation falls short: access to virtual piles is still restricted to the display's surface. Even with novel three-dimensional input techniques the pile is bound to the screen and direct manipulation is not possible [HCC07, HIW$^+$09]. On-screen navigation and selection in piles are mostly provided in two different ways: first, users can fan out all items of a specific pile. It results in both temporary higher screen real-estate requirements as well as a visual overload of the user [AB06]. Second, users can flip through the pile item-by-item (e.g., by moving a pen/finger over the pile to jump between items) which might take a long time in the worst case. These workarounds pull the two-step process of navigating a real pile apart: a rough homing in on the target's height followed by flipping from item to item.

Using a mobile device in mid-air above the surface can enable this kind of pile browsing (see figure 8.9). Users can vertically move their personal display on top of a pile. The distance to the surface determines the position within the pile. On the mobile device's screen, users are then able to see the item at this position. The pile itself remains unchanged on the tabletop. Its unchanged visual representation still allows others to interact with it. Similar systems exist that would allow for this kind of interaction. *SecondLight* as the most prominent one enables two different images nearly simultaneously by using a switchable diffuser [IHT$^+$08]. In contrast to this approach, our tracking solution does not rely on external hardware. Our approach could be used on various displays as well as with various mobile devices assuming they have a built-in camera. Furthermore, personal items can be brought into the system using a mobile device instead of using a display canvas. Nevertheless, the video-based approach requires the user to have a certain minimal distance between the device and the surface to still allow for a correct depth measurement. Therefore, navigating to the lowest item of a pile is not done by placing the mobile device directly on top of the surface but holding it at the minimal distance instead.

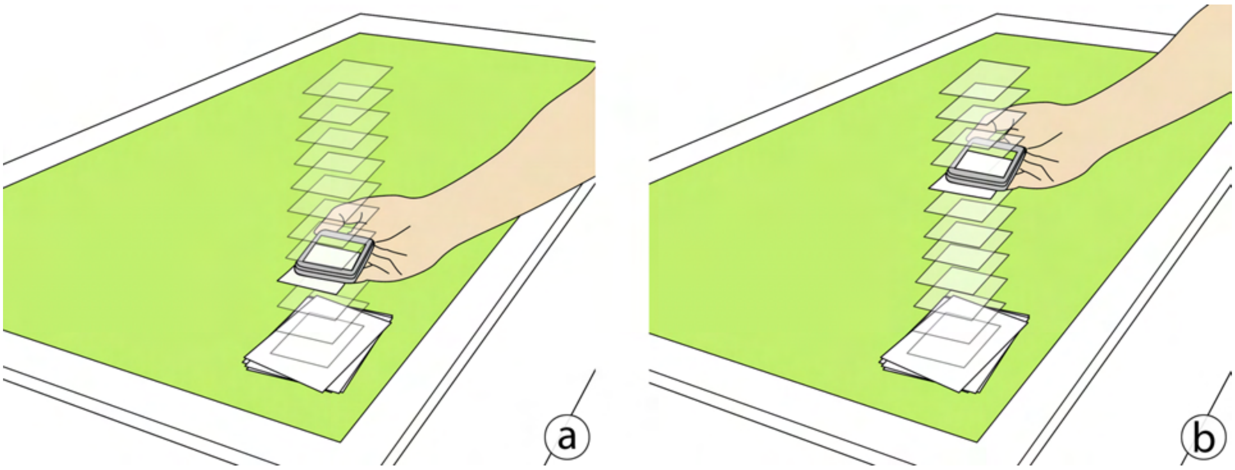


**Figure 8.9:** Using the depth information for pile navigation: (a) The user can see lower items within the pile by moving closer to the display's surface. (b) Vice versa, higher items can be addressed by moving further away.

The pile structure itself further has a huge influence on the interaction. Such piles can have an arbitrary number of items ranging from a few up to thousands of them. As mentioned before, addressing the bottom-most item is done by holding the device at minimal distance to the surface. Navigating towards items that are positioned higher within the pile can be done by moving the device upwards. However, the range within the device can be moved is defined by the person operating it and is naturally limited. The interaction also has a maximal distance above the surface which corresponds to the top-most item in the pile. The more items a pile contains the less mobile device movement is needed to switch from item to the next one. More precisely, piles with thousands of items are hard to navigate assuming that the physical difference between bottom and top is approximately one foot. Nevertheless, piles with only a few items can be easily navigated through using this approach. So far, this type of layered *above-the-surface* interaction has been researched for a fixed number of layers [SAL06, KGS07]. The idea presented here needs a new approach to allow for piles with arbitrary item amounts. One promising solution we currently investigate is to use coarse and fine-grained movement alternately. Users roughly navigate to the position the item should be within the pile and then switch into the precise mode. This then allows for accurately positioning by using the distance relatively. This means that the item shown on the mobile device changes when the user traveled a fixed distance.

## 8.4 Closing Remarks

With the ever increasing number of displays in our everyday life, we are flooded with information more and more. In order to allow users to actively interact with this content rather than just observing it, new interaction models need to be found. Furthermore, the variety of heterogeneous devices and displays forces users to learn specific input interfaces for nearly every task. For example, the input interface for buying a ticket (e.g., touch screen) may be significantly different from the one found on ATMs (e.g., a physical keypad). By using the personal mobile device as input device, the user adaptation can further be avoided. This thesis described one approach to allow users to interact with (possibly) distant displays in public environments. By investigating different scenarios (i.e., pure input versus local and remote content) as well as addressing the mobility of the interaction device, we hope to inform future research in the field of interacting with external, public displays using a mobile device.

# V

# Bibliography and Index

# BIBLIOGRAPHY

[AAK71] Y. I. Abdel-Aziz and H. M. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range Photogrammetry 1971*, pages 1–18. American Society of Photogrammetry, 1971.

[AB06] Anand Agarawala and Ravin Balakrishnan. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *CHI '06: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1283–1292, New York, NY, USA, 2006. ACM.

[AHH$^+$02] Galia Avidan, Michal Harel, Talma Hendler, Dafna Ben-Bashat, Ehud Zohary, and Rafael Malach. Contrast sensitivity in human visual areas and its relationship to object recognition. *The Journal of Neurophysiology*, 87(6):3102–3116, 2002.

[ANSG06] Dzmitry Aliakseyeu, Miguel A. Nacenta, Sriram Subramanian, and Carl Gutwin. Bubble radar: efficient pen-based interaction. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 19–26, New York, NY, USA, 2006. ACM.

