Diss. ETH No. 16193

Linking Physical and Virtual Worlds with Visual Markers and Handheld Devices

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH

for the degree of
Doctor of Sciences

presented by
MICHAEL ROHS
Diplom-Informatiker, Darmstadt University of Technology
born January 13, 1974
citizen of Germany

accepted on the recommendation of
Prof. Dr. Friedemann Mattern, examiner
Dr. Albrecht Schmidt, co-examiner

2005

Acknowledgments

First and foremost, I would like to thank my supervisor Prof. Friedemann Mattern for having given me the opportunity to work in his research group in the fascinating field of ubiquitous computing. During my time at ETH Zurich I learned a lot about research, teaching, and project work. I greatly appreciate the balance between the freedom in choosing my dissertation topic and the guidance in carrying out my research that Prof. Mattern provided. I further appreciate the readiness of Dr. Albrecht Schmidt to be my co-examiner and his constructive remarks on this thesis.

I am particularly grateful to all the fantastic students who did their semester projects, diploma, and masters theses within the scope of this work. These students are Kaspar Baltzer, Marcel Beer, Biörn Biörnstad, Nicolas Born, Sabine Do-Thuong, Beat Gfeller, Manuel Graber, Patrick Jayet, Nikolaos Kaintantzis, Erich Laube, Jean-Daniel Merkli, and Philipp Zweifel.

The fruitful cooperation with Tico Ballagas from RWTH Aachen and Jenn Sheridan from Lancaster University on interaction techniques for large displays was a very rewarding experience. Doing the usability studies at Lancaster University was really fun. I would also like to express my gratitude to Prof. Hans Gellersen and his group at Lancaster University, in particular Mike Hazas, Gerd Kortuem, Christian Kray, Kristof Van Laerhoven, David Molyneaux, Martin Strohbach, and Nicolas Villar for interesting discussions during my visits to Lancaster.

I thank my colleagues Ruedi Arnold, Jürgen Bohn, Vlad Coroama, Svetlana Domnitcheva, Christian Flörkemeier, Christian Frank, Oliver Kasten, Matthias Lampe, Marc Langheinrich, Matthias Ringwald, Christof Roduner, Kay Römer, Silvia Santini, Thomas Schoch, Frank Siegemund, and Harald Vogt for contributing to a very positive and inspiring working atmosphere and for being knowledgeable and constructive discussion partners and co-workers.

My deepest gratitude goes to Kirsten Mara, who became my wife between writing the third and the fourth quarter of this dissertation. Thank you for sharing my ups and downs during my life as a Ph.D. student and for bearing with me, especially in the final rush of writing. Finally, I dedicate this thesis to my parents. They made my education possible and provided me with constant love and encouragement.

Contents

Abstract

Linking the physical and the virtual world is a major research theme of ubiquitous and pervasive computing. This dissertation describes concepts and techniques for linking information and services to physical objects as well as for interacting with this information using mobile devices and embodied user interfaces. Such interfaces use gestures on the device body as a means of input. In the recent past, there have been considerable research efforts in linking computation to physical objects. However, these projects were mainly concerned with the physical linking technology per se or with the infrastructure required for identifier resolution. Other work on manipulative and embodied user interfaces focused on improving interaction with a handheld device itself, but did not integrate physical objects of the user's environment. In our work, we combine physical linking and embodied interaction and allow the interaction semantics to be a function of the object and the gestural sequence.

The proposed approach uses camera phones and similar devices as mobile sensors for two-dimensional visual markers. We not only retrieve the value that is encoded in the marker, but also detect the spatial orientation of the device relative to the marker in real time. We use the detected orientation for embodied interaction with the device and augment the live camera image according to the orientation with graphical overlays. By providing a video see-through augmented reality view on the background, the handheld device embodies a "symbolic magnifying glass." This allows for fine-grained interaction and enhances the currently limited input capabilities of mobile devices. We call this approach *marker-based interaction*. It turns camera phones and similar devices into versatile interfaces to – and mediators for – real-world objects.

In this thesis, we present a system for recognizing two-dimensional visual markers. The markers we developed are called *visual codes*. The recognition system provides a number of parameters for determining the spatial orientation of the device relative to the marker, such as the target point in code coordinates, rotation, tilting, distance, and movement of the device relative to the background. It is specifically designed for the requirements of mobile phones with limited computing capabilities and low resolution cameras. Moreover, the system provides the basis for augmenting objects in the live camera image with precisely aligned graphical overlays. Based on this foundation we have developed several mechanisms and concepts for marker-based interaction, namely: (1) a framework of physical interaction primitives, (2) marker-based interface elements, called *visual code widgets*, (3) interaction techniques for large-scale displays, and (4) handheld augmented reality applications.

Our conceptual framework of physical interaction primitives enables the use of camera-equipped mobile devices as embodied user interfaces, in which users

can specify input through physical manipulations and orientation changes of the device. The framework defines a set of fundamental physical gestures that form a basic vocabulary for describing interaction when using mobile devices capable of reading visual codes. These interaction primitives can be combined to create more complex and expressive interactions. The interaction primitives and their combinations have been evaluated in a usability study.

In comparison to interaction primitives, visual code widgets operate at a higher level of abstraction. Visual code widgets are printable elements of physical user interfaces, comparable to the interactive elements of conventional graphical user interfaces. Each widget type addresses a particular input problem and encapsulates a specific behavior and functionality. Visual code widgets thus define building blocks for applications that incorporate mobile devices as well as resources in the user's environment, such as paper documents, posters, and public electronic displays.

For large-scale displays, we have developed two interaction techniques that rely on visual movement detection and visual code recognition, respectively. The first one enables relative positioning of a cursor and is suited for direct manipulation of objects that are visible on the screen. The second one allows for absolute positioning on the screen and can be used for the selection of displayed objects. Both techniques have been evaluated in a qualitative usability study and are especially useful for displays that are not available for direct touch-based interaction, such as displays in public spaces.

The concepts and techniques that were developed in the scope of this dissertation have been investigated in various application areas. Examples that are detailed in the dissertation are: entry points into a smart campus environment, augmented board games, an interactive photo wall, a collaborative game for large-scale displays, digital annotations of physical objects, and smart product packaging.

Zusammenfassung

Die Verbindung der physischen mit der virtuellen Welt ist ein zentrales Forschungsthema im Bereich des Ubiquitous und Pervasive Computing. Die vorliegende Dissertation beschreibt Konzepte und Techniken zur Verknüpfung von physischen Objekten mit Information sowie zur Interaktion mit diesen Informationen mittels mobiler Geräte und gestenbasierter Benutzungsschnittstellen. In jüngster Vergangenheit ist die Verknüpfung von physischen Objekten mit digitalen Informationen bereits intensiv erforscht worden. Vorausgegangene Arbeiten haben sich allerdings hauptsächlich mit der Verknüpfungstechnologie oder mit Infrastrukturaspekten beschäftigt. Andere Arbeiten haben sich der Verbesserung der Interaktion mit mobilen Geräten durch physische Manipulationen gewidmet, jedoch keine Objekte in der Umgebung des Benutzers mit einbezogen. In der vorliegenden Arbeit werden physische Verknüpfungen und gestenbasierte Benutzungsschnittstellen kombiniert, wodurch die Interaktionssemantik eine Funktion des Objektes und der physischen Handhabung des mobilen Gerätes wird.

Der vorgeschlagene Ansatz nutzt Mobiltelefone und ähnliche Geräte mit integrierter Kamera als mobile Sensoren für zweidimensionale visuelle Marker. Es werden nicht nur die im Marker gespeicherten Daten, sondern auch die räumliche Orientierung des Gerätes relativ zum Marker in Echtzeit ermittelt. Die erkannte Orientierung wird zur physischen Interaktion mit dem Gerät genutzt und das Kamerabild entsprechend der Orientierung mit grafischen Ausgaben überlagert. Das Gerät liefert eine um Informationen erweiterte Sicht auf den mit der Kamera fokussierten Bereich und implementiert damit die Metapher einer „symbolischen Lupe“. Dies ermöglicht eine fein-granulare Interaktion und erweitert die bisher eingeschränkten Eingabemöglichkeiten von mobilen Geräten. Wir bezeichnen diesen Ansatz als *Marker-basierte Interaktion*. Er verwandelt mobile Geräte in vielseitig verwendbare Schnittstellen und Mediatoren für Objekte der realen Welt.

In der vorliegenden Arbeit wird ein System zur Erkennung von zweidimensionalen visuellen Markern vorgestellt. Die von uns entwickelten Marker bezeichnen wir als *Visual Codes*. Das Erkennungssystem liefert verschiedene Parameter zur Ermittlung der räumlichen Orientierung der Kamera relativ zum Marker, wie den anvisierten Punkt in der Ebene des Markers, die Rotation, die Neigung, die Distanz und die Bewegung des Gerätes relativ zum Hintergrund. Das System wurde speziell für die Anforderungen von Mobiltelefonen mit eingeschränkter Rechenkapazität und gering auflösender integrierter Kamera entworfen. Es dient darüber hinaus als Basis für die Erzeugung grafischer Einblendungen im Kamerabild, die einzelne Bildelemente geometrisch präzise überlagern. Aufbauend auf dieser Basis wurden verschiedene Mechanismen und Konzepte für Marker-basierte Interaktion entwickelt: (1) ein System kombinierbarer physischer Interaktionsprimitive, (2) Marker-basierte Bausteine für physische Benutzungsschnittstellen, sogenannte

Visual-Code-Widgets, (3) Interaktionstechniken für Grossbildschirme und (4) mobile Augmented-Reality-Anwendungen.

Das System kombinierbarer physischer Interaktionsprimitive erlaubt die Nutzung mobiler Geräte mit integrierter Kamera als *embodied user interfaces*, bei denen der Benutzer Eingaben mit Hilfe von physischen Manipulationen und Orientierungsänderungen des Gerätes vornehmen kann. Die Interaktionsprimitive definieren dabei eine Menge von grundlegenden physischen Gesten, die ein Basis-Vokabular zur Interaktion mit visuellen Markern mittels Kamera-Mobiltelefonen und ähnlichen Geräten darstellen. Die Interaktionsprimitive können kombiniert werden, um komplexere und ausdrucksstärkere Interaktionen zu ermöglichen. Die Eigenschaften der verschiedenen Interaktionsprimitive und ihrer Kombinationen wurden anhand einer Nutzerstudie evaluiert.

Im Vergleich zu den Interaktionsprimitiven befinden sich die Visual-Code-Widgets auf einer höheren Abstraktionsstufe. Visual-Code-Widgets sind auf Papier druckbare Elemente physischer Benutzungsschnittstellen, vergleichbar den interaktiven Elementen herkömmlicher grafischer Benutzungsschnittstellen. Jeder Widget-Typ definiert ein bestimmtes Eingabeparadigma, kapselt eine bestimmte Funktionalität und zeigt gegenüber dem Benutzer ein charakteristisches Verhalten. Visual-Code-Widgets stellen damit Grundbausteine für Anwendungen zur Verfügung, die sowohl mobile Geräte, als auch Ressourcen in der Umgebung des Benutzers, wie Papierdokumente, Poster und Grossbildschirme, mit einbeziehen.

Für Grossbildschirme wurden zwei Interaktionstechniken entwickelt, die auf optischer Bewegungserkennung und der Erkennung von visuellen Markern basieren. Die erste Technik erlaubt die relative Positionierung eines Cursors auf dem Grossbildschirm und ist für die direkte Manipulation von angezeigten Objekten geeignet. Die zweite Technik ermöglicht die absolute Positionierung auf dem Bildschirm und kann zur Selektierung von Bildschirm-Elementen verwendet werden. Die entwickelten Techniken wurden im Rahmen einer qualitativen Nutzerstudie untersucht und sind besonders für die Anwendung im öffentlichen Raum geeignet, wie z.B. in Bahnhöfen, Flughäfen oder Museen.

Die im Rahmen dieser Dissertation behandelten Konzepte und Techniken wurden in verschiedenen Anwendungsgebieten prototypisch untersucht. Beispiele, die in dieser Arbeit vorgestellt werden, sind: Einstiegspunkte in einen virtuellen Campus, virtuell erweiterte Brettspiele, eine interaktive Foto-Wand, ein kollaboratives Spiel für Grossbildschirme, die digitale Annotation physischer Objekte und smarte Produktverpackungen.

Chapter 1

Introduction

1.1 Motivation

Wherever we are, we have access to the physical world. Today, we also have ubiquitous access to digital information. Ubiquitous wireless networks and highly functional mobile devices allow us to be always connected. Yet, there is a missing link between our local environment and online information services. For our mobile devices, the world is always the same, if we assume uniform network coverage. For us, it is constantly changing as we move from place to place. The places and objects that surround us are often linked to specific activities, but our devices are not aware of this. By linking our computers to physical objects, we can offer many useful services. Physical artifacts become more responsive and computers get a more comprehensive picture of our world.

Linking physical and virtual worlds is a major research theme of ubiquitous and pervasive computing.[1] A virtual space like the Web is largely isolated from any particular physical environment. By contrast, an important goal of ubiquitous computing is to embed information and computation in the environment and thus situate and ground them in the physical context of the user [74]. A prerequisite to achieve this goal is to couple physical objects, places, and people with *virtual counterparts*. Virtual counterparts are representatives of physical objects in the virtual world. Conversely, *physically hyperlinked* objects become real-world representatives for their virtual counterparts. Such objects thus act as information anchors and structuring elements of the virtual space.

To enable the conscious interaction with a virtual space, the existence of a physical hyperlink has to be signaled to the user. This can be achieved by providing visible *entry points* into the virtual space. Entry points are collocated with physically hyperlinked objects. The physical environment thus becomes a distributed user interface for the virtual space, with interaction possibilities dispersed throughout physical space. This allows for a quick and effortless transition between the real and the virtual world.

The information services that are coupled to a physical object do not have to be executed within the object itself. In fact, the object might be completely passive, such as a newspaper article, a movie poster, a product package, or a door plate. If such *passive media* are incorporated into smart environments and *situated*

[1] The terms *ubiquitous computing* and *pervasive computing* are used interchangeably in this dissertation.

information spaces[2] [74], we need a computational mediator that can sense the object, visualize the virtual counterpart, and enable us to browse and manipulate it. Such a tool should help the user to bridge the gap between physical and virtual worlds by acting as a porthole into the information space.

Camera phones are in an excellent position to contribute to this vision, because they are ubiquitously available devices that are within reach of the user most of the time. They also have continuous wireless connectivity, which makes it easy to access up-to-date content from the background infrastructure and exchange information with communities of other users. Through their high portability, they naturally support mobile users in changing contexts and environments. The camera provides an additional input channel. A major aspect of this dissertation is thus the exploration of the opportunities of camera phones and similar devices as mediators for virtual counterparts of physical objects.

Our goal is to create an expressive means to "bridge the gulf between physical and virtual worlds" [211] for mobile users and allow them to spontaneously interact with encountered objects. We show how integrated cameras can act as a powerful input channel for mobile phones and turn them into interaction instruments for objects in the user's vicinity. In this way, camera phones can be used as enhanced input and control devices and physical/virtual intermediaries at the same time.

Figure 1.1: Marker-based embodied interaction with a table in a newspaper.

In the recent past, there have been considerable research efforts in linking computation to physical objects. However, these projects were mainly concerned with the physical linking technology per se or with the infrastructure required for identifier resolution. The richness of user interactions was mostly limited to a single physical gesture, such as bringing an object with an attached radio frequency identification (RFID) tag into the range of the reader to invoke the virtual counterpart. Other work on *embodied user interfaces*[3] [72] focused on improving the interaction with a handheld device itself by defining physical manipulations like tilting, but did not integrate physical objects of the user's environment. In our work, we combine physical linking and embodied interaction and allow the interaction semantics to be a function of the object and the gestural sequence.

[2] See Section 2.2.
[3] See Section 2.3.4.

The approach proposed in this dissertation uses camera phones as mobile sensors for two-dimensional visual markers. We not only retrieve the value that is encoded in the marker, but also detect the spatial orientation of the device relative to the marker in real time. We use the detected orientation for embodied interaction with the device and augment the live camera image with graphical overlays according to the orientation and aligned with elements in the background image (see Figure 1.1). The device thus embodies a "symbolic magnifying glass" that shows an augmented view of the real world. Technically, the handheld device realizes a video see-through augmented reality system.[4] This provides the basis for fine-grained interaction and enhances the currently limited input capabilities of mobile devices.

Our *marker-based interaction* approach thus turns camera phones into versatile interfaces to – and mediators for – real-world objects. We augment mobile phones with physical gestures, similar to those used in tangible[5] [103, 113] and embodied [72] user interfaces. On a larger scale, we investigate interaction via entry points that are dispersed throughout the environment. On a smaller scale, we examine the physical 3-D interaction space that is created by one or more visual markers. We define the term *marker-based interaction* as interaction with a camera-equipped mobile device within the 3-D space above one or more visual markers. The boundaries of this space are defined by the maximum distance at which the markers are detectable. The position and orientation of the device are sensed relative to the marker and are interpreted in terms of an input vocabulary of physical gestures or are directly mapped to actions on the mobile device or in the infrastructure.

Since we want to enable multiple gestures per visual marker, i.e. per physical object, a clear and consistent conceptual model is necessary. To this end, we define a state space of physical actions for visual markers. The states are discrete postures of a mobile device in 3-D space. The postures are defined via *interaction primitives*. Interaction primitives are abstractions of physical gestures and form a basic vocabulary for interaction when using mobile phones capable of reading visual codes. An essential part of the conceptual framework is that users are guided in their interactions. Each interaction primitive is associated with a visual cue that is shown on the device display and that informs the user about the interactions that are possible in the current state. The interaction primitives and the associated cues can be combined to form more expressive interactions. This enables rich input capabilities and effectively structures the output space. The guidance helps users to navigate in the state space and allows us to scale up the state space.

Future smart environments, including public places, will probably be filled with interactive displays of various sizes [153]. Today, such displays are found at train stations, airports, bus stops, or in shopping malls. However, most of these displays are limited to passive reception of the displayed information. Displays that are situated in public places are often not accessible for direct, touch-based input. Therefore, alternative approaches have to be explored. To contribute towards a solution, we investigate how camera phones can be used as personal input and control devices for large-scale displays. We employ a similar approach as for interaction with passive media, based on the paradigm of embodied user interfaces. If passers-by carry their own interaction device in the form of a camera phone, they can be

[4] See Section 2.3.5.

[5] See Section 2.3.3.

identified and authenticated by the system, data can be stored on the device for later usage, private information can be restricted for display on the personal device screen (rather than on the large-scale public display), and the vandalism problems of stationary interaction devices located in public spaces are reduced.

1.2 Contributions

The main thesis of this dissertation is that marker-based embodied interaction for camera phones enables ubiquitous information access for mobile users, allows for multiple gestural actions per physical object, and thus is a useful and versatile interaction paradigm for bridging the gap between the physical and the virtual world. To support this thesis, we describe concepts and techniques (1) to link information and services to physical objects and passive media and (2) to enable embodied and tangible interaction with this information. The main contributions can be summarized as follows.

- **The entry points concept for accessing virtual counterparts of physical objects.**

  Entry points are visible cues to the user that signal the availability of digital information in a physical environment. They are collocated with physical objects and represent the perceivable end of a physical hyperlink. Entry points enable the intentional interaction with a virtual counterpart. They provide access points to situated information spaces.