[Azu97] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.

[BAB$^+$07] Sebastian Boring, Manuela Altendorfer, Gregor Broll, Otmar Hilliges, and Andreas Butz. Shoot & copy: phonecam-based information transfer from public displays onto mobile phones. In *Mobility '07: Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pages 24–31, New York, NY, USA, 2007. ACM.

[Bal07] Rafael 'Tico' Ballagas. *Bringing Iterative Design to Ubiquitous Computing: Interaction Techniques, Toolkits, and Evaluation Methods*. PhD dissertation, Rheinisch-Westfälische Technische Hochschule, August 2007.

[Bar00] Joel F. Bartlett. Rock 'n' scroll is here to stay. *IEEE Computer Graphics and Applications*, 20:40–45, 2000.

[BB04] Jacob T. Biehl and Brian P. Bailey. Aris: an interface for application relocation in an interactive space. In *GI '04: Proceedings of Graphics Interface 2004*, pages 107–116, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[BB10] Sebastian Boring and Dominikus Baur. Can you see where i point at? In *2nd International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use (in Conjunction with Pervasive 2010)*, pages 7–8, 2010.

[BBB$^+$08] Jacob T. Biehl, William T. Baker, Brian P. Bailey, Desney S. Tan, Kori M. Inkpen, and Mary Czerwinski. Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In *CHI '08: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 939–948, New York, NY, USA, 2008. ACM.

[BBB$^+$10] Sebastian Boring, Dominikus Baur, Andreas Butz, Sean Gustafson, and Patrick Baudisch. Touch projector: Mobile interaction through video. In *CHI '10: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2010. ACM.

[BCC$^+$03] Patrick Baudisch, Edward Cutrell, Mary Czerwinski, Daniel C. Robbins, Peter Tandler, Benjamin B. Bederson, and Alex Zierlinger. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-operated systems. In *INTERACT '03: IFIP TC13 International Conference on Human-Computer Interaction*, pages 57–64, Amsterdam, The Netherlands, 2003. IOS Press.

[BCHG04] Patrick Baudisch, Edward Cutrell, Ken Hinckley, and Robert Gruen. Mouse ether: accelerating the acquisition of targets across multi-monitor displays. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1379–1382, New York, NY, USA, 2004. ACM.

[BCR03] Patrick Baudisch, Edward Cutrell, and George G. Roberston. High-density cursor: a visualization technique that helps users keep track of fast-moving mouse cursors. In *INTERACT '03: IFIP TC13 International Conference on Human-Computer Interaction*, pages 236–243, Amsterdam, The Netherlands, 2003. IOS Press.

[BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[BF98] Bill Buxton and George W. Fitzmaurice. Hmds, caves & chameleon: a human-centric analysis of interaction in virtual space. *SIGGRAPH Comput. Graph.*, 32(4):69–74, 1998.

[BF05] Hrvoje Benko and Steven Feiner. Multi-monitor mouse. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1208–1211, New York, NY, USA, 2005. ACM.

[BF07] Hrvoje Benko and Steven Feiner. Pointer warping in heterogeneous multi-monitor environments. In *GI '07: Proceedings of Graphics Interface 2007*, pages 111–117, New York, NY, USA, 2007. ACM.

[BGBS02] Patrick Baudisch, Nathaniel Good, Victoria Bellotti, and Pamela Schraedley. Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 259–266, New York, NY, USA, 2002. ACM.

[BGS01] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus context screens: combining display technology with visualization techniques. In *UIST '01: Proceedings of the 14th annual ACM symposium on User Interface Software and Technology*, pages 31–40, New York, NY, USA, 2001. ACM.

[BH97] Doug A. Bowman and Larry F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 35–38, New York, NY, USA, 1997. ACM.

[BHB07] Sebastian Boring, Otmar Hilliges, and Andreas Butz. A wall-sized focus plus context display. In *PERCOM '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications*, pages 161–170, Washington, DC, USA, 2007. IEEE Computer Society.

[BHP$^+$08] Gregor Broll, Markus Haarländer, Massimo Paolucci, Matthias Wagner, Enrico Rukzio, and Albrecht Schmidt. Collect&drop: A technique for multi-tag interaction with real world objects and information. In *AmI 2008: European Conference on Ambient Intelligence*, pages 175–191, London, UK, 2008. Springer-Verlag.

[BIF04] Hrvoje Benko, Edward W. Ishak, and Steven Feiner. Collaborative mixed reality visualization of an archaeological excavation. In *ISMAR '04: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 132–140, Washington, DC, USA, 2004. IEEE Computer Society.

[BJB09] Sebastian Boring, Marko Jurmu, and Andreas Butz. Scroll, tilt or move it: using mobile phones to continuously control pointers on large public displays. In *OZCHI '09: Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group*, pages 161–168, New York, NY, USA, 2009. ACM.

[BKHB09] Gregor Broll, Susanne Keck, Paul Holleis, and Andreas Butz. Improving the accessibility of nfc/rfid-based mobile interaction through learnability and guidance. In *MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–10, New York, NY, USA, 2009. ACM.

[BKN$^+$05] Stefan Berger, Rick Kjeldsen, Chandra Narayanaswami, Claudio Pinhanez, Mark Podlaseck, and Mandayam Raghunath. Using symbiotic displays to view sensitive information in public. In *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 139–148, Washington, DC, USA, 2005. IEEE Computer Society.

[BM86] Bill Buxton and Brad A. Myers. A study in two-handed input. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326, New York, NY, USA, 1986. ACM.

[BMK$^+$00] Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven A. Shafer. Easyliving: Technologies for intelligent environments. In *HUC 2000: Second International Symposium on Handheld and Ubiquitous Computing*, pages 12–29, London, UK, 2000. Springer-Verlag.

[BR03] Patrick Baudisch and Ruth Rosenholtz. Halo: a technique for visualizing off-screen objects. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 481–488, New York, NY, USA, 2003. ACM.

[BRP$^+$09] Gregor Broll, Enrico Rukzio, Massimo Paolucci, Matthias Wagner, Albrecht Schmidt, and Heinrich Hussmann. Perci: Pervasive service interaction with the internet of things. *IEEE Internet Computing*, 13:74–81, 2009.

[BRS05] Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Sweep and point and shoot: phonecam-based interactions for large public displays. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1200–1203, New York, NY, USA, 2005. ACM.

[BRSB08] Rafael Ballagas, Michael Rohs, Jennifer Sheridan, and Jan Borchers. The design space of ubiquitous mobile input. In Joanna Lumsden, editor, *Handbook of Research on User Interface Design and Evaluation for Mobile Technologies*. IGI Global, Hershey, PA, USA, 2008.

[BSP$^+$93] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, NY, USA, 1993. ACM.

[BSR$^+$07] Gregor Broll, Sven Siorpaes, Enrico Rukzio, Massimo Paolucci, John Hamard, Matthias Wagner, and Albrecht Schmidt. Supporting mobile service usage through physical mobile interaction. In *PERCOM '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications*, pages 262–271, Washington, DC, USA, 2007. IEEE Computer Society.

[BSW06] Patrick Baudisch, Mike Sinclair, and Andrew Wilson. Soap: a pointing device that works in mid-air. In *UIST '06: Proceedings of the 19th annual ACM symposium on User Interface Software and Technology*, pages 43–46, New York, NY, USA, 2006. ACM.

[CB03] Xiang Cao and Ravin Balakrishnan. Visionwand: interaction techniques for large displays using a passive wand tracked in 3d. In *UIST '03: Proceedings of the 16th annual ACM symposium on User Interface Software and Technology*, pages 173–182, New York, NY, USA, 2003. ACM.

[CCLK05] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy characterization for metropolitan-scale wi-fi localization. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 233–245, New York, NY, USA, 2005. ACM.

[CD06] William Claycomb and Shin Dongwan. Using a two dimensional colorized barcode solution for authentication in pervasive computing. In *PERSER '06: Proceedings of the 2006 ACS/IEEE International Conference on Pervasive Services*, pages 173–180, Washington, DC, USA, 2006. IEEE Computer Society.

[CDF$^+$05] Keith Cheverst, Alan Dix, Daniel Fitton, Chris Kray, Mark Rouncefield, Corina Sas, George Saslis-Lagoudakis, and Jennifer G. Sheridan. Exploring bluetooth based mobile phone interaction with the hermes photo display. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 47–54, New York, NY, USA, 2005. ACM.

[CHBL05] Maxime Collomb, Mountaz Hascoët, Patrick Baudisch, and Brian Lee. Improving drag-and-drop on wall-size displays. In *GI '05: Proceedings of Graphics Interface 2005*, pages 25–32, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[CKRW99] Patrick Chiu, Ashutosh Kapuskar, Sarah Reitmeier, and Lynn Wilcox. Notelook: taking notes in meetings with digital video and ink. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 149–158, New York, NY, USA, 1999. ACM.

[CLB08] Xiang Cao, Jacky Jie Li, and Ravin Balakrishnan. Peephole pointing: modeling acquisition of dynamically revealed targets. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1699–1708, New York, NY, USA, 2008. ACM.

[CMR90] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. The design space of input devices. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 117–124, New York, NY, USA, 1990. ACM.

[DB09] Raimund Dachselt and Robert Buchholz. Natural throw and tilt interaction between mobile phones and distant displays. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3253–3258, New York, NY, USA, 2009. ACM.

[DC02] James Davis and Xing Chen. Lumipoint: multi-user laser-based interaction on large tiled displays. *Displays*, 23(5):205–211, 2002.

[Den00] Denso Wave Inc. Denso Wave Inc., QRCode.com. `http://www.denso-wave.com/qrcode/index-e.html`, 2000. [cited 2010/02/03].

[DH72] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[DKM99] Sarah A. Douglas, Arthur E. Kirkpatrick, and I. Scott MacKenzie. Testing pointing device performance and user assessment with the iso 9241, part 9 standard. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 215–222, New York, NY, USA, 1999. ACM.

[DL01] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User Interface Software and Technology*, pages 219–226. ACM, 2001.

[DLFBH09] Alexander De Luca, Bernhard Frauendienst, Sebastian Boring, and Heinrich Hussmann. My phone is my keypad: privacy-enhanced pin-entry on public terminals. In *OZCHI '09: Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group*, pages 401–404, New York, NY, USA, 2009. ACM.

[DLvZH09] Alexander De Luca, Emanuel von Zezschwitz, and Heinrich Hussmann. Vibrapass: secure authentication based on shared lies. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 913–916, New York, NY, USA, 2009. ACM.

[EB09] Darren Edge and Alan F. Blackwell. Bimanual tangible interaction with mobile phones. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 131–136, New York, NY, USA, 2009. ACM.

[FBB$^+$05] Clifton Forlines, Ravin Balakrishnan, Paul Beardsley, Jeroen van Baar, and Ramesh Raskar. Zoom-and-pick: facilitating visual zooming and precision pointing with interactive handheld projectors. In *UIST '05: Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pages 73–82, New York, NY, USA, 2005. ACM.

[Fit93] George W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Communications of the ACM*, 36(7):39–49, 1993.

[Fit96] George W. Fitzmaurice. *Graspable User Interfaces*. PhD dissertation, Department of Computer Science, University of Toronto, 1996.

[FZC93] George W. Fitzmaurice, Shumin Zhai, and Mark H. Chignell. Virtual reality for palmtop computers. *ACM Transactions on Information Systems (TOIS)*, 11(3):197–218, 1993.

[GB05] Tovi Grossman and Ravin Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–290, New York, NY, USA, 2005. ACM.

[GBGI08] Sean Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang Irani. Wedge: clutter-free visualization of off-screen locations. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 787–796, New York, NY, USA, 2008. ACM.

[Gei98] Jörg Geißler. Shuffle, throw or take it! working efficiently with an interactive wall. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–266, New York, NY, USA, 1998. ACM.

[GER$^+$07] Jürgen Geisler, Ralf Eck, Nils Rehfeld, Elisabeth Peinsipp-Byma, Christian Schütz, and Sven Geggus. Fovea-Tablett®: A new paradigm for the interaction with large screens. In *Human Interface and the Management of Information. Methods, Techniques and Tools in Information Design*, pages 278–287, London, UK, 2007. Springer-Verlag.

[GFG$^+$08] Hans Gellersen, Carl Fischer, Dominique Guinard, Roswitha Gostner, Gerd Kortuem, Christian Kray, Enrico Rukzio, and Sara Streng. Supporting device discovery and spontaneous interaction with spatial references. *Personal and Ubiquitous Computing Journal*, 13(4):255–264, 2008.

[GFG$^+$09] Hans Gellersen, Carl Fischer, Dominique Guinard, Roswitha Gostner, Gerd Kortuem, Christian Kray, Enrico Rukzio, and Sara Streng. Supporting device discovery and spontaneous interaction with spatial references. *Personal Ubiquitous Computing*, 13(4):255–264, 2009.