- **A visual code system for camera phones.**

  We present a system for recognizing two-dimensional visual markers. The markers we developed are called *visual codes*. The recognition system provides a number of parameters for determining the spatial orientation of the device relative to the marker, such as the target point in code coordinates, rotation, tilting, distance, and movement of the device relative to the background. It is specifically designed for the requirements of mobile phones with limited computing capabilities and low resolution cameras. Moreover, the system provides the basis for augmenting objects in the live camera image with precisely aligned graphical overlays. Based on this foundation we have developed several mechanisms and concepts for marker-based interaction, namely: (1) a framework of physical interaction primitives, (2) marker-based interface elements, called visual code widgets, (3) interaction techniques for large-scale displays, and (4) handheld augmented reality applications.

- **A conceptual framework of physical interaction primitives.**

  Based on the parameters provided by the visual code system, we survey possible marker-based interaction techniques. We propose a conceptual framework that enables the use of camera phones as embodied user interfaces. It defines a set of fundamental physical gestures that form a basic vocabulary for describing interaction when using mobile phones capable of reading visual codes. These *interaction primitives* can be combined to form more expressive interactions that provide rich input capabilities and effectively structure the output space. An interaction specification language defines rules that

associate conditions of certain phone postures to actions, such as textual, graphical, and auditory output. These actions are performed by the mobile device. The interaction primitives and their combinations have been evaluated in a usability study.

- **A set of marker-based interface elements.**

  In comparison to interaction primitives, visual code widgets operate at a higher level of abstraction. *Visual code widgets* are printable elements of physical user interfaces, comparable to the interactive elements of conventional graphical user interfaces. Each widget type addresses a particular input problem and encapsulates a specific behavior and functionality. Visual code widgets thus define building blocks for phone applications that incorporate mobile devices as well as resources in the user's environment, such as paper documents, posters, and public electronic displays. Visual code widgets form an unobtrusive distributed user interface that is steadily available in the background. They support activity-centric computing, in that interactions with visual code widgets can seamlessly be integrated in the real-world task.

- **Interaction techniques for large-scale displays.**

  For large-scale displays we have developed two interaction techniques that rely on visual movement detection and visual code recognition. The first one, called *sweep*, relies on visual movement detection of the camera phone relative to the background and is suited for direct manipulation. The second technique, called *point & shoot*, realizes absolute positioning by shortly overlaying a visual code grid over the display. The techniques are especially useful for displays that are not available for direct touch-based interaction, such as large-scale displays in public spaces. They have been designed for spontaneous interaction and low threshold of use. The techniques have been evaluated in an extensive usability study.

- **Annotations of physical objects.**

  We explore the use of camera-equipped devices with pen input as a platform for generating digital annotations to real-world objects. We contribute ways to create and interact with digital annotations using the camera and pen-based input. Two prototypically implemented annotation techniques are presented. The first one uses visual codes for digital annotations of individual items in printed photos. The second one addresses the annotation of street signs and indication panels. It is based on image matching supported by interactively established 4-point correspondences.

- **Handheld augmented reality for product packaging.**

  We propose to view product packaging as a new kind of interactive medium that provides entry points to Web-based information services. We show interaction concepts for product packages that are equipped with visual codes. These concepts use camera-equipped handheld devices as well as fixed cameras and stationary displays. Stations with fixed cameras and displays may be placed in stores and malls. In the mobile case, the virtual counterpart

of the physical product is selected and displayed by the handheld device. In the stationary setting, product packages themselves are used as interaction instruments, in the sense of tangible user interfaces.

- **Applications.**

  The concepts and techniques that were developed in the scope of this dissertation have been investigated in various application areas. Examples that are detailed in the dissertation are: entry points into a smart campus environment, augmented board games, an interactive photo wall, a collaborative game for large-scale displays, digital annotations of physical objects, and smart product packaging.

1.3 Outline

After having introduced the topic and the contributions of this dissertation in Chapter 1, the next chapter provides the necessary background knowledge. It surveys and analyzes related work in order to provide a brief overview of the state of the art. It also introduces concepts and defines terminology that have been developed in the course of this dissertation. Finally, it highlights the requirements for marker-based interaction, thereby setting the stage for the following chapters.

Chapter 3 describes our visual code system for camera phones. For each detected marker, the system provides the encoded value and a number of orientation parameters that are used to estimate the spatial orientation of the camera phone relative to the marker. The visual code system serves as the technological basis for marker-based embodied interaction.

A framework of physical interaction primitives involving camera phones capable of reading visual codes is presented in Chapter 4. The chapter describes the combination of interaction primitives, the guidance of users with the help of iconic cues, an XML-based description language for visual code image maps, and a usability study that evaluates the interaction primitives.

Chapter 5 deals with visual code widgets – physical interaction elements similar to traditional graphical user interface widgets. They consist of a physical part and a virtual overlay generated by the camera phone. In comparison to the interaction primitives described in the chapter before, visual code widgets further abstract marker-based embodied interaction in that they encapsulate a certain behavior and state.

In Chapter 6 we discuss marker-based interaction with large-scale displays. We present two interaction techniques called *sweep* and *point & shoot*, respectively. In addition to presenting and evaluating the techniques, we discuss the requirements specific to interaction with displays located in public spaces.

Chapter 7 deals with user generated content related to physical objects. It explores the use of camera-equipped devices with pen input as a platform for creating digital annotations to real-world objects. We present two prototypically implemented annotation techniques.

Chapter 8 proposes a novel view of product packaging as a medium that provides entry points to Web-based information services. We show interaction concepts for product packages that are equipped with visual codes. These concepts use camera phones as well as stationary cameras and large-scale displays. We also outline the

requirements of an infrastructure that takes into account entities along the supply chain as well as third-party service providers.

We conclude this dissertation in Chapter 9 by summarizing its main contributions and enumerating directions for future work.

Aspects of this dissertation have been published in workshops, conferences, and journals [10, 11, 12, 30, 169, 170, 171, 172, 173, 175, 176, 177, 178]. Several semester projects and diploma theses have been supervised by the author, which dealt with concepts and ideas presented in this dissertation [13, 20, 25, 31, 59, 83, 85, 115, 117, 130, 142, 228].

Chapter 2

Linking Physical and Virtual Worlds

This chapter provides background knowledge on the topic of linking physical and virtual worlds. It surveys essential related work in order to provide an overview of the state of the art. It also introduces concepts and defines terminology that have been developed in the course of this dissertation. Finally, the chapter highlights the requirements for marker-based interaction, thereby setting the stage for the following chapters.

The chapter is structured according to two major viewpoints of linking physical and virtual worlds, which can be identified by looking at the topic from different perspectives: Infrastructure issues, in the following called "linking in the large," and mobile interaction issues, in the following called "linking in the small." We start by giving an overview of relevant aspects of ubiquitous and pervasive computing.

2.1 Ubiquitous and Pervasive Computing

2.1.1 The Vision

The term *ubiquitous computing* refers to a paradigm shift from the personal computer as the single locus of computing to a more distributed, embedded, and mobile form, in which individual computational components are interconnected and cooperate with each other. The term was first used by Mark Weiser at Xerox PARC in 1988. At the beginning of the 1990s, IBM coined the term *pervasive computing* to determine a more business-oriented variation of Weiser's vision. We will use the terms *pervasive computing* and *ubiquitous computing* interchangeably.

In his seminal article [215], Mark Weiser defines *ubiquitous computing* in terms of its place in everyday life and in the natural human environment rather than in terms of technology. The goal of ubiquitous computing is to seamlessly integrate computation with the physical world, such that it is no longer noticed when used. He compares ubiquitous computing to writing, with its "constant background presence" and availability "at a glance," and argues that computing in a similar way should be "an integral, invisible part of the way people live their lives." When reading printed words in a book, we do not need to focus on the physical handling of the book, the paper, or the font style. These aspects are sensed and controlled at the periphery of our awareness and do not impinge on our limited cognitive re-

sources that are relevant for conscious information processing at the foreground of attention. In his vision, using computers should be an effortless background task that is as simple and unconscious as reading a sign. Such a seamless integration of computers into the world lets computers "disappear" from consciousness, like familiar tools. The computer is no longer central to a task, but "ready-at-hand," operated at the "periphery" of awareness. According to Weiser, "only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals" that originate from tasks and activities of everyday life.

Weiser uses the term *embodied virtuality* to refer to the process of ubiquitously distributing computers in the physical world to make information more directly accessible throughout the environment. He proposes ubiquitous computers in different sizes, each suited to a particular class of activities. Ubiquitous computers know their location and adapt their behavior accordingly. In order to cooperate, they are all interconnected. Weiser acknowledges the need for differently sized display and interaction devices for embodied virtuality, ranging from small post-it-sized *tabs* (small handheld devices), over notebook-sized *pads* (graphics tablets), to wall-sized *boards* (large-scale shared displays) [216]. Computing elements may also be worn by users, like PARC's *active badges* [212], or invisibly embedded in the environment and not directly accessible to human users. Computing may be embedded in everyday objects, such as "walls, chairs, clothing, light switches, cars" and therefore [218] "ubiquitous computing is fundamentally characterized by the connection of things in the world with computation."

In summary, ubiquitous computing is concerned with a shift in focus away from the desktop towards integrating computation, communication, and sensing into the environment. The goal is to unobtrusively support people in their daily activities and relieve them from mundane chores. The environment is not confined to a single workplace or office building, but extends into the homes of users and is even available while on the move, for example, while walking in a city or traveling by car or train. Ubiquitous computing requires a background infrastructure that allows for the processing and exchange of information between interconnected devices. It also requires access mechanisms for the user to interact with this information. Weiser suggests to distribute generic and unpersonalized tabs, pads, and boards throughout an environment in order to be ready for spontaneous use when needed. This approach works fine in private or semi-public settings, such as homes or offices, but not for users on the move. While it is possible to provide situated displays in public environments, the installation of public interaction devices is a tougher issue. As a consequence, in our work we focus on the use of camera-equipped smart phones and personal digital assistants (PDAs) as personal ubiquitous display and interaction devices.

2.1.2 Technological Enablers and Challenges

From a technological perspective, there are several recent advances that can be considered as enablers for the realization of ubiquitous and pervasive computing. Mattern [136, 137, 138, 140] identifies a number of technological trends and reasons that make the achievement of Weiser's vision plausible in the future. In the following, we shortly outline these reasons and discuss further remaining challenges.

- **Microelectronics and Moore's Law.** Moore's "Law" [148] has become a synonym for the rapid advances of microelectronics. It states that the number of components that can be integrated on a single chip doubles about every 18 months. Moore's "Law" has been valid for forty years now and is likely to continue to hold for at least a decade. The consequence of this exponential growth is that microchips become more and more powerful: their sizes decrease, their clock rates increase, parallelism and pipelining can be effectively exploited, memory chips have ever higher capacities, and the energy per unit of computation and storage falls dramatically.

  In a similar manner, the bandwidths of communication networks and the capacities of magnetic and optical storage media also increase exponentially. For mobile and pervasive computing, requirements such as minimal size and very low energy consumption are often more important that maximal processing power. Size and energy consumption are crucial aspects for the integration of microelectronic components into everyday objects.

- **Wireless communication technology.** Wireless communication technology is another essential ingredient for enabling pervasive computing. Communication technologies such as WLAN, GSM, and UMTS have been developed for medium to long-range communication of mobile devices. They have low (GSM) to medium (UMTS, WLAN) data rates and differ in communication range. The typical range for WLAN is about 100 m at data rates of 11 Mbps or 54 Mbps. The data rates of GSM/GPRS are some tens of kbps. The latency is relatively high. UMTS supports up to 1920 kbps. This is orders of magnitude below what wired communication technologies offer.

  In recent years, low power communication standards, such as Bluetooth and ZigBee, have been developed. They are designed to satisfy the requirements of local communication between collocated devices. This is a design area that was not covered satisfyingly by other wireless communication technologies, such as WLAN and mobile phone networks (GSM, UMTS). In pervasive computing environments, maximum data rates are often not the primary issue, but rather the ability to periodically transfer a few bytes over a short distance, for example, in order to communicate sensor readings or to enable smart objects to tell about their state.

- **New materials and output media.** Materials science offers new electronically active materials that could completely change the appearance of computers as we know them today. Examples include semiconducting organic polymers, electronic ink, and smart paper. Semiconducting organic polymers could allow for cheaply printing electronic circuits. Light-emitting polymers enable highly flexible displays. Smart paper encloses black and white pigments within miniature capsules. The state of such paper, i.e. the displayed contents, persists without electricity and changes with an applied voltage at the desired position. These and other materials have the potential to be so seamlessly integrated into the environment and in everyday artifacts that they are no longer discerned as computing devices.

- **Smart labels.** Smart labels or radio-frequency identification (RFID) tags enable the identification of objects over a short range (up to a few meters).

They operate without an internal power source, but instead are inductively coupled to an electromagnetic field. This external energy supply allows them to power their internal processor and send a unique identifier plus a few hundred bytes of data in a few milliseconds. There are protocols that recover from collisions that occur if multiple such labels are in the field and are simultaneously activated. Smart labels are as thin as a sheet of paper and have a surface area of below one square millimeter. Sensing the presence of a physical object is an important precondition for linking it to computational functionality. Smart labels are one way to implement this requirement. Another possibility is to use optically detectable markers, as described in detail in this dissertation.

Near Field Communication[1] (NFC) is a standard that was developed by Nokia, Philips, and Sony in 2004. It uses the same communication principle as RFID (inductive coupling), but operates only over very short distances of a few centimeters and uses relatively low data rates. The main advantage of NFC is its low power consumption. One device in active mode generates an electromagnetic field, while passive devices act in the same way as RFID transponders. Active NFC units are small and energy conserving enough to be integrated into mobile phones.

- **Sensors.** Within the fields of microsystems and nanotechnology, sensors are developed for measuring a large variety of physical and chemical phenomena. They convert the measured values to electrical signals and finally to corresponding digital representations. Advanced sensors may also contain microprocessors themselves. They react to light, acceleration, temperature, humidity, pressure, magnetic fields, and also to gases and liquids. CCD cameras, like those integrated into mobile devices, form also a powerful class of sensors. In pervasive computing, distributed sensing systems are used as the "eyes and ears" of smart objects and environments. By sensing their environment and sharing their observations, smart everyday objects can cooperate [190].

Mattern also discusses a number of challenges that remain before ubiquitous computing can be realized on a large scale [137, 139]. Examples are suitable energy sources, infrastructure questions, privacy issues, social implications, and dependability. In [213], Want and Pering describe system challenges for ubiquitous and pervasive computing. In particular, they focus on power management, wireless device and resource discovery, user interface adaptation, location systems, and context awareness. Another system-oriented view is presented by Satyanarayanan [183]. There are also many challenges regarding the deployment of ubiquitous computing technologies in real-world settings beyond simple prototypes, as discussed in [14] and [48]. Abowd and Mynatt [2] discuss new paradigms of human-computer interaction that are inspired by ubiquitous computing. They identify three interaction themes – natural interfaces, context-aware applications, and automatic capture and access – and suggest *everyday computing* as a new research thrust that investigates continuous interaction when computing is around all the time.

[1] `www.nfc-forum.org`

- **Energy supply.** Energy consumption is an important issue for smart everyday objects. Such objects are in general moveable and cannot be assumed to be tethered to a continuous power supply. Instead, they have to carry their own energy resources in the form of a battery, harvest energy through their activity – like some kinds of sensors that receive energy from the physical parameters they measure –, or be exposed to an electromagnetic field – like RFID tags. Smart objects cannot rely on the user to change their batteries. They would suddenly become very obtrusive. Partial failures caused by some objects having depleted batteries are hard to track down. An environment filled with such objects would be unusable and a very unpleasant place to live in. Unfortunately, the advances of battery technology do not follow Moore's Law. Miniature fuel cells and micro heat engines have not yet reached a level of mass production. Overall, this means that suitable energy supplies are still a major challenge for pervasive computing.

  Passive tagging technologies, such as RFID tags and visual markers, pose a partial solution to the energy problem in that they do not rely on an internal power supply. RFID tags receive their energy through an electromagnetic field that is generated by a special reader device. Visual markers can provide identification by being captured by a camera. This is only a partial solution, since with these approaches all computation has to be outsourced to a remote microprocessor and executed spatially separate from the actual augmented object. The object is thus dependent on an external infrastructure or a handheld device. Also, the integration of sensors is not easily possible. If handheld devices are used for recognizing RFID tags or visual markers, the devices can activate and execute the information and services associated to the smart object. The devices may also contain sensors that perform the sensing task on behalf of the object.

- **Location and context.** In order to support users according to their current situation, objects within ubiquitous computing environments need to know their location and context. A large spectrum of services becomes possible, if the location of objects is continuously tracked. There are a multitude of location systems for ubiquitous computing [97] that measure signal strengths and signal runtimes in order to compute locations. Outdoor objects may be located using GPS [82], mobile phones can be located via GSM, objects sensed by stationary RFID readers are located within the range of the reader. GPS provides an accuracy of a few meters, but requires a unobstructed line of sight to at least three satellites. GSM cell-based localization achieves an accuracy of 300-550 m. UMTS can achieve an accuracy of 30-50 m. Using multiple overlapping networks, WLAN reaches an accuracy of 20-40 m. Other location systems use runtimes of ultrasound signals and even the signal strength of infrared transmitters and some have a much higher accuracy (for example, a few centimeters for ultrasound location systems [94, 214]).

  Context systems try to discover the situation of the user within a smart environment and adapt their behavior accordingly. A system that knows about context, in addition to explicit user input, has the potential to automate user actions and operate proactively according to the needs of the user. It exhibits "smartness" in guiding or adapting to the user. While this sounds

promising, it can be dangerous, because the actions of the system are not only dependent on the current explicit actions of the user, but also on what the system thinks the user's situation might be. When considering the limited and low-level sensing possibilities available today and the fundamental difficulty of the problem of drawing valid conclusions in an unconstrained everyday setting, it becomes apparent that there is much room for misinterpretation and subsequent user confusion. Making system actions transparent to the user and explaining them within a simple and coherent conceptual model is therefore a major research strand. Although a number of prototypes have been developed [56, 57, 182, 185], context aware systems remain an active area of research and are therefore an enabling technology as well as a challenge. Current research efforts focus on modeling context, for example by using ontologies [45, 88, 209], in order to exchange context between independent entities. A related focus is on context reasoning [199] to allow for higher-order inferences on low-level context data.

- **Privacy and trust.** In a world of ubiquitous computing, we will be constantly surrounded by invisibly embedded computers. Our casual moves, and not only explicit input actions, will be continuously sensed. How can we retain our privacy if smart objects and environments sense our presence, remember everything they have seen, and communicate their data to remote locations? How can people develop a sense of trust, if they are constantly watched and guided by invisible computing systems? Questions such as these are pressing in ubiquitous computing environments. Langheinrich [128] provides an in-depth discussion of privacy issues in ubiquitous computing and proposes a number of privacy protection mechanisms. Weiser et al. [219] characterize the privacy problem as follows: "The problem, while often couched in terms of privacy, is really one of control. If the computational system is invisible as well as extensive, it becomes hard to know what is controlling what, what is connected to what, where information is flowing, how it is being used, what is broken (vs what is working correctly, but not helpfully), and what are the consequences of any given action (including simply walking into a room). Maintaining simplicity and control simultaneously is one of the major open questions facing ubiquitous computing research." Analyses of social, economic, and ethical implications of ubiquitous computing can be found in the following publications [26, 27, 28, 29, 129].