[GMW05] François Guimbretière, Andrew Martin, and Terry Winograd. Benefits of merging command selection and direct manipulation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(3):460–476, 2005.

[GSF$^+$07] Arjan Geven, Peter Strassl, Bernhard Ferro, Manfred Tscheligi, and Harald Schwab. Experiencing real-world interaction: results from a nfc user experience field trial. In *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 234–237, New York, NY, USA, 2007. ACM.

[Gui87] Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19(4):486–517, 1987.

[Han05] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pages 115–118, New York, NY, USA, 2005. ACM.

[Has03] Mountaz Hascoët. Throwing models for large displays. In *Proceedings of HCI'2003, Designing for Society*, pages 73–77. British HCI Group, 2003.

[HBB07] Otmar Hilliges, Dominikus Baur, and Andreas Butz. Photohelix: Browsing, sorting and sharing digital photo collections. In *Tabletop 2007: Second IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 87–94, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[HCC07] Mark Hancock, Sheelagh Carpendale, and Andy Cockburn. Shallow-depth 3d interaction: design and evaluation of one-, two- and three-touch techniques. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1147–1156, New York, NY, USA, 2007. ACM.

[HCH$^+$09] Chuan-Heng Hsiao, Li-Wei Chan, Ting-Ting Hu, Mon-Chu Chen, Jane Hsu, and Yi-Ping Hung. To move or not to move: a comparison between steerable versus fixed focus region paradigms in multi-resolution tabletop display systems. In *CHI '09: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 153–162, New York, NY, USA, 2009. ACM.

[HCS06] Uta Hinrichs, Sheelagh Carpendale, and Stacey D. Scott. Evaluating the effects of fluid interface components on tabletop collaboration. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 27–34, New York, NY, USA, 2006. ACM.

[HELO05] Thomas Riisgaard Hansen, Eva Eriksson, and Andreas Lykke-Olesen. Mixed interaction space: designing for camera based interaction with mobile devices. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1933–1936, New York, NY, USA, 2005. ACM.

[HIB$^+$07] Steve Hodges, Shahram Izadi, Alex Butler, Alban Rrustemi, and Bill Buxton. Thinsight: versatile multi-touch sensing for thin form-factor displays. In *UIST '07: Proceedings of the 20th annual ACM symposium on User Interface Software and Technology*, pages 259–268, New York, NY, USA, 2007. ACM.

[Hin03] Ken Hinckley. Synchronous gestures for multiple persons and computers. In *UIST '03: Proceedings of the 16th annual ACM symposium on User Interface Software and Technology*, pages 149–158, New York, NY, USA, 2003. ACM.

[HIW$^+$09] Otmar Hilliges, Shahram Izadi, Andrew D. Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. Interactions in the air: adding further depth to interactive tabletops. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User Interface Software and Technology*, pages 139–148, New York, NY, USA, 2009. ACM.

[HJK$^+$10] Simo Hosio, Marko Jurmu, Hannu Kukka, Jukka Riekki, and Timo Ojala. Supporting distributed private and public user interfaces in urban environments. In *HotMobile '10: Proceedings of the Eleventh Workshop on Mobile Computing Systems &#38; Applications*, pages 25–30, New York, NY, USA, 2010. ACM.

[HKI08] Otmar Hilliges, David Kim, and Shahram Izadi. Creating malleable interactive surfaces using liquid displacement sensing. In *TABLETOP 2008: Third IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pages 157–160, Washington, DC, USA, Oct. 2008. IEEE Computer Society.

[HM03] Elaine M. Huang and Elizabeth D. Mynatt. Semi-public displays for small, co-located groups. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 49–56, New York, NY, USA, 2003. ACM.

[HMS$^+$01] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-Werner Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *UbiComp '01: Proceedings of the 3rd International Conference on Ubiquitous Computing*, pages 116–122, London, UK, 2001. Springer-Verlag.

[HOHS07] Paul Holleis, Friederike Otto, Heinrich Hussmann, and Albrecht Schmidt. Keystroke-level model for advanced mobile phone interaction. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1505–1514, New York, NY, USA, 2007. ACM.

[HPGK94] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. A survey of design issues in spatial input. In *UIST '94: Proceedings of the 7th annual ACM symposium on User Interface Software and Technology*, pages 213–222, New York, NY, USA, 1994. ACM.

[HR08] Robert Hardy and Enrico Rukzio. Touch & interact: touch-based interaction of mobile phones with displays. In *MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 245–254, New York, NY, USA, 2008. ACM.

[HRG$^+$04] Ken Hinckley, Gonzalo Ramos, Francois Guimbretiere, Patrick Baudisch, and Marc Smith. Stitching: pen gestures that span multiple displays. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 23–31, New York, NY, USA, 2004. ACM.

[HS80] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1980.

[HWI$^+$07] Juha Häikiö, Arto Wallin, Minna Isomursu, Heikki Ailisto, Tapio Matinmikko, and Tua Huomo. Touch-based user interface for elderly users. In *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 289–296, New York, NY, USA, 2007. ACM.

[IF04] Edward W. Ishak and Steven K. Feiner. Interacting with hidden content using content-aware free-space transparency. In *UIST '04: Proceedings of the 17th annual ACM symposium on User Interface Software and Technology*, pages 189–192, New York, NY, USA, 2004. ACM.

[IHT$^+$08] Shahram Izadi, Steve Hodges, Stuart Taylor, Dan Rosenfeld, Nicolas Villar, Alex Butler, and Jonathan Westhues. Going beyond the display: a surface technology with an electronically switchable diffuser. In *UIST '08: Proceedings of the 21st annual ACM symposium on User Interface Software and Technology*, pages 269–278, New York, NY, USA, 2008. ACM.

[Jac96] Robert J. K. Jacob. Human-computer interaction: input devices. *ACM Computing Surveys*, 28(1):177–179, 1996.

[JBR08a] Marko Jurmu, Sebastian Boring, and Jukka Riekki. Screenspot: multidimensional resource discovery for distributed applications in smart spaces. In *Mobiquitous '08: Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems*, pages 1–9, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[JBR08b] Marko Jurmu, Sebastian Boring, and Jukka Riekki. Screenspot resource discovery for smart spaces and mobilevue media sharing application. In *Mobiquitous '08: Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems*, pages 1–2, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[JF02] Brad Johanson and Armando Fox. The event heap: A coordination infrastructure for interactive workspaces. In *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 83, Washington, DC, USA, 2002. IEEE Computer Society.