- **Dependability.** Ubiquitous computing needs robust and reliable infrastructures. Computing will affect a much larger part of our life than is already the case with today's computers. If a large number of autonomous computing entities collaborate to provide some service, there is a huge potential for failures. Failing computers that are embedded in our everyday life mean that our everyday tasks and activities are impaired, which would be extremely annoying. New approaches are required to make pervasive computing systems resilient to failure and fault tolerant. In addition, attacks on pervasive systems can have severe and far reaching consequences.

- **Interaction design.** Interaction in ubiquitous computing is different from traditional interaction with desktop computers in many ways. If computers

are invisibly embedded, there need to be ways to signify interaction possibilities to users. Since ubiquitous computers will be available all the time, continuous interaction, task interruption, and resumption will be important issues. Interaction will take place in a larger range of modalities and levels of consciousness. Explicit interaction will be complemented by implicit interaction [184], in which everyday behavior of users is sensed and interpreted with respect to operations in the virtual world. Mobile users will have to be supported in special ways in order to be able to use computing services at their respective locations. New interaction instruments are required that link virtual resources to physical objects and allow users to interact with virtual resources. Since the user's attention is limited, a great deal of effort has to be spent on designing interfaces in such a way, that users are unobtrusively supported, but not overwhelmed by the large number of computing devices. The topic of interaction in ubiquitous computing will be discussed in more detail in the next section, as it is especially relevant in the context of this dissertation.

2.1.3 Interaction in Ubiquitous Computing

Whereas human interaction with computers has traditionally focused on explicit interaction at the focus of the user's attention, Weiser introduces the notion of *calm technology* [218]. In contrast to desktop computers, which have been designed for excitement of interaction, ubiquitous computers, which are all around us throughout the day, must not constantly demand our attention. Weiser sees calmness of the user interface as a central quality of ubiquitous computing systems [218]:

> The most potentially interesting, challenging, and profound change implied by the ubiquitous computing era is a focus on calm. If computers are everywhere they better stay out of the way, and that means designing them so that the people being shared by the computers remain serene and in control. Calmness is a new challenge that UC brings to computing. [...] But when computers are all around, so that we want to compute while doing something else and have more time to be more fully human, we must radically rethink the goals, context and technology of the computer and all the other technology crowding into our lives. Calmness is a fundamental challenge for all technological design of the next fifty years.[2]

Calmness does not preclude explicit user interaction with a system. Rather, in addition to the center of perception it also engages the periphery of perception and allows us to move objects of interest back and forth between the two. Extending the peripheral reach means that more information can be attuned to, than if conscious foreground processing would be the only form of engagement. Moving something to the center of attention means to take control of it. A large part of interaction design in ubiquitous computing thus means to "design for the periphery." It is therefore a prime interaction design challenge for ubiquitous computing to achieve acceptable levels of feedback and control, without overwhelming or unnecessarily distracting users.

[2] See `www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm`

In [202], Tolmie et al. explore the concept of *invisibility in use* in ubiquitous computing that was proposed by Weiser.[3] In a field study of domestic life they discovered the importance of routines. A *routine* is an automated procedural plan of action that relieves people from constantly having to invent sequences of action and consciously account for what they are doing. Routines are used as resources for acting in a given situation. Within a domestic routine, an action, such as knocking at a door, has a specific meaning at a specific time, that cannot be inferred from the action itself. The action is not meaningful in itself, but only as a sign or message in the context of a routine.

An object might be perceptively visible, yet practically *invisible in use* if it is part of a routine. An example is an alarm going off at the usual time, for example, every morning at 7 o'clock. The audible alarm is noted, but not remarkable, since it does not break the routine, which would be the case if the user had to explain to himself why the alarm goes off.

Routines are thus "invisible in use." They are present as background resources to help with everyday activities, but they do not demand attention or explanations during their execution. Routines are in line with Weiser's vision of seamless and invisible embedding of computation in the environment, "so fitting, so natural, that we use it without even thinking about it."[3]

Tolmie et al. point out that the concept of "invisibility in use" is complementary to perceptual invisibility, which is achieved by technological miniaturization and the invisible embedding of computing nodes into the environment. The concept is also different from the engagement of peripheral awareness, as is the goal of ambient displays [135], for example. Ambient displays concern the perceptual psychology of peripheral sensory processing.

Following this line of reasoning, the objective of the design of ubiquitous computing interfaces would be to "unremarkably" embed computation into routines in order to invisibly amplify the effectiveness of everyday actions. Or as Weiser put it [216]: "Whereas the intimate computer does your bidding, the ubiquitous computer leaves you feeling as though you did it yourself."

Consequently, it does not suffice to computationally augment physical objects per se. Rather, the routines within which artifacts are used have to be augmented. A routine in which an artifact is used can be very specific and locally restricted and so the kind of computational augmentation can be equally specific. The augmentation not only has to take into account what actions are done, but also what the significance and accomplishment of these actions are. Depending on an object's use within a routine, its augmentation can be "natural" and "intuitive" or not. Intuitive semantics of the augmentation is not only depending on the object per se, but also on the context of usage.

Heer [96] notes that even though physical invisibility might be aesthetically appealing, "total invisibility, and the lack of feedback and control that implies, is obviously undesirable." Heer distinguishes *invisibility in use* and *infrastructural invisibility*. In line with Tolmie, invisibility in use means that a computational tool fades into the conceptual background: We work *through* a tool, rather than *with* it. Infrastructural invisibility means that computation is embedded in the environment in such a way that it is tacit (unconscious and not readily verbally expressible) in thought and action. Invisibility in use arises from a potentially

[3]See `www.ubiq.com/hypertext/weiser/UbiHome.html`

long learning process and continued practice. A prime example is literacy. A vast institutionalized education system and years of schooling are required to achieve a level of effortless use.

If computing is deeply embedded in the environment and largely invisible, allowing the user to discover the available functionality and interaction possibilities becomes a major issue. The basic question is, how an invisible user interface actually presents itself to the user. In order to examine this issue, the distinction between explicit interaction and implicit interaction as proposed by Schmidt [184] is useful. *Explicit interaction* employs special actions that have been specifically designed for the purpose of human-computer interaction: Users are consciously interacting with a computing device at a certain level of abstraction, such as the command-line, direct manipulation using a graphical user interface (GUI), gesture, or speech input. In contrast, *implicit interaction* employs actions that users would perform anyway in their interactions with the physical environment, even if no computing capabilities were embedded. Hence, users' actions are not directed primarily towards a computing device, but are rather natural actions that arise from everyday engagement in the physical environment. During implicit interaction, users' actions are sensed and interpreted by a context aware system in the background. Implicit interfaces are thus based on two main concepts: perception and interpretation.

An example of implicit interaction, given in [184], is a garbage bin that senses product packages thrown in, in order to build up a shopping list. The action performed by the user is in no way adapted to the computational functionality. The system has a perception of the user's action (by scanning a bar code or an RFID tag) and a predefined interpretation of it (things that go into the garbage bin have to be noted on the shopping list). The user consciously interacts with the product package and the garbage bin and only implicitly with the computer. Simpler examples are automatic light controls; a door that automatically opens as the user approaches it; or a group of users gathering in front of a large shared display that thereupon automatically shows the state of the workspace when they last met.

Brodersen and Kristensen [38] note that implicit interaction has to be used with care, because it removes control from users. Implicit interaction does not expose the mappings between everyday user actions and associated system reactions. The relationships between the different computing entities is hidden from the user. Implicit interaction thus complicates the interpretation of the situation in a pervasive computing environment. An interaction with a physical object might be sensed by the system and have an implicit side effect that may be observed by the user only later in time (like the appearance of objects on the shopping list in the example above). Explicit interaction, by contrast, puts the user in control of the situation, but may be annoying to perform in an everyday setting. An important aspect in finding the right balance between implicit and explicit interaction seems to be, how the system's perceptions and interpretations are conveyed to the user. The realizable degree of implicit interaction depends on the "naturalness" of the mappings, the cost of misinterpretations by the system (a superfluous item on the shopping list should not be a severe issue in most cases), and the kind and timeliness of feedback given to the user. In this context, Norman's four suggestions for user-centered design ([152], p.188; quoted by [38]) are relevant:

- Make it easy to determine what actions are possible at any moment.
- Make things visible, including the conceptual model of the system, the alternative actions, and the results of actions.
- Make it easy to evaluate the current state of the system.
- Follow natural mappings between intentions and the required actions; between actions and the resulting effect; and between the information that is visible and the interpretation of the system state.

In other words, make sure that (1) the user can figure out what to do, and (2) the user can tell what is going on.

Brodersen and Kristensen [38] also emphasize the importance of the way a ubiquitous user interface conveys its functionality to the user. If users enter an unknown environment, they have to be able to discover the available functionality and how it may be utilized. In their *interaction through negotiation* paradigm, negotiation refers "to the mediating process relating the human users and the technological possibilities in a given situation." For a nomadic user, for example, the situation continuously changes and thus there is a need for constant negotiation with the currently available resources. To analyze interaction through negotiation, three aspects are identified:

- *Availability.* What are the current options?
- *Interpretability.* Are these options currently usable and if so how?
- *Connectivity.* What are the hidden connections (functional relationships) between the available resources and what do they mean?

Availability refers to the way a smart environment presents itself in an interaction situation and how interaction possibilities are perceived by users. Interpretability refers to the conceptual understandability of the interaction possibilities in a given situation. Finally, connectivity refers to the need to present a coherent interface for a collection of computing entities (Brodersen and Kristensen use the term "web of technology") to the user and to deconstruct that collection if needed. This allows the user to develop a conceptual model of the dependencies of the interconnected entities.

Sensing-based interaction employs physical gestures that users can perform with their mobile device, such as picking it up, holding it in a certain way, looking at it, walking around with it, or aiming a camera's viewfinder at a certain spot. These gestures may occur naturally and without special attention when handling a device, such as looking at the display and orienting the display in order to avoid glare. Such gestures can also be designed as a functional part of the user interface. An example is using tilting of a device for scrolling the display contents [16, 163]. For mobile interaction tasks in the real-world, such as using a PDA while walking along a busy street, interaction techniques that do not require the user's visual attention, but rely on the sensing of natural physical gestures can be very useful.

Hinckley et al. [98] discuss sensing-based handheld interaction with respect to *foreground* and *background* and transitions between the two. These concepts are defined in terms of the degree of user attention that the interaction requires rather than in terms of the location of the sensors and actuators (on the mobile device vs. in the environment). Foreground interaction requires the user's direct attention and relies on gestures that are especially designed for the interaction. Background interaction is enabled through *background sensing* that is defined as sensing natural gestures that users have to perform anyway in handling their device. As an example, a user may change the orientation of the device from portrait to landscape. As a response, the system may automatically change the orientation of the displayed contents.

The preceding paragraphs show that interaction within pervasive computing environments involves a large spectrum of issues, ranging from calm and invisible computing over implicit interaction to explicit interaction and interaction through negotiation. Other than PC-based interfaces, ubiquitous environments will be around us on a "24-by-7" basis (24 hours a day, 7 days a week) and thus need to employ a much larger range of interaction modes, than just input with a keyboard and a mouse and output on a single screen. Ubiquitous computing also engages the periphery of our attention, uses everyday objects as input devices, and links virtual semantics to physical objects and everyday routines.

2.2 Linking in the Large: Situated Information Spaces

2.2.1 Introduction and Example Infrastructures

The term *situated information space* was introduced by Fitzmaurice in [74]. It denotes a 3-D everyday environment in which information is associated with physical objects and locations. The information is embedded in the physical context in which it originated or in which it is most significant. For many kinds of everyday information, the user benefits from grounding it in a real-world context, rather than presenting it out of context at a stationary PC display. Embedding information helps the user to understand the organization of the information space and enables ubiquitous access to it. To Fitzmaurice [74], "the key idea is that the physical object anchors the information, provides a logical means of partitioning and organizing the associated information space, and serves as a retrieval cue for users." As an example object he describes a combined telephone and fax machine with customizable information "hot spots" on the physical device: incoming calls are located near the earpiece, outgoing calls are located near the mouthpiece, and the virtual phone book is placed next to the keypad.

A situated information infrastructure needs a way to visualize and interact with the embedded information. Fitzmaurice proposes a model in which a spatially aware handheld device acts as a porthole to the situated information. In this model, the device's location in 3-D space determines the content that is shown on the device's screen. Since we as human beings experience 3-D space and interact it with every day, it is beneficial if handheld devices also possess this ability – ideally at a spatial resolution and accuracy that matches human performance. Fitzmaurice

contrasts the use of a personal handheld display with the original vision of ubiquitous computing, namely to embed a multitude of interactive displays of different sizes into the environment [215]. These displays emit information either as a result of explicit user interaction or upon detecting the presence of a user, for example, by sensing active badges [212]. With environmental displays, users have to scan the environment for the display that shows the information they are looking for. Personal handheld displays can provide information in a selective push-oriented style, if they know what information is relevant for a certain user at a certain time or in a certain location. Personalization is more difficult with embedded displays in public environments, since the displays are shared by multiple people. Personal displays also provide a means to filter the flood of information that is generated by augmented physical objects and devices according to the preferences of the user.

We now outline two influential projects that aim to provide an infrastructure for linking physical and virtual worlds.

Sentient Computing

The *Sentient Computing* project [4] is concerned with making computing systems sensitive to the locations of objects and people within office buildings. By treating the concept of physical space as an integral part of computer applications and providing suitable programming abstractions, applications can observe and react to spatial events in the real world. The project is based on a high-resolution indoor location system [94, 214], called *ActiveBat*, that matches human performance in the perception of space. ActiveBat is an ultrasonic location system that achieves fine-grained 3-D positioning in real time, with 95% of the sensor readings being correct to within 3 cm in 3-D. In this system, a small ultrasonic transmitter and wireless transceiver, called *Bat*, is attached to persons and objects. A signal over the wireless link causes it to emit an ultrasonic pulse. Ultrasonic receivers mounted in the ceiling measure the time of arrival of the pulse and send their measurements over a wired network. A central controller combines the measurements to compute the bat's position. The Sentient Computing system maintains a model of objects in physical space and provides a programming model which allows developers to specify spaces to monitor and actions to execute, based on spatial relationships such as containment.

"Smart posters" are an example of physical objects that are augmented by computational functionality. Smart posters contain sensitive areas that act like buttons of traditional GUI applications. When the user clicks the button of a bat close to such an area, a specific action is triggered. In that way, posters that are stuck on the wall act as user interfaces that are available throughout the user's environment. An application example is using a poster to control a scanner with the bat. The bat can be placed at sensitive areas on the poster in order to start the scanning process, to specify where the scanning result should be emailed to, and even to adjust continuous parameters such as the contrast. A poster provides enough space to show detailed instructions and catches the eye to indicate the availability of the service. Other example applications are computer desktops that follow their users to the nearest workstation or phone calls that are routed to the nearest handset.

Cooltown

Whereas the Sentient Computing project deals with coupling fine-grained locations within (office) buildings to computational services, the "Cooltown" project[4] at HP Labs focuses on physical-virtual links in mobile and nomadic contexts. *Nomadic* users are defined as users who are "moving around to work, to shop, or to play" [120]. The places nomadic users enter, such as workplaces and shops, and the objects they deal with on an everyday basis, like toys and cars, are increasingly equipped with computers. Cooltown builds on the convergence of Web technology, wireless networks, and handheld client devices, in order to link nomadic users, physical entities, and information services.

Cooltown is Web-centric in that it provides an infrastructure that enables a *Web presence* for physical entities, which are categorized as *people*, *places*, and *things* [54]. The term *Web presence* denotes a home page of a physical entity and includes the automatic correlation of this home page with the physical entity itself. Example things that have been augmented within Cooltown are printers and artwork. Example places are museums and bus stops. The Web, as an already ubiquitous infrastructure, combined with wireless communication networks and handheld client devices, allows for decentralized but standardized access to the Web presence of such things. Arguments for using the Web as the base infrastructure for Cooltown are that it is widely deployed, it rests on open and robust standards, it allows for the integration of new standards, it does not require significant resources from client devices connecting to it, it supports human-to-computer and computer-to-computer interactions, and the Web page is a very versatile user interface. In contrast to other projects, Cooltown keeps the user in the control loop and does not try to automatize system functionality like service discovery [25] and component association. This means that the system does not have to infer user tasks or intentions, which avoids a whole class of error possibilities.

In Cooltown, *places* are defined as contexts for service provision, in which wireless networks and augmented physical things are available. Places have physical boundaries and semantics assigned to their use [44]. Various policies determine, which of the services that are linked to a place are made available to a user at a given point in time. The policy comprises the function of the place – which may change over time if the place is used for multiple purposes – user access rights, user preferences, and the capabilities of the client device. The contextual organization and Web representation of a place is realized with a specific infrastructure component, called a *place manager* [44]. The place manager maintains a directory of things in the place, which reflects the hierarchical containment relationship between things and places or places and other places.

The correlations between physical entities and their Web presences is achieved using different technologies in Cooltown. There is *direct sensing*, where a computationally enhanced physical entity provides a URL that can be sensed by the user's mobile device. This is typically implemented using tiny infrared beacons, that are placed next to the object to augment, for example, a piece of artwork. With *indirect sensing*, the physical entity just provides a unique identifier (it might only be unique within the place in which it is located), which has to be resolved by a *tag resolver* in an intermediary step. The tag resolver maps the identifier to

[4] `www.cooltown.com/cooltown`

a URL that can then be used to access a service. There are different technical options for providing the identifier, such as 1-D or 2-D barcodes, or RFID tags. In spite of a numerical identifier, tag resolvers might also use GPS coordinates or a photograph of the object of interest as parameters that it resolves to specific services. By using *template managers*, Cooltown provides mechanisms for creating new Web presences and linking new physical entities to them. A more involved form of interaction than just sensing URLs and identifiers from the environment is *e-squirt*. *E-squirt* is a real-world drag-and-drop technique that allows to send a previously captured URL to a nearby device via a short-range wireless link such as infrared or Bluetooth. E-squirting at a projector will project the corresponding Web page; e-squirting at a printer will print it.

Situated Computing

Hirakawa et al. [101] define *situated interaction* as "the integration of human-computer interaction and the user's situation in a particular working context in a mobile environment." This definition acknowledges that mobile interaction is not only a function of the device, but also of the user's activities and context. Hirakawa et al.'s *situated computing* paradigm takes into account the user's situation, which consists of both physical context and real-world activity. During their activities, field workers need to access and create information at different and changing work sites. Hirakawa et al. argue that computation and services should consequently be linked to the physical location and type of work, rather than a rigid directory structure within a file system.

Situated information spaces will not only rely on explicit user input, but in addition on implicit interaction and background sensing to a large extend. Bellotti et al. [21] provide a systematic framework for the design of sensing-based interactive systems. Sensing systems expand user input beyond key-pressing and point-and-click towards speech, gestures, tangible objects, and context awareness. Bellotti et al. propose five questions for sensing systems that need to be answered when such systems are designed (and used). The questions address design challenges specific to sensing-based interactive systems. They provide guidance and help designers to avoid potential pitfalls when creating such systems. The five questions are:

- **Address.** How to address (or avoid to address) one (or more) of the available devices or system parts?
- **Attention.** How to know if the system is "listening" and ready for interaction?
- **Action.** How to identify and select a possible action, a target object, and parameters of the action?
- **Alignment.** How to evaluate if the system is doing or has done the right thing?
- **Accident.** How to avoid mistakes? How to undo unwanted actions?

These questions are different from Norman's *seven stages of action* [152] (p. 45ff): forming the goal, forming the intention to act, specifying an action sequence, executing the actions, perceiving the system state, interpreting the state, evaluating

the state with respect to the goal. The above questions focus on communicative rather than cognitive aspects of interaction. For GUI systems, most of the above questions are not the main issue, since there are predefined solutions for addressing a system, getting its attention, selecting actions, and obtaining feedback. With keyboards and mice, all user input has a clear focus. Furthermore, in GUI systems the human user drives the interaction and the computer rarely takes the initiative. With sensing-based systems, the communicative act itself becomes an issue of interest – as it is in social science when considering human-human interaction.

In situated information spaces, there need to be mechanisms or conventions that convey to users how they can *address* the system. Examples are speech input, physical gestures, or handheld devices that sense objects in the environment. Users also need to be able to distinguish between conventional un-augmented (inattentive) objects and augmented (*attentive*) objects in the environment. If the system parts can be identified, the possible *actions* have to be discovered by the user. This could be realized by learning speech commands or physical gestures, by having a few fixed and well-known operations per augmented object, or by using display devices that indicate the possible operations. The set of possible actions might also be adapted to the user's current situation, preferences, or abilities, if these are known to the system. Furthermore, the range of functionality might be designed such that the physical characteristics of an object signify its virtual affordances. The physical form might facilitate certain ways of holding the object. Its appearance and conventional real-world use might suggest an intuitive natural mapping to a certain computational functionality. Tangible user interfaces [103, 113] try to use physical features of objects in these ways. We describe tangible user interfaces in more detail in Section 2.3.3. By themselves, everyday augmented objects do not provide an easy way of *feedback* of their virtual state that allows the user to evaluate what the system is doing. Timely and appropriate feedback requires output devices. Possible options are handheld devices, displays situated in the environment, or audio output. Very low bandwidth output might be used to signal the successful completion of an operation, such as a small set of distinctive sounds, lights, or tactile feedback. Finally, dealing with misinterpretations and *errors* is an inevitable aspect of interaction with computers as well as communication between humans. A situated environment has to provide mechanisms to inform users about mistakes, allow users to retract to former system states, and to cancel system actions in progress.

The Design Space of Situated Information Spaces

In our work, we are trying to specifically support the mobile or nomadic user. The places that nomadic users visit may be unfamiliar to them, they may be public or semi-public. Places are typically linked with specific activities that are performed within them. The table below gives examples for places and objects that nomadic users may encounter and the activities they might perform within that context. A single space may be used for multiple activities. User activities depend on the role of the user (consumer vs. shop manager), the time of day (breakfast vs. afternoon), the kind of activity (professional vs. leisure), the social situation (individual vs. group), the social status of the person, and many other factors.

The examples given in Table 2.1 show that there is a wide variety of places with very different technical capabilities. While we can assume that factories and

user	place	object	activity
field technician	factory	machine	performing work steps
teenager	tram station	sign	scanning arrival times
woman	home	newspaper	reading travel ads
man	city	cinema poster	memorizing movie info.
consumer	shop	product package	studying nutritional info.
kid	home, breakfast	package	playing game
group of kids	home, afternoon	board game	playing
businessman	airport	large display	copying information
students	underpass	large display	uploading and rating photos
student	classroom	exercise sheet	working on assignment
woman	city	vending machine	buying refreshments
traveler	train	flyer	checking arrival times
architect	on site	construction plan	making annotations
nurse	patient's bedside	ID bracelet	recording temperature

Table 2.1: Examples of places, physical objects located within them, and typical user activities.

hospitals are equipped with the newest technology, this might to a lesser degree be the case for tram stations, shops, and construction sites. However, we can assume that wireless communications network coverage will be available everywhere, even though with varying fidelity in terms of latency and bandwidth. Moreover, most of the objects that are dealt with in the above examples are non-electronic. Nonetheless, there is tremendous value coupling the right kind of information or service to them. Some examples are only possible because of the presence of information technology. Therefore, the introduction of information technology in the given situations empowers users in that it opens up new opportunities of action that were not possible without it.

Our aim is to support users in any of these situations, given that they carry a mobile device that is capable of sensing objects in the immediate surroundings and that can use these objects as endpoints of user interfaces that are distributed in the real world. A major point is how the issues that Brodersen [38] and Bellotti [21] identify are addressed. How are users made aware of interaction possibilities in unfamiliar everyday environments and, once they are aware of them, how do they use them?

2.2.2 Entry Points Concept

Our *entry points concept* tries to give a possible answer to these questions. Our concept of linking physical entities to virtual counterparts (see Figure 2.1) consists of

- an entry point,
- a physical hyperlink, and
- a virtual counterpart.

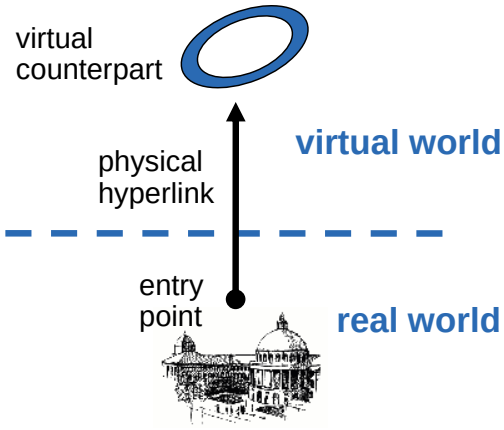


Figure 2.1: A visible entry point to a virtual counterpart.

The *entry point* is collocated to or integrated with the physical object to which it belongs. It serves a dual purpose: it provides a cue to the user and it can be detected by a sensor. Entry points are thus significant for the user as well as the system. For the user, they provide a visual cue that signals interaction possibilities with the virtual space. A collection of entry points within an environment forms a set of information anchors that partition the information space according to the corresponding physical entities. The second function of an entry point is the detectability by some sensor. The term entry point is primarily a conceptual notion. It does not need to physically exist. If a physical object is machine-recognizable without any instrumentation and as long as it is perceived by users as providing a digital service, such an object might serve as its own entry point. There are various technical options for sensing the entry point, such as reading barcodes, scanning RFID tags, sensing infrared or Bluetooth beacons, or determining the current location.

The *physical hyperlink* is the technical realization of the link between the entry point and the virtual counterpart. It might be implemented using a handheld device that can sense the entry point and fetch the virtual counterpart over its wireless link. The concept of physical hyperlink includes issues such as naming and resolving sensed identifiers, taking context information and different policies into account, and locating the virtual counterpart.

The *virtual counterpart* is a representation in the virtual space of the physical entity to which the entry point is attached. In the simplest case, it is a textual description or a Web page that is presented to the user. The counterpart may also trigger actions in the background infrastructure or on the mobile device. It might record its relation to other entities, such as the current place or state of the physical object it represents. We will discuss the various options in more detail below.

Kindberg [119] discusses physical hyperlinks and identifier resolution in the context of a Web-based infrastructure. In this paradigm, *tags* are physically bound artifacts. Each tag carries an identifier that the system resolves to a globally unique identifier and finally to Web resources. Kindberg identifies the following subtasks in an identifier resolution system:

- *Identifier creation.* Identifiers (IDs) for resources are created.

- *Binding.* Identifiers are associated with resources. *Physical binding* involves attaching a tag to an artifact. *Virtual binding* involves creating a table entry in order to map an identifier onto a virtual resource.

- *Identifier capture.* Identifiers are captured, i.e. sensed, from a physical entity.

- *Conversion.* Raw identifiers are sometimes converted to globally unique identifiers (GUIDs).

- *Resolution.* Resources are looked up given a particular set of bindings, an identifier, and optionally contextual parameters.

It has to be noted that the proposed system is inherently based on identifiers. Another approach could be to use some intrinsic property of the physical entity for associating resources. An example would be to use the color and texture of the surface of an object as a parameter to retrieve relevant resources with similar color and texture. The object would then act as an activator for a kind of associative memory. In this case it is obvious that the retrieved resources are pretty arbitrary.

But even in a resolution system in which an identifier acts as a key, the resources to retrieve are not uniquely defined. The resolution may depend on contextual parameters, such as the user's preferences, the current location, and the current time. More importantly, the resolution depends on the resolver. The resolver can introduce a particular semantics or even a particular world view. Mattern [137] discusses some of the problems with ideologically biased resolvers. On the other hand, if no "authoritative" binding exists, users may benefit from multiple resolvers that provide information according to their needs. In the Web-based resolution system of [119] users can choose a resolver by browsing to a Web page that contains machine-readable directives that set the URL of the resolver to use. Thus, when looking up product information, users who are concerned about genetically modified food would choose another resolver than users who suffer from diabetes. This approach is very extensible and scalable and lets anybody with control over a Web page set up a resolver.

2.2.3 Modeling Virtual Counterparts

A central aspect of an infrastructure for situated information spaces is the way virtual counterparts are modeled. Important questions are what classes of virtual counterparts exist, how virtual counterparts are structured, what functionality they offer, if they provide an application programming interface, how they can cooperate with other virtual counterparts, and so on. The data model has to include a description of the virtual counterpart's attributes, the services it offers, its access policies, and much more. For the cooperation of virtual counterparts that originate from different sources, an open data model is required. If, for example, the frozen food pizza wants to tell the oven about its baking temperature and duration, the counterpart of the pizza and the counterpart of the oven need a common way to do this. The open character of a "Web of things," in which objects come from different sources, precludes a proprietary specification of the virtual representation of an artifact. Instead, a flexible, consensual model is necessary that allows for the exchange of data between independent objects.

In order to reach a shared understanding despite of this heterogeneity, the use of *ontologies* seems beneficial. An ontology [67] is a shared conceptualization of a domain of discourse. It consists of a formal, explicit description of relevant concepts, including their attributes and relationships.

One approach to implement ontologies is the *Semantic Web*.[5] Its goal is to equip resources on the Web with a well-defined meaning, which is suited for automatic processing. The Semantic Web is based on the Resource Description Framework[6] (RDF) and the Web Ontology Language[7] (OWL). While RDF allows to make statements about Web resources, OWL can model relationships between resources. From these representations, rule-based tools can infer implicit information. Modeling problems of the Semantic Web are very similar to the problem of modeling virtual counterparts. In fact, the W3C document that specifies the requirements of OWL, explicitly mentions ubiquitous computing as a use case.[8]

Even if these foundations are established, modeling virtual counterparts using ontologies and rule-based programming is a major research problem, which is not the focus of this dissertation. The first task of such an approach could be to develop a suitable *upper-level ontology* as a framework for basic concepts that are of general relevance within ubiquitous computing. Example concepts are place, person, activity, device, and artifact. Such an upper-level ontology would then be extended by domain-specific ontologies. As a case in point, Cooltown's categorization of people, places, and things can be seen as a very rudimentary definition of upper-level concepts. However, the semantics of these concepts and their attributes are not formally defined. Increasingly, context aware systems use ontologies to represent context [45, 88, 209].

A virtual counterpart is supposed to mirror the state of the physical object it represents. The virtual counterpart thus changes with sensor data it receives from its artifact and with user interactions. These events make the virtual counterpart a very dynamic entity. The virtual counterpart typically includes an artifact memory that records its interaction history over time.

2.2.4 Smart Campus Environment

We used the entry points concept within an explorative project,[9] whose subject was to couple a virtual campus to the physical campus environment at ETH Zurich. The project was part of a larger initiative at ETH Zurich, named "ETH World."[10] ETH World plans to provide a virtual environment that sustains the university community, supports research, teaching and learning, unifies various university services, and is used by everyone working or studying at ETH Zurich.

As Weiser noted [217], the university campus is an interesting application environment for applying ubiquitous computing concepts. In a campus environment, a substantial number of users share a large amount of their information needs. These needs include information about schedules, locations of class rooms, lectures, assignments, lab equipment, presentations, seminars, sports events, student

[5] www.w3.org/2001/sw
[6] www.w3.org/RDF
[7] www.w3.org/2004/OWL
[8] www.w3.org/TR/webont-req/#usecase-ubiquitous
[9] www.ethworld.ethz.ch/projects/details_EN?project_id=137
[10] www.ethworld.ethz.ch

ads, or even the current mensa menu. Much of this information is directly related to objects, places, and people that are situated within the campus environment. The ubiquitous installation of wireless communications facilities such as WLAN, together with small handheld devices and technologies for detecting objects or locations, make it possible to satisfy the information needs of users in a campus environment in new ways.

While the World Wide Web with its manifold services in the area of teaching and research exists mainly isolated in the virtual world, a *ubicomp campus infrastructure* is supposed to augment the physical campus infrastructure and to be closely related to the physical entities, places, and people within it. A ubicomp campus as we envision it can be seen as an instance of a situated information space. For the usefulness and usability of a virtual campus it is vital to make it accessible starting from the physical objects and the physical environment of the people involved. In our approach, we provide ubiquitous access to the virtual campus by embedding visible *entry points* in the environment. The visibility of the entry points ensures that the virtual campus is reinforced in the consciousness of its users and used in an everyday manner.

The "ETHOC" system[11] that we developed as part of the project, provides the necessary infrastructure to enable users to attach online information and functionality to printed documents. The system performs the creation, administration, and intermediation of online resources related to paper documents. To information providers, it offers a Web-based author portal for generating unique identifiers that can be printed as barcodes and for associating online content and actions to printed material. To users it offers simple means to interact with virtual counterparts of printed documents using a variety of devices, such as WAP-enabled mobile phones or PDAs. Moreover, it stores a personal access history for each user.

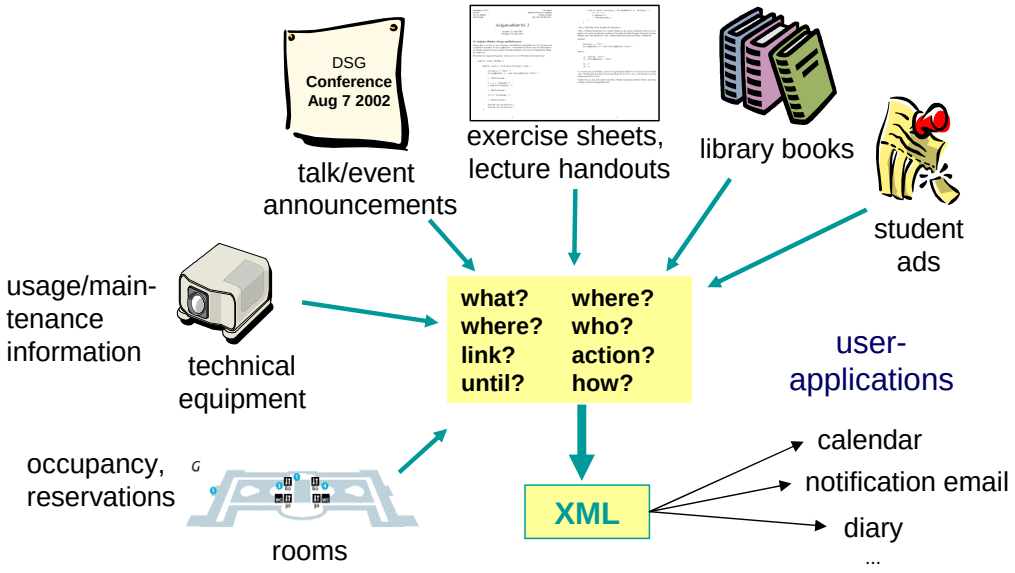


Figure 2.2: Examples of objects within a smart campus that might be linked to online information.

Potentially relevant information related to physical objects might concern place and time as well as object-specific information or online information on the Web [30].

[11] "ETHOC" stands for "Every Thing Has Online Content."

Examples for such couplings in the context of a university campus are (see Figure 2.2):

- lecture halls that are associated with their occupancy states and reservation plans;
- technical equipment that is connected to its maintenance schedule or user manual;
- flyers, posters, and announcements that are linked to background information, electronic calendar entries or ticket reservation systems; and
- lecture handouts and exercise sheets that are coupled to related multimedia content or exemplary solutions.

Not only passive information, but also online functionality and actions can be triggered when following a link from the physical object to its virtual counterpart. This includes, for example, performing reservations, storing calendar entries on the user's mobile device, triggering email messages, or signing up for event notifications.

To effectively support the everyday activities in research and teaching, the members of the university community have to be given the opportunity to actively participate. ETHOC provides an easy-to-use system that enables students, assistants, faculty, and staff

- to augment printed material they create as well as physical resources they use for teaching and research by online content and functionality, and
- to interact with the virtual resources using a variety of mobile and stationary devices of potentially limited capability.

Overview of the ETHOC System

In its current form, ETHOC focuses on paper documents as instances of popular real-world objects of a university campus. The tasks that ETHOC performs are the creation, administration, and intermediation of online resources related to such documents. The system has two interfaces: one for information providers and another one for information users. The first interface is a Web-based author portal for generating and embedding ETHOC barcodes into documents. It also takes care of managing associated online content and document meta data as well as of specifying the associated actions. The second interface offers simple means to interact with virtual counterparts using a variety of different devices. Examples range from WAP-enabled mobile phones equipped with attached barcode readers, over PDAs with wireless connectivity, to the full fledged ETHOC browser for Java-enabled PCs and laptops (see Figure 2.3).

The following scenario illustrates the usage of the ETHOC system from the author's as well as from the client's perspective:

> Bob has just graduated and decides to sell some of his stuff via the student bulletin board before he moves out. He prepares the ad with his favorite word-processing program, including a short description of the items on sale, his home address, and a date for the sale. He uses

ETHOC's author interface to get an associated ETHOC barcode – formatted as a gif image – which he inserts into his ad. Bob enters his contact information and the date and time for viewing the items on sale. Additionally, he takes some pictures of the items and uploads them to the ETHOC system. Finally, he uploads the ad, prints it, and sticks it to the student bulletin board.

Meanwhile, in Alice's introductory computer science class, lecture handouts and exercise sheets are distributed. They contain ETHOC barcodes that are linked to a newsgroup for discussing the exercises, to a source code skeleton needed to solve the programming assignment, and to survey articles relevant to today's class. Alice scans the codes with a tiny barcode reader that is attached to her mobile phone and the links are stored in her personal online history.

After the lecture, a poster announcing an interesting talk attracts her attention. She scans the attached code with her phone. The talk is added to her personal history and a WML page is instantly displayed, shortly describing the speaker's bio as well as directions to the lecture hall. In addition, it indicates that the location of the talk has been rescheduled. Clicking on an item labeled "appointment" inserts an entry for the talk into her mobile phone's calendar. A little later she spots a posting on the student bulletin board announcing various things for sale. She quickly scans the ETHOC code to store it in her access history and decides to look at it later.

At home she connects her laptop to the Internet and – using the ETHOC Java client – looks at the items she has scanned today. She selects the sale entry and looks at the pictures of the items. Using the contact information, she calls Bob for an item she is interested in and verifies the date and time for the sale.

After selecting the exercise entry, the source code skeleton for the programming assignment is automatically downloaded. Following the resource links, she finds two papers that discuss an aspect of today's talk in detail. At the exercise newsgroup she posts a question about a particular topic she did not understand in the lecture.

The next day she is having a coffee with friends at the cafeteria, when suddenly the alarm on her mobile phone goes off. It is the reminder for the talk that she would have missed otherwise.

The scenario illustrates the notion of *environment-mediated communication (EMC)* [80]: Electronic information is dispersed throughout the environment, enabling casual interaction and anonymous communication. EMC describes how communication between persons is mediated by entities of the physical environment. EMC is motivated by the traditional use of the environment for the mediation of information between people, in which messages are left at specific places or particular objects for later retrieval by other people: ads and announcements are posted to bulletin boards, post-its are pasted on office doors or folders. Information dispersed in this way is bound to a physical object or location and therefore reduces

information overload. Information is organized according to physical entities and information of only local relevance is filtered by location.