[JFW02] Brad Johanson, Armando Fox, and Terry Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1:67–74, 2002.

[JHWS02] Brad Johanson, Greg Hutchins, Terry Winograd, and Maureen Stone. Pointright: experience with flexible input redirection in interactive workspaces. In *UIST '02: Proceedings of the 15th annual ACM symposium on User Interface Software and Technology*, pages 227–234, New York, NY, USA, 2002. ACM.

[JOMS06] Hao Jiang, Eyal Ofek, Neema Moraveji, and Yuanchun Shi. Direct pointer: direct manipulation for large-display interaction using handheld cameras. In *CHI '06: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1107–1110, New York, NY, USA, 2006. ACM.

[JPR07] Marko Jurmu, Mikko Perttunen, and Jukka Riekki. Lease-based resource management in smart spaces. In *PerCom Workshops*, pages 622–626, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[KB94] Gordon Kurtenbach and William Buxton. User learning and performance with marking menus. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 258–264, New York, NY, USA, 1994. ACM.

[KB99] Hirokazu Kato and Mark Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 85–94, Washington, DC, USA, 1999. IEEE Computer Society.

[KBS94] Paul Kabbash, William Buxton, and Abigail Sellen. Two-handed input in a compound task. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 417–423, New York, NY, USA, 1994. ACM.

[KFA$^+$04] Azam Khan, George Fitzmaurice, Don Almeida, Nicolas Burtnyk, and Gordon Kurtenbach. A remote control interface for large displays. In *UIST '04: Proceedings of the 17th annual ACM symposium on User Interface Software and Technology*, pages 127–136, New York, NY, USA, 2004. ACM.

[KGS07] Raghavendra S. Kattinakere, Tovi Grossman, and Sriram Subramanian. Modeling steering within above-the-surface interaction layers. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 317–326, New York, NY, USA, 2007. ACM.

[KH04] John Krumm and Ken Hinckley. The nearme wireless proximity server. In *UbiComp '06: Proceedings of the 6th International Conference on Ubiquitous Computing*, pages 283–300, London, UK, 2004. Springer-Verlag.

[KKG05] Gerd Kortuem, Christian Kray, and Hans Gellersen. Sensing and visualizing spatial relations of mobile devices. In *UIST '05: Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pages 93–102, New York, NY, USA, 2005. ACM.

[KKM$^+$06] Juha Kela, Panu Korpipää, Jani Mäntyjärvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo, and Sergio Di Marca. Accelerometer-based gesture control for a design environment. *Personal Ubiquitous Computing*, 10(5):285–299, 2006.

[KM98] Carsten Kirstein and Heinrich Müller. Interaction with a projection screen using a camera-tracked laser pointer. *International Multi-Media Modeling Conference*, 0:191–192, 1998.

[KRA01] Naohiko Kohtake, Jun Rekimoto, and Yuichiro Anzai. Infopoint: A device that provides a uniform user interface to allow appliances to work together over a network. *Personal Ubiquitous Comput.*, 5(4):264–274, 2001.

[LA94] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact.*, 1(2):126–160, 1994.

[LC08] Alison Lee and Umesh Chandra. Enabling meetings for "anywhere and anytime". In *CollaborateCom 2008: Proceedings of the 4th International Conference of Collaborative Computing: Networking, Applications and Worksharing*, pages 792–804, London, UK, 2008. Springer-Verlag.

[LHGL05] Yang Li, Ken Hinckley, Zhiwei Guan, and James A. Landay. Experimental analysis of mode switching techniques in pen-based user interfaces. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 461–470, New York, NY, USA, 2005. ACM.

[LK81] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: Proceedings of the 7th international joint conference on Artificial intelligence*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[LKC05] Celine Latulipe, Craig S. Kaplan, and Charles L. A. Clarke. Bimanual and unimanual image alignment: an evaluation of mouse-based techniques. In *UIST '05: Proceedings of the 18th annual ACM symposium on User Interface Software and Technology*, pages 123–131, New York, NY, USA, 2005. ACM.

[LLK$^+$03] Chunyuan Liao, Qiong Liu, Don Kimber, Patrick Chiu, Jonathan Foote, and Lynn Wilcox. Shared interactive video for teleconferencing. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 546–554, New York, NY, USA, 2003. ACM.

[Low99] David G. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, pages 1150–1157, Los Alamitos, CA, USA, 1999. IEEE Computer Society.

[MBGH07] Kaj Mäkelä, Sara Belt, Dan Greenblatt, and Jonna Häkkilä. Mobile interaction with visual and rfid tags: a field study on user perceptions. In *CHI '07: Proceedings of* *the SIGCHI conference on Human factors in computing systems*, pages 991–994, New York, NY, USA, 2007. ACM.

[MBN+02] Brad A. Myers, Rishi Bhatnagar, Jeffrey Nichols, Choon Hong Peck, Dave Kong, Robert Miller, and A. Chris Long. Interacting at a distance: measuring the performance of laser pointers and other devices. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 33–40, New York, NY, USA, 2002. ACM.

[MC02] Michael Moyle and Andy Cockburn. Analysing mouse and pen flick gestures. In *In Proceedings of the SIGCHI-NZ Symposium on Computer-Human Interaction*, pages 266–267, 2002.

[Met01] Kaspar Metz. Project Blinkenlights. `http://www.blinkenlights.net/`, 2001. [cited 2010/02/10].

[MHT04] Kento Miyaoku, Suguru Higashino, and Yoshinobu Tonomura. C-blink: a hue-difference-based light signal marker for large screen interaction via any mobile terminal. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 147–156, New York, NY, USA, 2004. ACM.

[MI09] David C. McCallum and Pourang Irani. Arc-pad: absolute+relative cursor positioning for large displays with a mobile touchscreen. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User Interface Software and Technology*, pages 153–156, New York, NY, USA, 2009. ACM.

[MMH07] Andrew Maunder, Gary Marsden, and Richard Harper. Creating and sharing multimedia packages using large situated public displays and mobile phones. In *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 222–225, New York, NY, USA, 2007. ACM.

[MSG98] Brad A. Myers, Herb Stiel, and Robert Gargiulo. Collaboration using multiple pdas connected to a pc. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 285–294, New York, NY, USA, 1998. ACM.