The short interaction time of just scanning an item that is then automatically inserted into a personal online history is crucial for usability in a mobile environment. It does not require much effort and takes just a few seconds. The user can later review the scanned items and is not distracted from the current activity.

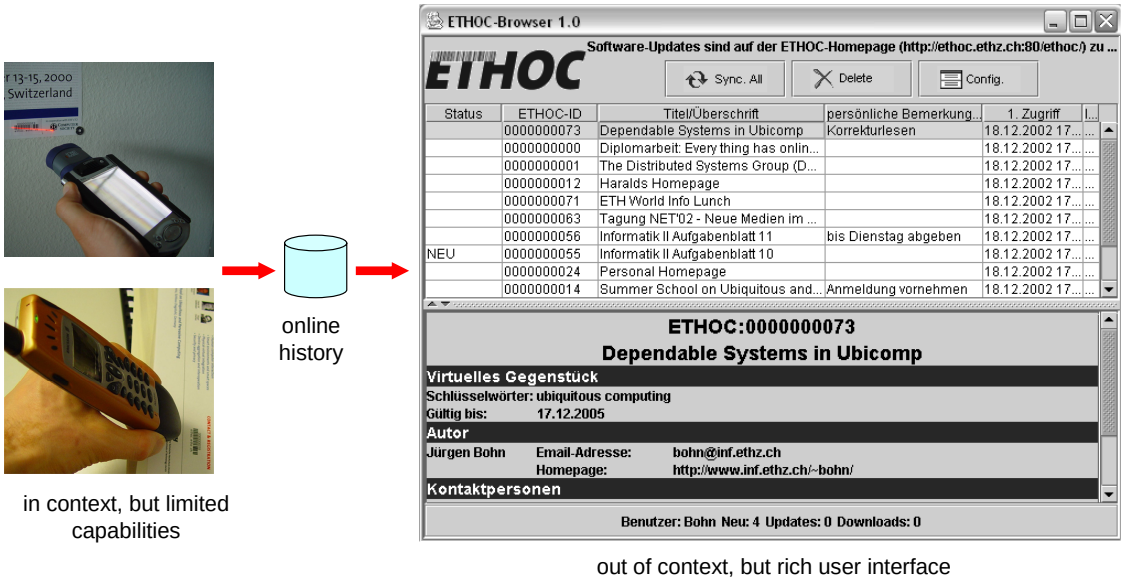


Figure 2.3: The ETHOC browser displays the user's history of scanned objects.

The scenario also shows that a variety of devices can be used to interact with virtual counterparts. Information that is situated in a real world context can be picked up in the originating context using an unobtrusive mobile device. The information can later be reviewed and interacted with in another, more suitable situation, using a stationary device with better display capabilities. That way, mobile and stationary devices complement each other. The former having severely limited display and interaction capabilities, but being easy to use in a mobile context of an everyday real-world situation. The latter being out of context, but offering richer user interface capabilities. This is illustrated in Figure 2.3.

The majority of the information picked up might be immediately useful to users in their current situation while some information might be more useful at a later point in time, potentially in another context. ETHOC supports this by the automatic storage in the user's personal online history of scanned objects.

Figure 2.4 shows what happens when the user scans a barcode with a mobile phone. The wireless communication technology is WAP over GSM/GPRS in this case, but might also be WLAN or Bluetooth. The scanned ETHOC ID is sent as part of an HTTP request to the ETHOC server. In addition to the ID of the scanned object, the request contains information about the client device capabilities, which is used to render the result in a format that the client can display.

The ETHOC server stores the scanned ID in the user's personal online history, where it is available for later retrieval with other devices. Depending on the device capabilities, an HTML or WML page that contains hyperlinks to the document's online content is generated and sent back to the client device. By following the links, the associated online content – for example, background information, contact

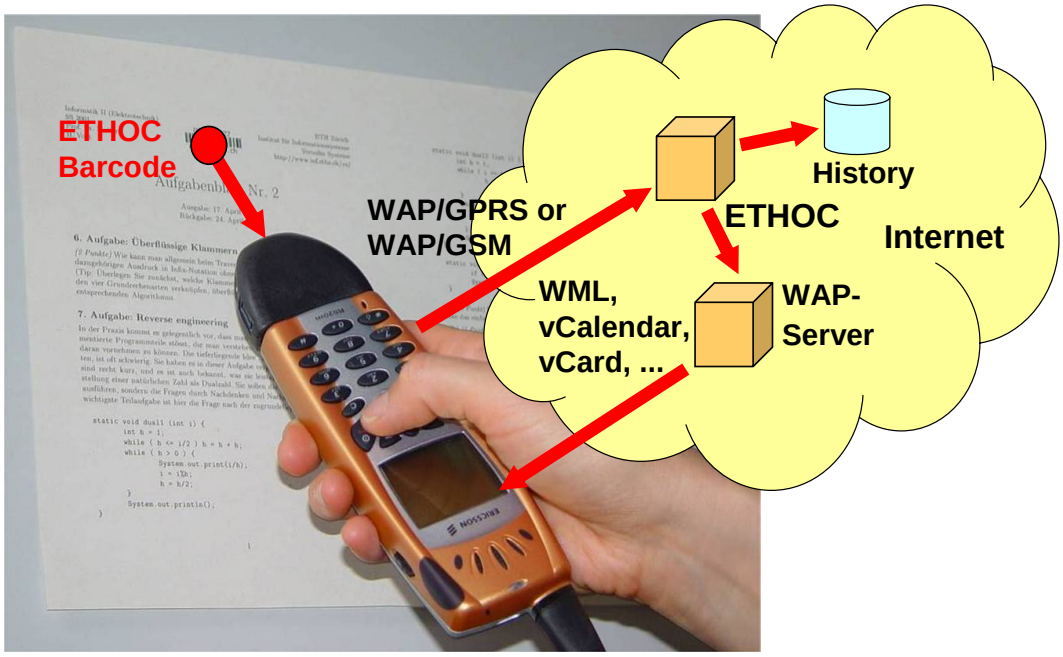


Figure 2.4: Scanning an ETHOC code using a mobile phone with an attached barcode reader.

information, or calendar entries – can then be retrieved. What functionality of the virtual counterpart is available at a time depends on the device currently in use; the automatic insertion of a calendar entry, for instance, only makes sense for a mobile phone or PDA, but not for the Web browser used in a public Internet café.

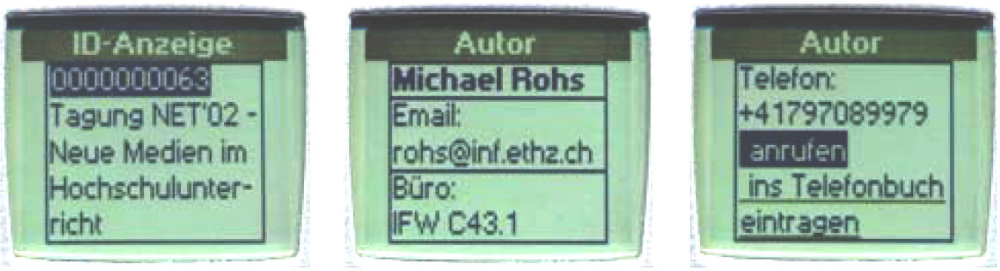


Figure 2.5: On the mobile phone, a WML page is shown as the result of scanning a paper document.

Even though the display size of current mobile phones is severely limited, they are a useful tool in the showcase described above. Using the standard vCalendar [52] format, calendar entries can automatically be inserted into the user's personal calendar to act as reminders for deadlines or events. Using the vCard [51] format, phone calls can immediately be placed. Last but not least, WML has capabilities that come close to those of HTML. Figure 2.5 shows a document as displayed on a mobile phone. The left screenshot shows the ETHOC ID and the title of the document, the middle and right parts show information about the document author. The user can store the author's address in the calendar of the phone.

ETHOC Architecture

The ETHOC system consists of a component for XML-based data processing, an authoring portal, client support for different clients, and a data management component.

Within ETHOC, virtual counterparts are modeled as XML documents. They are transformed into client-readable representations using XSL. There is an XSLT description for each supported output format, i.e. HTML, WML, and XML. Whenever a document is requested, the output format is chosen based on the type of client (or user agent). XML is also used to store the configuration data of the system, which allows for extensibility. Moreover, the broad range of available XML tools can simply be integrated. The authoring components can be easily extended. As a case in point, we added a news client to allow for discussions about exercise sheets after the development of the main system was finished. The virtual counterparts are structured in four sections: author information, document related information, contact information, and actions.

The Web-based authoring tool allows to specify the properties of virtual counterparts, to register as an author and to edit the personal profile. After logon, an author can follow a preconfigured sequence of authoring modules for editing existing or creating new virtual counterparts. The following modules have been implemented: author notification for life cycle management support, feedback questionnaire, news client, and the main module for editing the core attributes. Once an author has entered all required information, a unique ETHOC identifier is created and displayed as a barcode image or a visual marker. The images can then be downloaded and printed.

The data that is managed by the ETHOC system is stored in a MySQL[12] database. This includes keeping track of assigned ETHOC codes and their corresponding virtual counterparts.

Concerning practical experiments, we annotated exercise sheets with ETHOC codes in an undergraduate lecture. This allowed us to timely provide source code fragments and exemplary solutions to students. By means of the integrated news client module, we successfully setup custom news groups for the discussion of specific exercises and coupled them with the corresponding exercise handouts.

2.3 Linking in the Small: Marker-Based Interaction

Having discussed infrastructure issues of linking physical and virtual worlds in situated information spaces, we now turn to interaction concepts involving the use of handheld devices that are relevant in the context of physical-virtual links.

2.3.1 Introduction

If information is linked to physical objects and embedded in the environment, we need ways to access this information and interact with it. In [215], Weiser proposes to embed a multitude of interactive displays of different sizes into the

[12] `www.mysql.com`

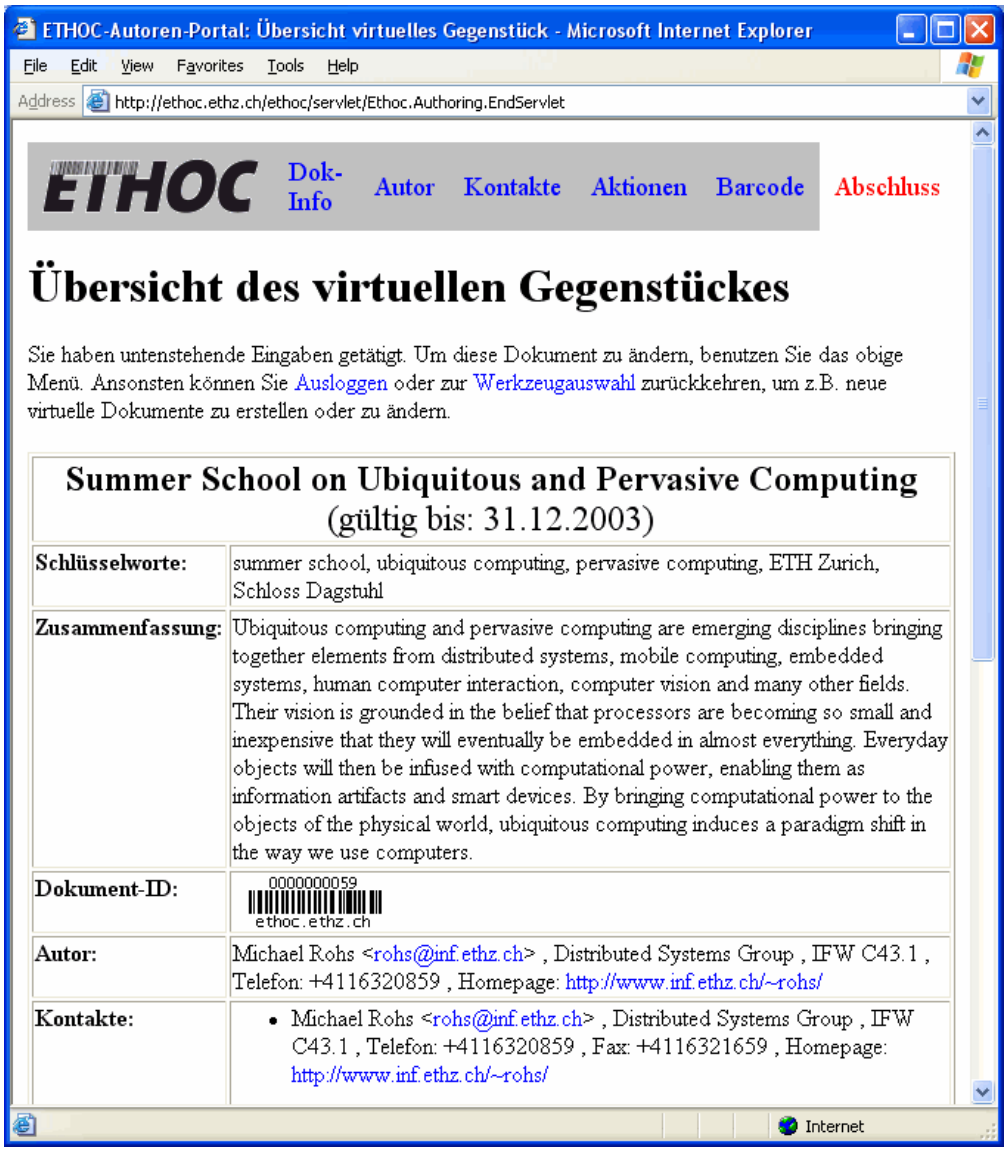


Figure 2.6: The Web-based authoring tool allows to specify the properties of the virtual counterpart.

environment [215]. These displays emit information either as a result of explicit user interaction, upon detecting the presence of a user [212], or depending on other contextual information. A problem with this approach is that ubiquitous displays potentially generate a flood of information. The presentation has to be designed very carefully, in order not to continuously distract users from their activities. In order to address this issue, research on ambient displays [135] tries to develop modes of presentation that do not require the user's direct attention. Another problem is reliability. If a large number of devices cooperate in order to provide a service to the user, it is likely that some of these devices fail. Furthermore, it is still expensive to embed a large number of interaction devices into the environment or into every object; sometimes it is not feasible for aesthetic reasons. Finally, the approach is not very well suited for mobile and nomadic use situations.

In this work, we focus on the use of handheld devices primarily for explicit interaction with the user's current environment and physical objects. The use of the

camera and the recognition of object identities can be seen as an implicit component to the interaction, since the camera as a sensor provides this information, rather than being explicitly input by the user. Handheld devices, like mobile phones, are continuous companions of the user and can easily be personalized according to the user's preferences. They are well suited as mediators for interaction with the physical world if they can sense their surroundings. They are less suited as an ambient display and during periods of user inattention to the display. On the input side, the integration of sensors [99] strives towards a component of implicit input, thereby reducing the amount of explicit input that needs to be provided by the user. We briefly outline other work on physical-virtual interaction with mobile devices and then describe our own approach.

Chameleon [73, 77] is a spatially aware handheld device that acts as a porthole to situated information. The device's location in 3-D space determines the content that is shown on the device's screen. Since we as human beings experience 3-D space and interact with it every day, it is beneficial if handheld devices also possess this ability – ideally at a spatial resolution and accuracy that matches human performance. Unfortunately, the location of the handheld device is not sufficient to determine which object the user wants to interact with, since the device can only sense its position and orientation in space, but no physical objects in its vicinity. Moreover, accurate position detection, for example at outdoor places, is still a difficult problem. If a location system is limited to in-building places, like offices, it cannot be used in nomadic situations, which is a serious limitation. [223] suggests two-handed interaction techniques for spatially aware handheld devices that enable the simultaneous navigation and manipulation. In one implementation, the 2-D position of a handheld device relative to a table surface is sensed. The user can, for example, draw objects that are larger than the screen by moving the device relative to the table surface with the non-dominant hand, while at the same time drawing with the stylus using the dominant hand.

Numerous research projects are dedicated to use mobile devices to interact with physical objects [15, 120, 131, 132, 155, 156, 173]. The objects are typically equipped with some kind of tagging technology such as radio frequency identification (RFID) tags, visual markers, infrared beacons, or Bluetooth beacons. Want et al. [211] combine RFID tags and readers, wireless networking, infrared beacons, and handheld computers to link physical objects to computation. Ailisto et al. [5] explore physical selection schemes, such as proximity and pointing, for nearby devices and objects and compare different tagging technologies. Välkkynen et al. [204] define pointing, scanning, and touching as basic actions for physical browsing. Physical browsing is defined as getting hyperlink information from physical objects using mobile devices. Scanning is a non-directional method that discovers nearby services. Pointing is a selection performed over (short) distance. Users aim at a tag with a laser beam, upon which the tag gets activated and transmits the hyperlink information. Finally, touching is a close-distance selection technique. The handheld device has to be brought in very close proximity of the tag. Touch-based physical browsing is commercialized as a technology called Near Field Communication.[13] It enables the association of two devices by bringing them in close proximity in order to bootstrap a communication. It also allows to connect handheld devices and contactless smartcards.

[13] See Section 2.3.6.

Direct manipulation [189] falls short when attempted on mobile devices [22]. The limited screen space inhibits the detailed graphical presentation of an object. In addition, in a situated environment it does not make sense to show a sophisticated graphical representation of an object on the screen, if it is physically present. InfoPoint [123], also called InfoStick [122], is an interaction device that implements a physical direct manipulation metaphor. It enables the intuitive transfer of data in a physical drag-and-drop style. It allows to transfer data to and from appliances as well as passive objects. An example is to drag a photo from a digital camera and drop it onto a printer. The system consists of a handheld wand with an integrated CCD camera, an LCD display, a "select" button, a "get" button, and a "put" button. The wand is tethered to a belt-worn laptop PC, which is connected to a backend system through WLAN. The backend system is connected to the individual appliances that are embedded in the environment. Physical objects and appliances as well as InfoPoints can "store" virtual resources. To this end, the backend system is also connected to a shared database that maps object IDs to associated resources. Finally, for each InfoPoint there is a database that keeps track of the InfoPoint's resources. Objects are detected by means of 2-D visual markers and the CCD camera. If the InfoPoint detects a marker on a physical object, it queries the shared database for the name of that object and the resources that are associated to it. Moreover, it queries the database of the InfoPoint for its current resources. Resources and appliances contain type information. For the put operation, this allows to constrain the list of resources to those that are actually compatible with the target device. The "select" button can be used to scroll through the list of candidate objects. The "put" and "get" buttons are finally pressed to carry out the desired operation. The functionality resembles Cooltown's e-Squirt mechanism [120], but is more elaborate.

In [162], Rekimoto proposes the *magnifying glass approach* to augmented reality[14] (AR) as an alternative to head-mounted displays that are predominantly used in AR. In this approach, a handheld device with an attached camera is used as a video-see through system.[15] Instead of enlarging the view of the real world, it enlarges it in terms of information. We call this metaphor *symbolic magnifying glass*. Head-mounted displays are cumbersome to wear and, because of their obtrusiveness, are less socially acceptable in everyday life than handheld devices. In a usability experiment, Rekimoto found [162] that task completion times for focusing multiple targets in the environment is shorter when using a symbolic magnifying glass than when using a head-mounted display. Moreover, users much preferred a handheld pointing device over pointing by head movement. The experiment was done with the NaviCam [164], which consists of an LCD screen, a CCD camera, and a gyro sensor (which was not used in the experiment), all tethered to a Unix workstation. The NaviCam recognizes objects in the environment through a color barcode. The output is textual and not registered with the camera image. The magnifying glass approach allows to quickly switch between the augmented view and the real world and to quickly refocus target objects. Furthermore, traditional pen-based input techniques on the touch screen can still be used. The main disadvantage of the magnifying glass approach is that the user cannot operate hands free, which is not acceptable for some applications. In [166], the interaction style

[14]For a discussion of augmented reality, see Section 2.3.5.

[15]See Section 2.3.5.

realized with the magnifying glass approach is defined as *augmented interaction*, in which "a user can see the world through this device with computer augmented information regarding the situation." Augmented interaction "aims to reduce computer manipulations by using environmental information as implicit input."

In our work, we combine the symbolic magnifying glass approach with embodied user interfaces – also called manipulative user interfaces. In these interfaces, handheld devices are equipped with a number of sensors [99], which sense the physical manipulations of the user beyond simple button presses and pen input. Examples are tilting, shaking, squeezing, or holding in a certain orientation. These physical gestures are sensed and mapped to specific operations on the device. Manipulative user interfaces have been used for navigation on a large map [163], scrolling through photos in a digital photo album [16], text entry [222], and for associating devices [102]. Embodied interfaces physically "embody" their user interface and have metaphoric links to a real-world object. For an overview of embodied user interfaces see Section 2.3.4.

The technical basis of our work is a 2-D visual marker system, called *visual codes*. We define the term *marker-based interaction* as interaction with a camera-equipped mobile device within the 3-D space above one or more visual markers. The boundaries of this space are defined by the maximum distance at which the markers are detectable. The position and orientation of the device are sensed relative to the marker and are interpreted in terms of an input vocabulary of physical gestures. Alternatively, they are directly mapped to actions on the mobile device or in the infrastructure. Hansen et al. [90] detect a printed circle with a camera phone and estimate the position of the phone relative to the circle given the position and size of the circle in the camera image. They call the space that is spanned starting at the circle (or another fixed point) to the end of the camera view a *mixed interaction space*, because it is a *physical* space in which a *digital interaction* takes place. If the fixed point is continuously tracked, mixed interaction spaces adhere to the principles of direct manipulation [189], since the actions are rapid, incremental, and reversible, and feedback is visible immediately. The same is true for marker-based interaction, in which the visual marker is tracked in real time in the live camera image. Hansen et al. distinguish between *fixed-point interfaces*, which just provide an orientation point for the interaction, and *identity interfaces*, in which the space has a specific identity. Whereas Hansen's space is an instance of a fixed-point interface, marker-based interaction operates in identity interfaces. The identity is given by the value that is encoded in the marker. Hachet et al. [89] propose a camera-based 3-DOF[16] interface for handheld computers. The method is based on two-handed input. One hand holds the device, the other hand holds a card with color codes. The 3-D position of the camera relative to the card is computed and mapped to different user interface operations such as pan and zoom, rotations, and navigation in tree maps. Since there is only one card with color codes that is reused for multiple applications, a unique identity is not provided. The method could consequently be seen as a fixed-point interface, albeit the interaction space is not generated by a single point, but by a square area with multiple color codes.

[16]Three degrees of freedom of movement in space (x,y,z).

2.3.2 Camera Phones as Ubiquitous Interaction Devices

The Global System for Mobile Communications (GSM) [158] forms a ubiquitous mobile communication infrastructure. In December 2004, digital mobile phone networks had 1.688 billion subscribers, of which about 75% use GSM-based networks. In December 2004, GSM was available in 199 countries on all continents.[17] As Davies and Gellersen observe in [48], off-the-shelf handsets used to access these networks offer far more capabilities than early ubiquitous computing devices, such as ParcTabs. In roughly the same form factor as Tabs, they include PDA applications, text-messaging facilities, voice communications, Web access, and even simple voice command recognition. An important property is that users view them as a commodity that they can use anywhere they are. The handsets are very cheap and exchangeable. The latter is largely a consequence of separating the subscriber identity module (SIM) from the handset. Even if this does not completely match Weiser's vision of hardware devices left lying around for anybody to pick up, the handset hardware is perceived as less important than the access it provides to the digital world and to other people. This can be interpreted as a step towards mental disappearance [195] of computers, freeing users to focus on their goals.

Camera phones are available in everyday settings. They provide constant wireless connectivity – over short distances via infrared and Bluetooth and over long distances via GSM/GPRS and UMTS. High-end models have enormous computing resources, comparable to those of desktop PCs a few years ago. An essential capability is that the integrated camera can be used as a sensor for visual markers and that the recognition algorithms can be executed on the device itself. Real-time marker recognition allows for novel interfaces that would not be possible when sending the camera image to a backend server first. In summary, off-the-shelf camera phones provide a very solid technical basis to act as a highly available mediator between the physical and the virtual world and as a ubiquitous interaction device. The introduction of further sensors would make them an even more interesting platform in this respect. (See also the special issue of IEEE Pervasive Computing, which is devoted to the topic "The Smart Phone – A First Platform for Pervasive Computing" [1]).

As noted in [184], the interaction periods of ultra-mobile devices, such as smart phones, are much shorter than that of larger portable devices, such as laptops: A few seconds to look up a phone number to a few minutes when entering a note or text message. However, mobile phones are continuously in reach of the user and the threshold of use is much lower: Mobile phones are typically constantly switched on, always ready for instantaneous interaction. In addition, mobile devices are used on the move and not in a quiet desktop setting. This means that users are less likely to completely focus on interactions with the device. In a mobile setting, interruptions from the environment are generally unpredictable and frequent. The interaction style is therefore very different from the use of a PC at a desktop. Over the course of a day, longer periods of inactivity will be dispersed with short interactions for accessing information in the environment and longer interactions for phone calls. An important requirement is therefore that interactions with the environment can very quickly be established, without any further setup procedure. Even a Bluetooth discovery process takes too long and is too distracting [191].

[17] `www.gsmworld.com/news/statistics/index.shtml`

2.3.3 Tangible User Interfaces

Tangible user interfaces (TUIs) – also called graspable or physical user interfaces – are an area within human-computer interaction where users manipulate physical objects that embody digital information. TUI manipulations exploit human spatiality, "our innate ability to act in physical space and interact with physical objects" [187]. TUIs take advantage of these rich physical manipulation skills, thereby simplifying interaction with digital tools and data. By stepping out of the confines of rectangular displays into the physical world, TUIs significantly extend the interaction design space in comparison to screen-based graphical user interfaces (GUIs). TUIs are in line with the ubiquitous computing vision [215], in which computation is embedded in physical artifacts and available throughout our everyday environment.

TUIs were first proposed by Fitzmaurice et al. [75, 76] as *graspable user interfaces*: "a physical handle to a virtual function where the physical handle serves as a dedicated functional manipulator." Ishii and Ullmer defined TUIs as "devices that give physical form to digital information, employing physical artifacts as representations and controls of the computational data" [113] and later as "interfaces in which physical objects play a central role as both physical representations and controls for digital information" [203].

To allow for a comparison and categorization of different research efforts, Fishkin developed a taxonomy [70] that uses *embodiment* and *metaphor* as its two axes. He characterizes TUIs very broadly as systems in which (1) some input event as a physical manipulation on some everyday physical object occurs, (2) a computer senses this event and alters its state, and (3) the system provides feedback. This characterization includes all of the research efforts in the wide spectrum of TUIs, but is far too general to allow for a meaningful categorization. The taxonomy thus uses embodiment and metaphor to cover the design space of TUIs. Embodiment has the categories *full*, *nearby*, *environmental*, and *distant*. Metaphors are classified as *none*, *noun*, *verb*, *verb-and-noun*, and *full*.

Our stationary approach to interaction with product packaging, which is presented in Chapter 8, is an example of a tangible user interface.

2.3.4 Embodied User Interfaces

Embodied user interfaces (EUIs) [71, 72, 91], sometimes also called *manipulative user interfaces*, treat physical manipulations on the body of a handheld device as an integral part of its user interface. Embodied user interfaces try to extend the language users can provide as input for handheld devices and artifacts by incorporating a variety of sensors into them. Example sensors are accelerometers, tilt sensors, capacitive coupling, and infrared range finders. Users can interact with such a device by tilting, translating, rotating, or squeezing it. The physical manipulation and the virtual representation – i.e. input and output – are integrated and tightly coupled within the same device. Whereas in traditional GUIs virtual objects can be *directly manipulated*, embodied user interfaces allow for the direct manipulation of physical artifacts that embody the user interface. Embodied user interfaces mediate – i.e. sense and interpret – the actions of the user in an unobtrusive way. By making the user's task and the actions needed to accomplish the task similar, the device ideally becomes invisible in use and is no longer perceived as

mediator. The means to achieve this goal is to take advantage of everyday spatial skills and make the interaction more similar to physical manipulations of ordinary non-computational physical objects.

Fishkin et al. [72] identify three characteristic features of embodied user interfaces:

- embodiment,
- coincidence of input and output, and
- metaphorically appropriate manipulations.

The handheld device *embodies* a real-world task that is typically performed on a particular physical object. There is *coincidence of input and output* in the device. And finally, through its size and shape the handheld device offers *specific and familiar affordances* for particular kinds of actions. Embodiment, coincidence, and appropriate manipulations imply that the embodied task should correspond to an analogous real-world task. The stronger the metaphor, the more intuitive the manipulation, and the more transparent and invisible the interface becomes.

Fishkin et al. differentiate between different types of manipulations, such as *spatial* (the device is rotated, shaken, etc.), *structural* (the device is squeezed, folded, etc.), or *environmental* (the device is heated, lit, etc.). Each manipulation has different affordances and user expectations.

Ergonomic considerations constrain which physical manipulations are advantageous for a certain task and which ones should be avoided. For example, in empirical studies [226, 227] it was shown that assignment of the muscle groups in manipulating 6 DOF[18] input devices is a critical factor for user performance. As predicted before [42], in fine-grained manipulations fingers (the small muscle groups) have performance advantages over the wrist, elbow, and shoulder (the large muscle groups). Consequently, the affordances of input devices, like their shape and size, should be designed such that the fingers are included in their physical manipulation.

Other empirical evidence [9] suggests that the type and amplitude of movement have to be taken into consideration when comparing performance of different limb segments. Finger, wrist, and forearm have different optimal movement amplitudes. It has been found, for example, that the optimal angular displacement of the forearm is 12°, of the wrist is 25°, and of the finger is 45° [179]. Moreover, the human motor system has varying ability to control the different limb segments, resulting in different input bandwidths. Overall, designs for embodied user interfaces should consider the differences in form, function, and performance of the limb segments and use the one – or the combination of limb segments working in synergy – that are most suitable for the task at hand.

Fishkin et al. [72] formulate a number of design principles for embodied user interfaces. We briefly summarize these design principles here.

- **Embodiment principle.** The device's hardware and displays (visual, auditory, tactile) should embody the real-world task and its states. Directly physically manipulating the device results in changes to the task state.

[18] Six degrees of freedom of movement (x,y,z) and orientation (pitch,roll,yaw) in space.

- **Physical effects principle.** The mapping from physical manipulation to virtual system action should be such that users perceive it as a physical effect of the manipulation. An example would be squeezing a device's case in order to compress a file that is stored on the device. In this case, compression is a physical effect of squeezing, which is consistent with real-world experience. Another example would be shaking a device in order to randomize elements shown in a list.

- **Metaphor.** If there are no physical interpretations for a given system action as the result of a physical manipulation, the physical effects principle cannot be applied. There is, for example, no obvious physical cause for compiling source files. In this case, the next best solution is to find metaphorical analogies for the mapping from physical manipulation to system action. An example would be orbiting the device around some fixed external point in a rotary movement, like the handle of the crank of a meat grinder. The analogy would be that compiling is like transforming a "raw material" into a more "digestible" state. In rare cases, no metaphor might apply, in which the designer has to resort to abstract mappings.

- **Kinesthetic manipulation principles.** *Kinesthesia* or *proprioception* is the sense of the movement and posture of body parts relative to other neighboring parts of the body.[19] The word *kinesthesia* is composed from the Greek words *kinesis* for movement and *aisthesis* for feeling. The Latin word *proprius* stands for "one's own." Unlike other senses that provide feedback about the state of the outside world, proprioception provides internal feedback on the state of the body limbs. Kinesthesia is essential for *muscle memory* and *hand-eye coordination*. It enables humans, for example, to touch their nose with their eyes closed with an error of no more than 2 cm.

  The kinesthetic manipulation principles concern the type of motions and postures that should be used in embodied user interfaces. The most important principle relates to the *comfort* of manipulations and sequences thereof. Inconvenient or even unhealthy manipulations have to be avoided. The parameters of the physical manipulation – like speed, force, and precision – should be *appropriate* for the task. A task that requires fine precision, for example, should not be bound to arm motion, since this would not be a good match between the high precision that the task requires and the coarse precision that arm movement is only able to produce. The roles of the hands are often complementary: the non-dominant hand sets the context for the dominant hand. Finally, the manipulations employed in an embodied user interface should be compatible with socio-cultural factors when used in the public, i.e. they should not be intrusive or embarrassing or break cultural norms.

- **Sensing principles.** In embodied user interfaces, sensors provide raw input. Ideally, they should be invisible in use, such that the user perceives interaction as interaction with the device and not with the sensors – the user would be unaware on the kind and functionality of the embedded sensors. Unfortunately, sensing technology imposes limits on what manipulations can

[19] `en.wikipedia.org/wiki/Proprioception`

be sensed [99]. Sensors only provide a limited and possibly erroneous view on the user's gestures and manipulations. Sensor data has to be interpreted with respect to the user's task and the available command vocabulary. This has to be done in a robust way in order to avoid misinterpretations. In addition, the time lag has to be short for effective and enjoyable interactions.

- **Communication principles.** The vocabulary and grammar of physical manipulations within an embodied user interface defines an artificial language for communication between human and machine. User manipulations on a device emit a stream of commands to the device. The integrated output (visual, auditory, or tactile) provides a feedback channel for communication to the user. In this kind of dialog, the user drives the interaction, while the device receives and reacts to input commands. Fishkin et al. derive a number of communication principles from gestural systems such as sign languages. The *start signal* that activates an interaction should fit into the gesture space, yet be different from other gestures. Suitable gestures are wrist-flicking and squeezing. The *stop signal* might be implemented implicitly, for example indicated by a temporal pause. A gesture should correspond to *appropriate linguistic units*, such as nouns, verbs, adverbs, and adjectives. Finally, *gestural sequencing* concerns the arrangement of individual gestures in temporal sequence. Fishkin et al. suggest to transmit the most significant information as the first part of a gestural sentence, followed by refinements, as is known from many sign languages.

In embodied interfaces, users need to know the vocabulary of physical manipulations, i.e. the set of possible commands. Depending on the context, users might also be hinted at the subset of operations that is applicable in a given situation. The set of primitive manipulations of our system and our approach to combining and indicating them to the user is presented in Chapter 4. In our framework, we do not follow all of the communication principles described above, since we do not base the interaction primitives on known linguistic units, such as nouns, verbs, etc.

Our work can be seen as an instance of an embodied user interface, since we treat the camera phone as the embodiment of an optical magnifying glass. Inspired by Figure 3 in [72], Table 2.2 compares the properties, manipulations, and types of feedback of an optical magnifying glass and its embodied counterpart.

In the real world, we use an optical magnifying glass to be able to observe more detail when looking at an object. We can see what we were not able to observe without the magnifying glass as a mediator. Given that the camera resolution is high enough, the camera phone as a symbolic magnifying glass could be implemented in the same way, i.e. so as to show more detail. This would result in a strong metaphor and even meet the physical effects principle. However, the embodied device allows for further mappings: arbitrary information can be shown upon manipulating the device relative to some marker-equipped object. In this case, the metaphor gets weaker, but the type of physical manipulation is the same as for an optical looking glass. Whether the physical effects principle is fulfilled and whether the metaphor is intuitive depends on the chosen mapping. Technically, arbitrary mappings are possible, which means more freedom for design than in the real world, potentially at the cost of the loss of metaphorical analogies. On the device side, there are manipulations which do not have an effect on the feedback of the real-world object,

Real-World Task	Device-Embodied Task
Physical object:	**Physical device:**
optical magnifying glass	symbolic magnifying glass (alias camera phone with marker tracking)
Real properties of objects:	**Represented objects and properties:**
provides a magnified view on the focused area	provides a symbolically augmented view on the focused area
the lens is optically transparent	the display is perceived as optically transparent, since it shows a live camera image of the focused area
the lens is mounted in a frame with a handle	the display is integrated in the device body (that is designed for holding it)
fixed functional mapping	arbitrary mapping
Manipulations on objects:	**Manipulations on device:**
move the magnifying glass closer to or further away from the object	move the camera phone closer to or further away from the marker
change focused area	change focused area, potentially change focus to another marker
tilt the magnifying glass	tilt the camera phone
(no corresponding manipulation)	rotate the camera phone, wait in a static posture, press a key on the keypad
Feedback from objects:	**Feedback from device and representation:**
magnification changes with distance	level of detail changes with distance, other mappings are possible
different magnified views are shown depending on focused area and tilting	different aspects are shown depending on focused area, tilting, rotation, time stayed in an area, and pressed key
(no corresponding feedback)	auditory feedback, tactile feedback

Table 2.2: Properties, manipulations, and feedback of an optical magnifying glass and its embodied counterpart.

such as rotating around the optical axis. In the embodied magnifying glass, there are also new types of feedback, for example auditory and tactile feedback. These are not available in the real-world object. Nonetheless, even if the embodied device has richer means of input and output for which there is no analogy in the real world, it still leverages familiar spatial manipulation skills to enhance the interaction.

2.3.5 Augmented Reality

Augmented reality (AR) [7, 8, 66] is concerned with supplementing the user's sensory perceptions with computer-generated information. Typically, AR research focuses on the sense of sight, with a head-mounted "see-through" device as the typical setup. Other senses, such as hearing, touch, and smell might also be employed by an AR system. Azuma et al. [8] define the characteristic features of AR systems as:

- combines real and virtual objects in a real environment;
- runs interactively, and in real time; and
- registers (aligns) real and virtual objects with each other.

Whereas *virtual reality (VR)* immerses the user in a completely computer-generated world and thereby replaces the real world, augmented reality only supplements it. These computer-generated supplements that are integrated in the user's view of the real world have the potential to make tasks easier to perform, since hopefully relevant information is provided at the right position at the right time. Potential application domains are medical visualization (visualization and training aid for surgery), manufacturing and repair (superimposed 3-D animated drawings instead of numerous complicated manuals), annotation and visualization (show where the pipes and electric lines are inside the walls), entertainment (sports broadcasting, real time annotations on race cars), and military aircraft navigation and targeting (aiming the aircraft's weapons by looking at the target).

A central basic problem of AR is tracking the position and orientation of physical objects and – if a head-mounted display is used – tracking the viewer's head in order to accurately align the computer-generated overlay graphics to objects in the real-world view. This alignment problem is known as *registration*. The system has to be able to maintain accurate registration when users turn their heads or objects move about. If system lag is too high, the illusion of a unified augmented view is destroyed, and real and augmented view are perceived as separate. The visual system is very sensitive to these issues. In particular, view changes are problematic, since a high angular accuracy is required. Objects at a distance appear to move very rapidly, if the head is turned.