[MSSU04] Anil Madhavapeddy, David Scott, Richard Sharp, and Eben Upton. Using cameraphones to enhance human-computer interaction. In *Sixth International Conference on Ubiquitous Computing (Adjunct Proceedings: Demos)*, 2004.

[MSW92] Richard Mander, Gitta Salomon, and Yin Yin Wong. A ”pile” metaphor for supporting casual organization of information. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 627–634, New York, NY, USA, 1992. ACM.

[MWW06] Sumit Mehra, Peter Werkhoven, and Marcel Worring. Navigating on handheld displays: Dynamic versus static peephole navigation. *ACM Trans. Comput.-Hum. Interact.*, 13(4):448–457, 2006.

[NASG05] Miguel A. Nacenta, Dzmitry Aliakseyeu, Sriram Subramanian, and Carl Gutwin. A comparison of techniques for multi-display reaching. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 371–380, New York, NY, USA, 2005. ACM.

[NGAS09] Miguel Nacenta, Carl Gutwin, Dzimitri Aliakseyeu, and Sriram Subramanian. There and back again: Cross-display object movement in multi-display environments. *Journal of Human-Computer Interaction*, 24(1):170–229, 2009.

[Nin06] Nintendo Inc. Wii at Nintendo. `http://www.nintendo.com/wii`, 2006. [cited 2010/02/10].

[NMG08] Miguel A. Nacenta, Regan L. Mandryk, and Carl Gutwin. Targeting across displayless space. In *CHI '08: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 777–786, New York, NY, USA, 2008. ACM.

[NSC$^+$06] Miguel A. Nacenta, Samer Sallam, Bernard Champoux, Sriram Subramanian, and Carl Gutwin. Perspective cursor: perspective-based interaction for multi-display environments. In *CHI '06: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 289–298, New York, NY, USA, 2006. ACM.

[OF09] Alex Olwal and Steven Feiner. Spatially aware handhelds for high-precision tangible interaction with large displays. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 181–188, New York, NY, USA, 2009. ACM.

[Olw06] Alex Olwal. Lightsense: enabling spatially aware handheld interaction devices. In *ISMAR '06: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 119–122, Washington, DC, USA, 2006. IEEE Computer Society.

[ON01] Dan R. Olsen, Jr. and Travis Nielsen. Laser pointer interaction. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–22, New York, NY, USA, 2001. ACM.

[OTGW07] Eamonn O'Neill, Peter Thompson, Stavros Garzonis, and Andrew Warr. Reach out and touch: Using nfc and 2d barcodes for service discovery and interaction with mobile devices. In *Pervasive 2007: Proceedings of the 5th International Conference on Pervasive Computing*, pages 19–36, London, UK, 2007. Springer-Verlag.

[PBWI96] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *UIST*

*'96: Proceedings of the 9th annual ACM symposium on User Interface Software and Technology*, pages 79–80, New York, NY, USA, 1996. ACM.

[PFC$^+$97] Jeffrey S. Pierce, Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. Image plane interaction techniques in 3d immersive environments. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 39–43, New York, NY, USA, 1997. ACM.

[PJO09] Nick Pears, Daniel Jackson, and Patrick Olivier. Smart phone interaction with registered displays. *IEEE Pervasive Computing*, 8(2):14–21, 2009.

[PKS$^+$08] Peter Peltonen, Esko Kurvinen, Antti Salovaara, Giulio Jacucci, Tommi Ilmonen, John Evans, Antti Oulasvirta, and Petri Saarikko. It's mine, don't touch!: interactions at a large multi-touch display in a city centre. In *CHI '08: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1285–1294, New York, NY, USA, 2008. ACM.

[PMI05] J. Karen Parker, Regan L. Mandryk, and Kori M. Inkpen. Tractorbeam: seamless integration of local and remote pointing for tabletop displays. In *GI '05: Proceedings of Graphics Interface 2005*, pages 33–40, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[PRA06] Shwetak N. Patel, Jun Rekimoto, and Gregory D. Abowd. icam: Precise at-a-distance interaction in the physical environment. In *Pervasive 2006: Proceedings of the 4th International Conference on Pervasive Computing*, pages 272–287, London, UK, 2006. Springer-Verlag.

[PSJ$^+$07] Peter Peltonen, Antti Salovaara, Giulio Jacucci, Tommi Ilmonen, Carmelo Ardito, Petri Saarikko, and Vikram Batra. Extending large-scale event participation with user-created mobile media on a public display. In *MUM '07: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pages 131–138, New York, NY, USA, 2007. ACM.

[QBG08] Till Quack, Herbert Bay, and Luc J. Van Gool. Object recognition for the internet of things. In *IOT 2008: Proceedings of the First International Conference on The Internet of Things*, pages 230–246, London, UK, 2008. Springer-Verlag.

[RAK03] Jun Rekimoto, Yuji Ayatsuka, and Michimune Kohno. Synctap: An interaction technique for mobile networking. In *Mobile HCI '03: 5th International Symposium of Human-Computer Interaction with Mobile Devices and Services*, pages 104–115, London, UK, 2003. Springer-Verlag.

[RAKO03] Jun Rekimoto, Yuji Ayatsuka, Michimune Kohno, and Haruo Oba. Proximal interactions: A direct manipulation technique for wireless networking. In *INTERACT '03: IFIP TC13 International Conference on Human-Computer Interaction*, pages 511–518. IOS Press, 2003.

[RCB$^+$05] George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, and Desney Tan. The large-display user experience. *IEEE Computer Graphics and Applications*, 25:44–51, 2005.

[RCL$^+$98] George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. Data mountain: using spatial memory for document management. In *UIST '98: Proceedings of the 11th annual ACM symposium on User Interface Software and Technology*, pages 153–162, New York, NY, USA, 1998. ACM.

[Rek96] Jun Rekimoto. Tilting operations for small screen interfaces. In *UIST '96: Proceedings of the 9th annual ACM symposium on User Interface Software and Technology*, pages 167–168, New York, NY, USA, 1996. ACM.

[Rek97] Jun Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *UIST '97: Proceedings of the 10th annual ACM symposium on User Interface Software and Technology*, pages 31–39, New York, NY, USA, 1997. ACM.

[Rek02] Jun Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 113–120, New York, NY, USA, 2002. ACM.