On the hardware side, the key components of an AR system are displays, trackers, and graphics computers. Three types of visual displays are used in AR: head-mounted displays (HMDs) – also called head-worn displays (HWDs) –, handheld displays, and projection displays. The most traditional output device type in AR is HMDs. HMDs have already been used by Sutherland in the 1960s [198]. There are two categories of HMDs: *optical see-through* and *video see-through*. Optical see-through HMDs have a half-silvered mirror that transmits light from the environment and also reflects light from a small projector, which emits the computer-generated overlay. Optical see-through HMDs let the user see the real world with full resolution and field of view. On the downside, the overlays are not fully opaque. Furthermore, there can be eye accommodation problems, since focusing objects in the real world requires normal eye accommodation, whereas the overlays appear at a fixed distance on the screen. In, video see-through HMDs, the user looks at an opaque display that shows the camera image supplemented by computer-generated overlays. Here, eye accommodation is not a problem, since all objects are projected in the plane of the display. Moreover, real and virtual view always match. Unfortunately, the resolution of the real-world image is lower (depending on the camera resolution). For both categories of HMDs it is important that the camera is positioned such that its optical path is close to that of the user's eye, in order to avoid parallax errors. Projection displays are small head-mounted projectors or stationary projectors located in the environment, typically ceiling-mounted as in [220].

Finally, handheld AR systems use devices with flat-panel LCD displays and integrated cameras to provide video see-through augmentations. The handheld device acts as a magnifying glass or window onto the real world. Examples are NaviCam [164] and our own approach. NaviCam uses color-coded stickers to track objects in the environment. A back-end workstation processes the images generated by the NaviCam and creates an augmented view consisting of textual information. Mobility is limited due to tethering. Wagner et al. [207] is a more recent example of work in handheld AR. The "Invisible Train"[20] is a simple multi player game in which a virtual train is overlaid onto physical toy rail-tracks. The course of the train can be controlled by altering the track switches and the train speed can be adjusted. The game is controlled using touch-screen input. By contrast, our work is more resembling embodied user interfaces and has a stronger focus on physical actions as input. In order to enable marker-based tracking, Wagner and Schmalstieg [208] have ported the ARToolKit [118] to the PocketPC platform. Moehring et al. [146, 147] present a video-see through augmented reality system for camera phones. Their system supports optical tracking of passive paper markers and the registration of 2-D and 3-D graphics.

With advances in hardware capabilities, camera-equipped handheld devices gain more and more interest as a platform for augmented reality systems. HMDs are still cumbersome and – at least in their current form factor – it is difficult to imagine that they will be used on a day-to-day basis outside the laboratory or special workplace settings. In contrast, handheld devices are very well suited for certain applications of AR. Camera phones in particular are small and unobtrusive and are a constant everyday companion for many people. However, some application domains require hands free operation or a larger screen area.

2.3.6 Tagging Technologies

There is a multitude technologies for implementing the physical link to a virtual counterpart. We focus here on passive tagging technologies. For a taxonomy of tagging technologies, see Section 2.2.3 of [190]. Passive tags do not have their own power supply, but depend on an external electromagnetic field – in the case of RFID tags – or visible light – in the case of 1-D or 2-D barcodes. The basic functionality common to all passive tags is that they are physically attached to an artifact and that they provide an identification number to the appropriate sensing device. We also discuss Near Field Communication, since it has been developed for real-world interaction with mobile devices, although it is not a purely passive tagging technology.

1-D barcodes. 1-D barcodes are ubiquitous on consumer products. Such codes typically store a Universal Product Code (UPC) or a European Article Number (EAN). 1-D barcodes are typically detected with laser scanners. Because of their thin lines, 1-D barcodes require a macro lens to be detected with a low resolution camera. 1-D barcodes have also been extensively used in ubiquitous computing projects to link everyday products to online information. Examples are WebStickers [131, 132], Informative Things [15], Cooltown [120], and EntryPoints [173]. Mobile as well as stationary sensing devices have been used in these projects. In the

[20] `www.studierstube.org/invisible_train`

EntryPoints project, for example, we used Eriksson phones with attached Airclic[21] barcode readers, iPAQs with SocketScan[22] laser scanners for the SD card slot, and Symbol[23] CS2000 scanners. The latter is a handheld device that simply stores scanned codes internally. Stored codes can later be uploaded to a PC via the serial port.

2-D barcodes and 2-D visual markers. 2-D barcodes and 2-D visual markers have a higher data capacity per area than 1-D barcodes. We distinguish between 2-D barcodes, like the PDF417 [111], which are typically detected with laser scanners, and 2-D visual markers, like CyberCode [165], which are detectable by CCD cameras. There are ISO-standardized codes such as Data Matrix [110], MaxiCode [109] and QR Code [108] (see Figure 2.7). Other 2-D visual markers are the result of research efforts. Examples are CyberCode [165], TRIP [53], and our Visual Codes [169, 170, 175].

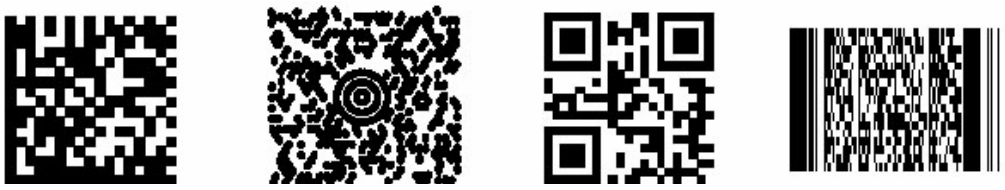


Figure 2.7: ISO-standardized 2-D visual markers.

Existing 2-D markers differ in the application area for which they are designed, the number of encoded bits, the robustness against failures, as well as in the quality of the hardware which is necessary for their recognition. All of the ISO-standardized codes have a multitude of options and features, such as multiple variants of error correction, encoding of different character sets, and multiple masking patterns.

Data Matrix [110] markers are laid out as a rectangular arrangement of black and white squares. The data area consists of 10×10 to 144×144 squares, which are protected by a Reed-Solomon code [160]. The data area is surrounded on two sides by an L-shaped finder pattern and on the opposite sides by alternating black and white squares. Larger markers can be created from these simple ones. The L-shaped frame is contained multiple times in the larger markers.

MaxiCode [109] is a 2D visual marker with a fixed size of 28.14×26.91 mm. A unique pattern, which is surrounded by a hexagonal frame, allows for the localization of the marker. The maximum capacity is 93 alpha-numerical characters. MaxiCode is used extensively in the shipping industry, for example by UPS.

In the QR Code [108] ("quick response code"), special localization patterns are placed in three corners of the marker. The marker size varies from 21×21 to 177×177 black-and-white squares. The marker can store up to 2953 bytes and has four selectable error correction modes. QR Codes are very common in Japan for recognition with camera phones.

PDF417 [111] ("portable data files") is a 2D barcode, which consists of multiply vertically stacked lines, each of which corresponds to a 1D barcode. The code can comprise 3 to 90 lines, each of which can store one to 30 characters.

[21] www.airclic.com
[22] www.socketscan.com
[23] www.symbol.com

Passive radio frequency identification tags. Passive radio frequency identification (RFID) tags [69, 210] enable the wireless identification of objects over a short range (up to a few meters). They consist of a small chip and an antenna. An external reader creates an electromagnetic field that serves as an energy supply and a communication medium. This external energy supply allows them to power their internal processor and send a unique identifier plus a few hundred bytes of data in a few milliseconds. There are protocols that recover from collisions, which occur if multiple such labels are in the field and are simultaneously activated. The microchip on RFID tags is typically very small. The overall size of the tag is mostly determined by the size of the antenna. Passive RFID tags can be split into four different categories, depending on the used radio frequency [127]: low frequency tags (100-135 kHz), high frequency tags (13.56 MHz), ultra high frequency tags (868 MHz, 915 MHz, 950-956 MHz), and microwave tags (2.45 GHz and 5.8 GHz). Low frequency tags are mainly used for anti-theft systems and animal identification. High frequency tags are the most common variant. They are used in applications such as tracking books in a library, airline baggage tracking, and pallet tracking. RFID tags can be invisibly integrated into everyday objects, since they do not require a line of sight as visual tags. Some kinds of RFID tags are rewritable, i.e. they can act as a small data store in addition to supplying their identifier.

Near field communication. Near Field Communication (NFC) enables local interaction between mobile devices, between mobile devices and stationary devices, and between mobile devices and "smart" objects [60]. When two devices that are equipped with NFC technology are brought in touching distance, they perform a handshake and form a peer-to-peer network.

NFC is an official ISO standard [112] and is now promoted by the NFC-Forum,[24] consisting of Nokia, Philips, Sony, and other companies. Similar to RFID, NFC works by inductive coupling. Like high frequency RFID tags, NFC operates in the unregulated 13.56 MHz band. It has a maximum communication range of approximately 10 cm. Three communication data rates are defined: 106 kb/s, 212 kb/s, and 424 kb/s. The extremely limited communication range and relatively low data rates result in very low power consumption. There are two possible roles in the communication process: initiator and target. The initiator starts and controls the data exchange, whereas the target answers requests from the initiator. There are two communication modes: one-way (passive) and two-way (active). In active mode both devices generate their own electromagnetic field. In passive mode the initiator generates the electromagnetic field and the target uses load modulation to transfer data. The latter scheme is similar to RFID, except that the target NFC unit still has to be powered internally. It just does not produce the external RF field. With RFID, even the internal operation of the chip is powered from the field of the reader (indirectly by charging a capacitor). Active NFC units are small and energy conserving enough to be integrated into mobile phones. Passive NFC units are very cheap and even smaller than RFID tags. Interestingly, the NFC protocol is compatible with contactless smart card protocols, in particular with Felica from Sony and Mifare from Philips. This means that an NFC-equipped mobile phone looks like a smart card to the other communication partner and can be used to perform secure transactions, such as payment.

[24] `www.nfc-forum.org`

The NFC-Forum not only defines the basic technology, but in its working groups also investigates aspects such as smart card emulation, configuration of other wireless networking technologies, "smart" poster applications, and security. Various categories of touch-based interactions are defined for NFC:

- *Touch and go.* Access control applications in which the NFC device stores the ticket. Picking up a URL from a "smart" poster.
- *Touch and confirm.* Mobile payment applications in which the user has to confirm a transaction.
- *Touch and connect.* Linking devices to form a peer-to-peer network, in order to exchange files or synchronize address books. In this case, NFC only bootstraps other communication networks, such as Bluetooth or WLAN, which have a higher data rate than NFC.
- *Touch and explore.* Exploration of another device's capabilities and services.

The proposed application scenarios include NFC tags on product packaging to take part in a competition, touching a poster to order a taxi, calling predefined numbers, accessing the weather forecast or the train schedule, and paying for your parking lot, bus tickets, and movie tickets.

Attribute	**Visual Codes**	**RFID**	**NFC**
Domain of application	end-user interaction	automating the supply chain	end-user applications
Reliability	vulnerable to dirt and extreme light conditions, requires line of sight	random detection failures depending on tag orientation, metal surfaces and liquids impair reliability	probably depends on operation mode, for passive mode similar to RFID
Detection capacity	$\geq$ 50 tags per image (640×480 pixels)	10-30 tags/s for LF and HF systems, 100-500 tags/s for UHF systems	always 2 devices communicating
Transmission range	depending on the size of the visual code	about 1m LF and HF tags, up to 7m for long range UHF tags	about 10 cm
Cost per tag	negligible	about $0.40, depending on quantity	unknown
Availability of reader devices	many mobile phones already have integrated cameras	stationary installations mainly in the supply chain, expensive	will be available for end-users at acceptable prices
Forgery proofness	nonexistent	medium	very high, since implements smart card functionality
Privacy	is not an issue, since no automatic detection	great concern among consumer protection agencies	is less of an issue because of limited transmission range
Environmental impact	easily recyclable if printed on paper	severe negative impact if not recyclable	severe negative impact if not recyclable

Table 2.3: Comparison of RFID, Visual Codes, and NFC.

Comparison of RFID and visual codes. RFID and visual codes offer very different, potentially even complementing, feature sets. In Table 2.3, we compare both technologies with respect to a number of relevant attributes (adapted from [142]). RFID is optimized for the automatic detection of a large volume of tags in a short amount of time. It is therefore suited for automating the supply chain. In contrast, the automation potential of visual codes is very limited. Visual codes require a direct line of sight between the marker and the camera, whereas RFID tags can be invisibly embedded into an artifact. However, depending on factors such as the placement of tags in the field of the reader or the presence of metals and liquids nearby, detection failures can occur. Visual codes and NFC devices are designed as a user interface technology for conscious interaction by the user. The visual code system determines the orientation of the camera relative to the marker. Orientation detection is not feasible with current RFID or NFC technology. The wide availability of reader devices for visual codes lowers the barrier of entry for pilot projects and simplifies large-scale user-oriented prototyping. The cost of visual codes is negligible, whereas RFID tags have to become much cheaper for a wide-spread adoption. Finally, visual codes as well as NFC are not likely to cause privacy debates like RFID, since they cannot be scanned unnoticed.

2.4 Summary

In this chapter we have surveyed other work and introduced concepts related to the topic of linking physical and virtual worlds. The first part introduced the vision, enablers, and challenges of ubiquitous computing. It also discussed interaction in ubiquitous computing. The second part discussed infrastructure aspects and introduced influential example projects. We have introduced the entry points concept and its application in a smart campus environment. The third part focused on interaction aspects of physical-virtual links and discussed the symbolic magnifying glass metaphor, marker-based interaction and mixed interaction spaces, embodied user interfaces, and various tagging technologies.

Chapter 3

Visual Codes: A Marker System for Camera Phones

In this chapter we present a system that turns camera phones into mobile sensors for 2-dimensional visual codes. The proposed system induces a code coordinate system and visually detects phone movements. It also provides the rotation angle and the amount of tilting of the camera as additional input parameters. These features enable applications beyond simple marker detection, such as item selection and interaction with large-scale displays. With the code coordinate system, each point in the viewed image – and therefore arbitrarily shaped areas – can be linked to specific operations. A single image point can even be associated with multiple information aspects by taking different rotation and tilting angles into account.

3.1 Introduction

With the integration of cameras, mobile phones have evolved into networked personal image capture devices. As image resolution improves and computing power increases, they can do more interesting things than just taking pictures and sending them as multimedia messages over the mobile phone network. Programmable camera phones can perform image processing tasks on the device itself and use the result as an additional means of input by the user and a source of context data.

In this chapter, we present a visual code system that turns camera phones into mobile sensors for 2-dimensional visual codes. We assume scenarios where camera phones are used to sense a scene that contains one or more visual codes. By recognizing a code tag, the device can determine the code value, the targeted object or image element (even if the object or image element itself is not equipped with a code tag), as well as additional parameters, such as the viewing angle of the camera. The system is integrated with a visual phone movement detection scheme, which provides three degrees of freedom and turns the mobile phone into an optical mouse. Code recognition and motion detection are completely performed on the phone itself. The phone's wireless communication channel is used to retrieve online content related to the selected image area or to trigger actions (either in the background infrastructure or on a nearby larger display), based on the sensed code and its parameters.

These features enable local interaction with physical objects, printed documents, as well as virtual objects displayed on electronic screens in the user's vicinity.

Mobile phones are in reach of their users most of the time and are thus available in many everyday situations. They are therefore ideal bridging devices between items in the real world and associated entities in the virtual world. Visual codes provide visible *entry points* into the virtual world, starting from the local surroundings. This offers a natural way of local interaction and strengthens the role of mobile phones in a large number of usage scenarios. The visual code system also provides the basis for superimposing textual or graphical information over the camera image in close real-time in the sense of augmented reality. This entails a manifold of application possibilities in situations where information is to be closely linked to physical objects. An example is the maintenance of devices or apparatuses in the field: Individual parts of an apparatus are associated with visual codes. With the help of the visual code system, graphical information which is aligned with the items in the image, is superimposed over the camera image.

The novelty of the proposed system is its code coordinate system, the visual detection of phone movement, and the determination of the rotation angle and amount of tilting. These features enable interesting applications, beyond simple item selection, such as interaction with nearby active displays. The recognition algorithm precisely determines the coordinates of a targeted point in the code coordinate system, which is independent of the orientation of the camera relative to the code tag (distance, rotation, tilting) and is also independent of the camera parameters (focal distance, etc.). This enables the association of each point in the viewed image – and therefore arbitrarily shaped areas – with specific operations. A single visual code can be associated with multiple such areas and a single image point can be associated with multiple information aspects using different rotation and tilting angles.

3.2 Related Work

Sony's *CyberCode* [165] is related to our approach, but does not operate on mobile phone class devices and does not use phone movement and other additional parameters for interaction in the way we propose. CyberCodes store 24 bits of data. In addition to the ID, the 3-D position of the tagged objects is determined. Proposed applications for CyberCodes comprise augmented reality systems, various direct manipulation techniques involving physical objects, and indoor guidance systems.

TRIP [53] is an indoor location tracking system based on printable circular markers, also called "TRIPtags." It employs CCD cameras plugged into standard PCs for code recognition, 3-D location, and orientation detection. TRIPtags have an address space of just 19683 ($= 3^9$) possible codes, which makes them impracticable to encode universally unique IDs, like Bluetooth MAC addresses. In contrast to our system, TRIP is designed for use with stationary cameras which are distributed in a networked environment. It relies on a CORBA infrastructure and a centralized recognition engine named "TRIPparser." In our system, code recognition is completely done on the mobile phones, which enhances scalability, and code sightings are distributed wirelessly.

The *FieldMouse* [192] is a combination of a barcode reader and a pen-shaped mouse. The mouse detects relative (Δx, Δy) movement. If the location of a barcode on a flat surface is known to the system, absolute locations can be computed by

first scanning the barcode and then moving the FieldMouse. This enables various kinds of paper-based GUIs.

A number of commercial efforts exist to recognize product codes with mobile phones. An example is *AirClic*,[1] which provides tiny barcode readers that can be attached to mobile phones. The disadvantage of this approach is the necessity of an additional device, which increases the physical size and weight of the mobile phone and consumes additional energy. Barcode readers also do not provide the orientation and selection features of camera-based approaches.

SpotCodes[2] are a commercialized derivative of the above-mentioned TRIP tags for use with camera phone devices. They recognize the rotation of the code tag in the image, but do not provide an orientation-independent code coordinate system and do not detect relative camera movement independent of codes in the camera image. A number of interaction possibilities are described on the Web page[2] and in [133], such as rotation controls and sliders.

An increasing number of companies offer mobile phones with the ability to read *QR Codes* [108]. They implement the core functionality of decoding QR Codes. They do not, however, have the code coordinate system, rotation, tilting, and visual movement detection features that are integrated in our system.

The same applies to Semacode,[3] which uses standard *Data Matrix* [110] codes to implement physical hyperlinks and load Web pages in the phone's browser. Example applications are live urban information, such as the position of GPS-equipped buses, information on nearby shops and services, and semacodes on business cards and conference badges.

3.3 Visual Code Layout and Recognition Algorithm

3.3.1 Hardware Limitations

The mobile phone devices we consider have severely limited computing resources and often lack a floating point unit. Hence, the use of floating point operations has to be minimized. The typical phone camera generates low to medium quality color images with a resolution of 640×480 pixels (VGA[4]) for still images and 160×120 pixels (QQVGA[4]) for view finder images. The relatively poor image quality determines the minimal size of code features that can be reliably recognized. The code features therefore have to be more coarsely grained than those of most available visual markers. In our evaluation it became clear that color should not be used as a code feature, because of the large differences in color values, depending on varying lighting conditions. Moreover, color codes are more expensive to print and harder to reproduce than simple black-and-white codes.