[RGS$^+$06] Adrian Reetz, Carl Gutwin, Tadeusz Stach, Miguel Nacenta, and Sriram Subramanian. Superflick: a natural and efficient technique for long-distance object placement on digital tables. In *GI '06: Proceedings of Graphics Interface 2006*, pages 163–170, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[RLC$^+$06] Enrico Rukzio, Karin Leichtenstern, Vic Callaghan, Paul Holleis, Albrecht Schmidt, and Jeanette Chin. An experimental comparison of physical mobile interaction techniques: Touching, pointing and scanning. In *UbiComp '06: Proceedings of the 8th International Conference on Ubiquitous Computing*, pages 87–104, London, UK, 2006. Springer-Verlag.

[RN95] Jun Rekimoto and Katashi Nagao. The world through the computer: computer augmented interaction with real world environments. In *UIST '95: Proceedings of the 8th annual ACM symposium on User Interface Software and Technology*, pages 29–36, New York, NY, USA, 1995. ACM.

[RO08] Michael Rohs and Antti Oulasvirta. Target acquisition with camera phones when used as magic lenses. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1409–1418, New York, NY, USA, 2008. ACM.

[Roh04] Michael Rohs. Real-world interaction with camera phones. In *UCS '04: Proceedings of the 2nd International Symposium on Ubiquitous Computing Systems*, pages 74–89, London, UK, 2004. Springer-Verlag.

[RRA$^+$06] Derek Reilly, Malcolm Rodgers, Ritchie Argue, Mike Nunes, and Kori Inkpen. Marked-up maps: combining paper maps and electronic information resources. *Personal Ubiquitous Computing*, 10(4):215–226, 2006.

[RRC$^+$06] Gonzalo Ramos, George Robertson, Mary Czerwinski, Desney Tan, Patrick Baudisch, Ken Hinckley, and Maneesh Agrawala. Tumble! splat! helping users access and manipulate occluded content in 2d drawings. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 428–435, New York, NY, USA, 2006. ACM.

[RS99] Jun Rekimoto and Masanori Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 378–385, New York, NY, USA, 1999. ACM.

[RSA06] Jukka Riekki, Timo Salminen, and Ismo Alakärppä. Requesting pervasive services by touching rfid tags. *IEEE Pervasive Computing*, 5:40–46, 2006.

[RSH04] Enrico Rukzio, Albrecht Schmidt, and Heinrich Hussmann. Physical posters as gateways to context-aware services for mobile devices. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '04)*, pages 10–19, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[RSK05] Enrico Rukzio, Albrecht Schmidt, and Antonio Krüger. The rotating compass: a novel interaction technique for mobile navigation. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1761–1764, New York, NY, USA, 2005. ACM.

[RSKH07] Michael Rohs, Johannes Schöning, Antonio Krüger, and Brent Hecht. Towards real-time markerless tracking of magic lenses on paper maps. In *Adjunct Proceedings of the 5th Intl. Conference on Pervasive Computing (Pervasive), Late Breaking Results*, pages 69–72, 2007.

[RSR$^+$07] Michael Rohs, Johannes Schöning, Martin Raubal, Georg Essl, and Antonio Krüger. Map navigation with mobile devices: virtual versus physical movement with and without visual context. In *ICMI '07: Proceedings of the 9th international conference on Multimodal interfaces*, pages 146–153, New York, NY, USA, 2007. ACM.

[RWDBI05] Derek Reilly, Michael Welsman-Dinelle, Colin Bate, and Kori Inkpen. Just point and click?: using handhelds to interact with paper maps. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 239–242, New York, NY, USA, 2005. ACM.

[SAL06] Sriram Subramanian, Dzimitry Aliakseyeu, and Andrés Lucero. Multi-layer interaction for digital tables. In *UIST '06: Proceedings of the 19th annual ACM symposium on User Interface Software and Technology*, pages 269–272, New York, NY, USA, 2006. ACM.

[SCP95] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual reality on a wim: interactive worlds in miniature. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[SCR07] Iván Sánchez, Marta Cortés, and Jukka Riekki. Controlling multimedia players using nfc enabled mobile phones. In *MUM '07: Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia*, pages 118–124, New York, NY, USA, 2007. ACM.

[SDP$^+$09] Alireza Sahami Shirazi, Tanja Döring, Pouyan Parvahan, Bernd Ahrens, and Albrecht Schmidt. Poker surface: combining a multi-touch table and mobile phones in interactive card games. In *MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–2, New York, NY, USA, 2009. ACM.

[SGH$^+$99] Norbert A. Streitz, Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-land: an interactive landscape for creativity and innovation. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 120–127, New York, NY, USA, 1999. ACM.

[SH06] Johan Sanneblad and Lars Erik Holmquist. Ubiquitous graphics: combining handheld and wall-size displays to interact with large images. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 373–377, New York, NY, USA, 2006. ACM.

[SHII09] Daisuke Sakamoto, Koichiro Honda, Masahiko Inami, and Takeo Igarashi. Sketch and run: a stroke-based interface for home robots. In *CHI '09: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 197–200, New York, NY, USA, 2009. ACM.

[Shn83] Ben Shneiderman. Direct manipulation: A step beyond programming languages (abstract only). *IEEE Computer*, 16(8):57–69, 1983.

[SHS$^+$09] Thomas Seifried, Michael Haller, Stacey D. Scott, Florian Perteneder, Christian Rendl, Daisuke Sakamoto, and Masahiko Inami. Cristal: Design and implementation of a remote control system based on a multi-touch display. In *ITS 2009: The ACM International Conference on Interactive Tabletops and Surfaces 2009*, pages 37–44, New York, NY, USA, 2009. ACM.

[SIDT02] Colin Swindells, Kori M. Inkpen, John C. Dill, and Melanie Tory. That one there! pointing to establish device identity. In *UIST '02: Proceedings of the 15th annual ACM symposium on User Interface Software and Technology*, pages 151–160, New York, NY, USA, 2002. ACM.

[SKM06] Johannes Schöning, Antonio Krüger, and Hans Jörg Müller. Interaction of mobile camera devices with physical maps. In *Adjunct Proceeding of the Fourth International Conference on Pervasive Computing*, pages 121–124, 2006.

[SMA10] SMART Technologies. (DViT) Digital Vision Touch Technology. `http://smarttech.com/DViT/`, 2010. [cited 2010/03/24].

[SMK01] Miika Silfverberg, I. Scott MacKenzie, and Tatu Kauppinen. An isometric joystick as a pointing device for handheld information terminals. In *GI '01: Proceedings of Graphics Interface 2001*, pages 119–126, Toronto, Ont., Canada, Canada, 2001. Canadian Information Processing Society.

[SOC08] Jürgen Scheible, Timo Ojala, and Paul Coulton. Mobitoss: a novel gesture based interface for creating and sharing mobile multimedia art on large public displays. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 957–960, New York, NY, USA, 2008. ACM.

[SRHH09] Khoovirajsingh Seewoonauth, Enrico Rukzio, Robert Hardy, and Paul Holleis. Touch & connect and touch & select: interacting with a computer by touching it with a mobile phone. In *MobileHCI '09: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–9, New York, NY, USA, 2009. ACM.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600, Los Alamitos, CA, USA, 1994. IEEE Computer Society.

[STB07] Garth Shoemaker, Anthony Tang, and Kellogg S. Booth. Shadow reaching: a new perspective on interaction for large displays. In *UIST '07: Proceedings of the 20th annual ACM symposium on User Interface Software and Technology*, pages 53–56, New York, NY, USA, 2007. ACM.

[Tar08] Sasu Tarkoma. Fuego toolkit: a modular framework for content-based routing. In *DEBS '08: Proceedings of the second international conference on Distributed event-based systems*, pages 325–328, New York, NY, USA, 2008. ACM.

[TFK$^+$02] Michael Tsang, George W. Fitzmaurice, Gordon Kurtenbach, Azam Khan, and Bill Buxton. Boom chameleon: simultaneous capture of 3d viewpoint, voice and gesture annotations on a spatially-aware display. In *UIST '02: Proceedings of the 15th annual ACM symposium on User Interface Software and Technology*, pages 111–120, New York, NY, USA, 2002. ACM.

[TG04] Edward Tse and Saul Greenberg. Rapidly prototyping single display groupware through the sdgtoolkit. In *AUIC '04: Proceedings of the fifth conference on Australasian user interface*, pages 101–110, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[THY$^+$94] Masayuki Tani, Masato Horita, Kimiya Yamaashi, Koichiro Tanikoshi, and Masayasu Futakawa. Courtyard: integrating shared overview on a large screen and per-user detail on individual screens. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 44–50, New York, NY, USA, 1994. ACM.

[TKW08] Georg Treu, Axel Küpper, and Thomas Wilder. Extending the lbs-framework trax: Efficient proximity detection with dead reckoning. *Computer Communications*, 31(5):1040–1051, 2008.

[TMC04] Desney S. Tan, Brian Meyers, and Mary Czerwinski. Wincuts: manipulating arbitrary window regions for more effective use of screen space. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1525–1528, New York, NY, USA, 2004. ACM.

[TPMT$^+$01] Peter Tandler, Thorsten Prante, Christian Müller-Tomfelde, Norbert Streitz, and Ralf Steinmetz. Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In *UIST '01: Proceedings of the 14th annual ACM symposium on User Interface Software and Technology*, pages 11–20, New York, NY, USA, 2001. ACM.

[TYT$^+$92] Masayuki Tani, Kimiya Yamaashi, Koichiro Tanikoshi, Masayasu Futakawa, and Shinya Tanifuji. Object-oriented video: interaction with real-world objects through live video. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 593–598, New York, NY, USA, 1992. ACM.

[UIG98] Brygg Ullmer, Hiroshi Ishii, and Dylan Glas. mediablocks: physical containers, transports, and controls for online media. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 379–386, New York, NY, USA, 1998. ACM.

[VB04] Daniel Vogel and Ravin Balakrishnan. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *UIST '04: Proceedings of the 17th annual ACM symposium on User Interface Software and Technology*, pages 137–146, New York, NY, USA, 2004. ACM.

[VB07] Daniel Vogel and Patrick Baudisch. Shift: a technique for operating pen-based interfaces using touch. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 657–666, New York, NY, USA, 2007. ACM.

[VCBE08] Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards. Using a mobile phone as a ”wii-like” controller for playing games on a large display. *International Journal of Computer Games Technology*, 2008(145–150), 2008.

[WFB$^+$07] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. Lucid touch: a see-through mobile device. In *UIST ’07: Proceedings of the 20th annual ACM symposium on User Interface Software and Technology*, pages 269–278, New York, NY, USA, 2007. ACM.

[WFGH99] Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging physical and virtual worlds with electronic tags. In *CHI ’99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 370–377, New York, NY, USA, 1999. ACM.

[WIH$^+$08] Andrew D. Wilson, Shahram Izadi, Otmar Hilliges, Armando Garcia-Mendoza, and David Kirk. Bringing physics to the surface. In *UIST ’08: Proceedings of the 21st annual ACM symposium on User Interface Software and Technology*, pages 67–76, New York, NY, USA, 2008. ACM.

[WJF$^+$09] Daniel Wigdor, Hao Jiang, Clifton Forlines, Michelle Borkin, and Chia Shen. Wespace: the design development and deployment of a walk-up and share multi-surface visual collaboration system. In *CHI ’09: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1237–1246, New York, NY, USA, 2009. ACM.

[WLS08] Daniel Wagner, Tobias Langlotz, and Dieter Schmalstieg. Robust and unobtrusive marker tracking on mobile phones. In *ISMAR ’08: Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 121–124, Washington, DC, USA, 2008. IEEE Computer Society.

[WMI08] James R. Wallace, Regan L. Mandryk, and Kori M. Inkpen. Comparing content and input redirection in mdes. In *CSCW ’08: Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 157–166, New York, NY, USA, 2008. ACM.

[WPLS05] Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg. Towards massively multi-user augmented reality on handheld devices. In *Pervasive 2005: Proceedings of the 3rd International Conference on Pervasive Computing*, pages 208–219, London, UK, 2005. Springer-Verlag.

[WS03] Andrew Wilson and Steven Shafer. Xwand: Ui for intelligent spaces. In *CHI ’03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 545–552, New York, NY, USA, 2003. ACM.

[WS07] Andrew D. Wilson and Raman Sarin. Bluetable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In *GI ’07: Proceedings of Graphics Interface 2007*, pages 119–125, New York, NY, USA, 2007. ACM.

[Yee03a] Ka-Ping Yee. Interaction techniques and applications for peephole displays. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 636–637, New York, NY, USA, 2003. ACM.

[Yee03b] Ka-Ping Yee. Peephole displays: pen interaction on spatially aware handheld computers. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1–8, New York, NY, USA, 2003. ACM.

[YTH$^+$05] Koji Yatani, Koiti Tamura, Keiichi Hiroki, Masanori Sugimoto, and Hiromichi Hashizume. Toss-it: intuitive information transfer techniques for mobile devices. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1881–1884, New York, NY, USA, 2005. ACM.

# INDEX