The evaluation of example pictures taken with the integrated camera of a Nokia 7650 yielded the following results:

[1] `www.airclic.com`

[2] `www.highenergymagic.com`, `www.shotcode.com`

[3] `semacode.org`

[4] `en.wikipedia.org/wiki/Video_Graphics_Array`

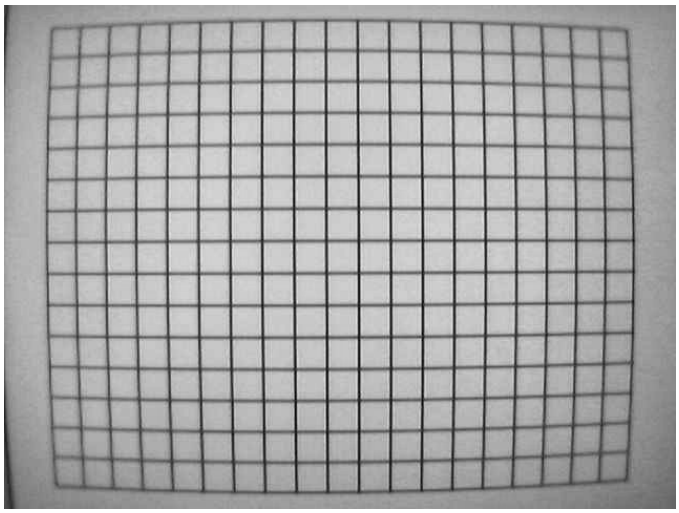


Figure 3.1: Barrel distortion.

- *Barrel distortion.* The image shown in Figure 3.1 was taken from a grid of straight lines. The grid lines in the resulting image are not straight, but curved away from the center: A rectangle centered in the image looks like a barrel. This effect is significantly less severe with more recent hardware.

- *Low contrast.* The contrast between dark and light parts of an image is very low. The brightness of the image is not constant, but the center of the image is brighter than the corner regions.

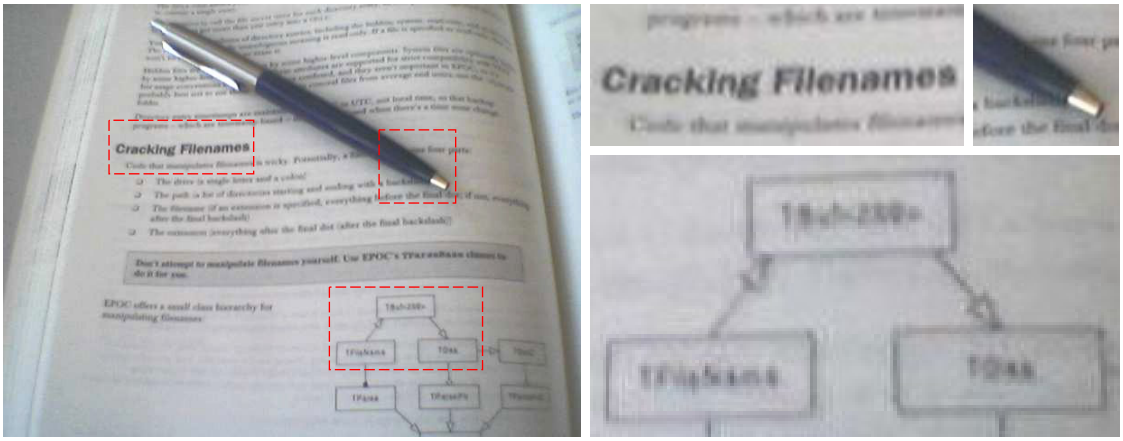


Figure 3.2: Blurred image details in a photo taken with the Nokia 7650 camera phone.

- *Blurred images.* Figure 3.2 shows the image of a book page together with enlargements of some details, like a pen, a title, and a drawing. The enlargements show that the object shapes are blurred and fuzzy.

- *JPEG compression artifacts.* The enlargements in Figure 3.2 also show substantial JPEG compression artifacts. These artifacts can be reduced a little bit by decreasing the compression rate, which comes with the penalty of an increased storage size of the image on the device. This is only an issue if

images are transferred to a backend server as a JPEG image for recognition. The native formats are not JPEG and consequently do not show JPEG compression artifacts.

Because of the mobility inherent to camera phones, scanned codes might appear at any orientation in the camera image. They can be arbitrarily rotated and tilted, which complicates image recognition. We decided to constructively use these characteristics by measuring the amount of tilting and rotation of a code tag in the image and use them as additional input parameters. Another feature we deemed essential is the ability to map arbitrary points in the image plane to corresponding points in the code plane, i.e. to compute the code coordinates of arbitrary image pixels, and vice versa. In particular, this enables the conversion of the pixel coordinates of the camera focus – which is the point the user aims at – into corresponding code coordinates and the selection of image elements located at these code coordinates. This coordinate mapping can also be used for removing the perspective distortion of image parts.

These characteristics mark out the design space for the visual codes and form the basis for the further discussion. The final code layout, which was designed according to the analysis above, is pictured in Figure 3.3. It consists of the following elements: a larger and a smaller guide bar for determining the location and orientation of the code, three cornerstones for detecting the distortion, and the data area with the actual code bits. The combination of larger and smaller guide bars is beneficial for detecting even strongly tilted codes. In the right half of Figure 3.3 the code coordinate system is shown. Each code defines its own local coordinate system with its origin at the upper left edge of the code and one unit corresponding to a single code bit element. Depending on the code size, the mapping between points in the image plane and points in the code plane is more precise than a single coordinate unit. The x-axis extends in horizontal direction to the left and to the right beyond the code itself. Correspondingly, the y-axis extends in vertical direction beyond the top and bottom edges of the code. For each code found in a particular input image, the code recognition algorithm establishes a bijective mapping between arbitrary points in the code plane and corresponding points in the image plane. Figure 3.4 shows the code coordinate system of a visual code that is captured from a tilted and rotated perspective.

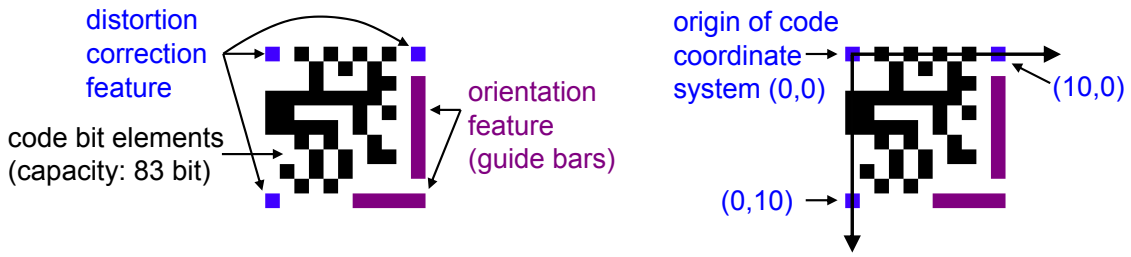


Figure 3.3: The layout of the visual code (left) and the code coordinate system (right).

The recognition algorithm requires one bit cell of white space around a visual code. Multiple codes can be laid out in a grid and positioned closely together, as long as one bit cell of white space is left between corner stones and guide bars of neighboring codes. The grid arrangement is depicted in Figure 3.5.

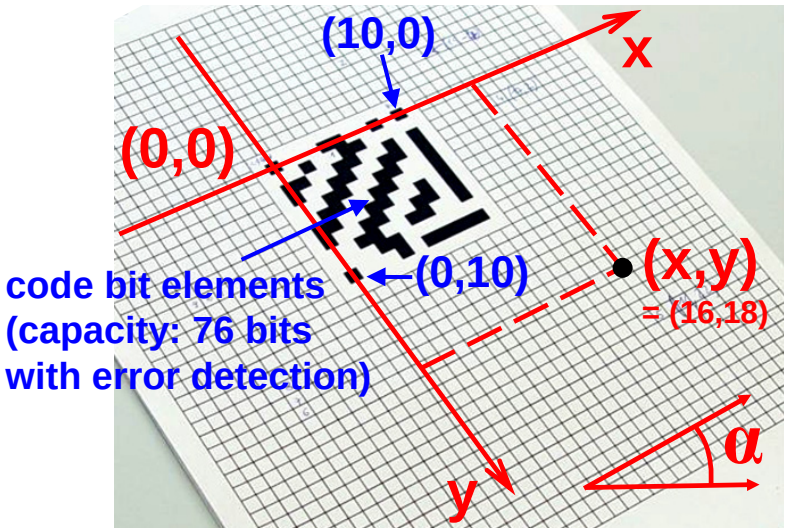


Figure 3.4: Code coordinate system of a tilted and rotated visual code.

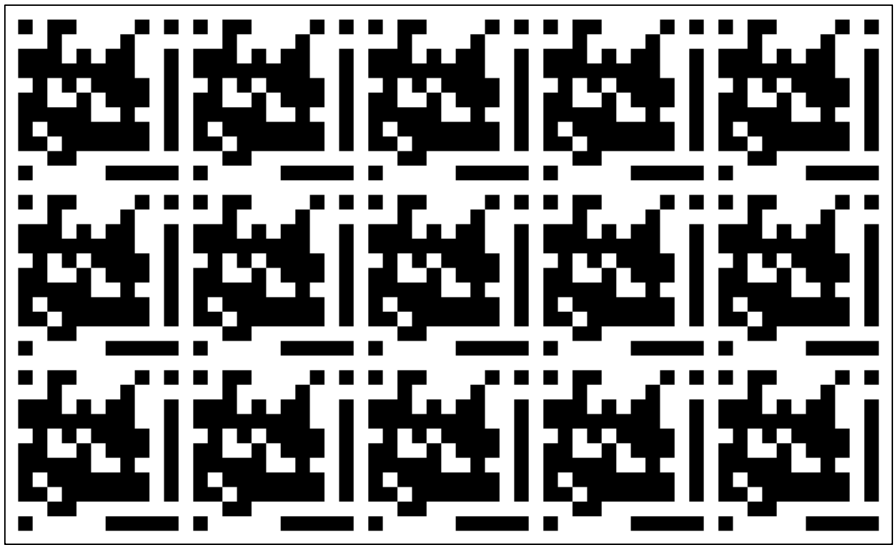


Figure 3.5: Multiple visual codes laid out in a grid. One bit cell of white space is required between neighboring codes.

3.3.2 Recognition Algorithm

The recognition algorithm performs the following main steps on the camera image and produces a code information object for each detected code.

- **Input:** Camera image
- **Output:** Set of code information objects, comprising
  - the code value,
  - the image pixel coordinates of the corner stones and guide bars,
  - the rotation angle of the code in the image,
  - the amount of horizontal and vertical tilting,
  - the distance of the camera to the code,

- a projective warper object for the code, which implements a planar homography (see below) used to transform image coordinates to code coordinates and vice versa,
- the width and height of the originating image,
- a flag indicating the result of error checking.

Gray scaling and adaptive thresholding.

To produce a gray scaled version of the colored input image, we use the formula $gray = (red + green)/2$, instead of the ITU-standardized formula for luminance $Y = 0.2126 \times red + 0.7152 \times green + 0.0722 \times blue$. The blue color component is omitted, since it has the lowest quality in terms of sharpness and contrast. Our simple formula is computationally efficient and produces an adequate starting point for thresholding. Efficiency in this step is of utmost importance for the performance of the whole recognition algorithm, because every single image pixel has to be gray scaled.

An adaptive thresholding method is taken to produce a black-and-white version of the gray scaled image, because the brightness of the camera image is not constant and the visual code may be unevenly illuminated. We slightly modified the adaptive thresholding algorithm described in [221], where the basic idea is to use a weighted moving average of the gray values while running through the image in a snake-like fashion (alternating left to right and right to left scanline traversal). Our adaptation takes the previous scanline of each examined scanline into account in order to avoid artifacts in every other line, resulting from the zigzag traversal of the scanlines. The average $g_s(n)$ is updated according to

$$g_s(n) = g_s(n-1) \cdot (1 - \frac{1}{s}) + p_n$$

with p_n denoting the gray value of the current pixel and $s = \frac{1}{8}w$ the width of the moving average (w is the image width). g_s is initialized with $g_s(0) = \frac{1}{2}cs$, where c is the maximum possible gray value. The color of the thresholded pixel $T(n)$ is then chosen as ($t = 15$):

$$T(n) = \begin{cases} 1 \text{ if } p_n < \frac{g_s(n)}{s} \cdot \frac{100-t}{100} \\ 0 \text{ otherwise} \end{cases}$$

Gray scaling and adaptive thresholding turned out to be the most time consuming phase of the recognition process. Therefore, we replaced any floating point operations in this part by shifted integer operations, which resulted in a significant performance improvement.

Region identification and labeling.

This step consists of finding regions of neighboring black pixels, counting them, and assigning a number to each. The algorithm we use is a well known two-phase method: In the first phase, the image is traversed row by row, assigning preliminary labels to the regions found. During this process, it may happen that two regions with different labels turn out to be in fact the same region. In this case, the equivalence of the two temporary labels is stored in a table. The second phase

resolves the equivalencies by merging the corresponding regions and assigns a final label to each region.

In the implementation, gray scaling, adaptive thresholding, and the first phase of region labeling are bundled for performance reasons and are done in a single scan through the image, i.e., pixels that are thresholded as foreground pixels are immediately assigned a label and any label equivalences are recorded.

Calculation of region shapes and orientations.

In order to identify candidates for orientation bars among the regions found, the notion of second-order moments is used [205]. From these moments, the major and minor axis of each region is determined. The ratio of the lengths of these axes is a good measure for the "eccentricity" of a region: perfect circles and squares have a ratio equal to one whereas line segments have a ratio close to zero. This is very useful to identify regions with a bar-like shape.

The second-order moments of a region consisting of the set of pixels R and having the center of gravity $(\bar{x}, \bar{y})$ are defined as follows:

$$\mu_{xx} = \frac{1}{|R|} \sum_{(x,y)\in R} (x - \bar{x})^2,$$

$$\mu_{yy} = \frac{1}{|R|} \sum_{(x,y)\in R} (y - \bar{y})^2,$$

$$\mu_{xy} = \frac{1}{|R|} \sum_{(x,y)\in R} (x - \bar{x})(y - \bar{y}),$$

$$\text{where} \quad \bar{x} = \frac{1}{|R|} \sum_{(x,y)\in R} x, \qquad \bar{y} = \frac{1}{|R|} \sum_{(x,y)\in R} y$$

From these moments, an ellipsis $E = \{(x, y)|dx^2 + 2exy + fy^2 \leq 1\}$ that has the same major and minor axis as the region can be defined by setting:

$$\begin{pmatrix} d & e \\ e & f \end{pmatrix} = \frac{1}{4\mu_{xx}\mu_{yy} - \mu_{xy}^2} \begin{pmatrix} \mu_{yy} & -\mu_{xy} \\ -\mu_{xy} & \mu_{xx} \end{pmatrix}$$

Furthermore, the orientation vector of the major axis is calculated as

$$\begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix}, \quad \text{where} \quad \alpha = \frac{1}{2} \arctan \frac{2e}{d - f}.$$

Locating and evaluating the codes.

Locating codes in the image is done by looking for guide bar candidates and by finding corresponding cornerstones. Guide bar candidates are found by simply selecting those regions whose axis ratio lies in a certain range around the expected ideal axis ratio. The range has to be large enough to allow for tilted codes. For each of these candidates, the size and orientation of the region is used to estimate the expected positions of the second guide bar and the three cornerstones. It is then checked whether these features are actually present at the estimated positions. Cornerstone candidates found are only accepted if their axis ratio is above a certain limit.

Computing the projective mapping from code coordinates to image coordinates.

Since the code elements are coplanar, there exists a unique homography (projective transformation matrix) between the code plane and the image plane. The projective mapping can be calculated once four corresponding points are known [95]. In our algorithm, the correspondences are the centers of the three cornerstones plus the center of the second guide bar:

Code element	Image coordinates	Code coordinates
upper left cornerstone	(x_0, y_0)	$(0, 0)$
upper right cornerstone	(x_1, y_1)	$(10, 0)$
second guide bar	(x_2, y_2)	$(8, 10)$
lower left cornerstone	(x_3, y_3)	$(0, 10)$

Code coordinates (u, v) are mapped to image coordinates (x, y) with

$$x = \frac{au + bv + 10c}{gu + hv + 10}, \quad y = \frac{du + ev + 10f}{gu + hv + 10}.$$

The parameters a to h are calculated from the four reference points (x_i, y_i), $i \in \{0, \ldots, 3\}$, as follows:

$$\Delta x_1 = x_1 - x_2 \qquad \Delta y_1 = y_1 - y_2 \qquad \Delta x_2 = x_3 - x_2 \qquad \Delta y_2 = y_3 - y_2$$

$$\Sigma x = 0.8x_0 - 0.8x_1 + x_2 - x_3 \qquad \Sigma y = 0.8y_0 - 0.8y_1 + y_2 - y_3$$

$$\begin{array}{rclcrclcrcl} g & = & \dfrac{\Sigma x \Delta y_2 - \Sigma y \Delta x_2}{\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2} & & a & = & x_1 - x_0 + g x_1 & & d & = & y_1 - y_0 + g y_1 \\ h & = & \dfrac{\Sigma y \Delta x_1 - \Sigma x \Delta y_1}{\Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2} & & b & = & x_3 - x_0 + h x_3 & & e & = & y_3 - y_0 + h y_3 \\ & & & & c & = & x_0 & & f & = & y_0 \end{array}$$

Computing the projective mapping from image coordinates to code coordinates.

The inverse mapping to the one described above is important for applications which select items visible in the image. Given the coordinates of a pixel, the corresponding code coordinates can thus be obtained. Image coordinates (x, y) are mapped to code coordinates (u, v) as follows:

$$u = 10 \cdot \frac{Ax + By + C}{Gx + Hy + I}, \quad v = 10 \cdot \frac{Dx + Ey + F}{Gx + Hy + I}, \quad with$$

$$\begin{array}{rclcrclcrcl} A & = & e - fh & & D & = & fg - d & & G & = & dh - eg \\ B & = & ch - b & & E & = & a - cg & & H & = & bg - ah \\ C & = & bf - ce & & F & = & cd - af & & I & = & ae - bd \end{array}$$

Rotation angle.

The rotation angle gives the rotation of the visual code in the image in degrees counterclockwise (0-359°). A code that has the same orientation as the image has rotation angle 0° (like the ones in Figure 3.3). The rotation is determined by mapping the points (0,0) and (100,0) from the code coordinate system to the image coordinate system, resulting in the image points (a_x, a_y), and (b_x, b_y). The rotation angle is then determined as the arc tangent of the difference quotient of a and b.

Horizontal and vertical tilting.

The term *tilting* denotes the amount of inclination of the image plane relative to the code plane. *Horizontal tilting* is the amount of inclination of the image plane relative to the horizontal axis of the code. Analogously, *vertical tilting* denotes the amount of inclination of the image plane relative to the vertical axis of the code. Tilting values of 1 mean no tilting, a value less than 1 means tilting towards the left/top, and a value greater than 1 means tilting towards the right/bottom.

The tilting parameters are computed as follows: Four image points with constant distance h (image height) from the image center point in the axis directions of the code coordinate system are computed. They are mapped to corresponding code coordinates and their distances to the center point are computed. The ratios of these distances determine the tilting parameters t_x and t_y. They are independent of the size of the code in the image. From these ratios the tilting angles t_x^α and t_y^α can be determined, if a constant r is known that depends on the camera parameters. It can be obtained experimentally. For the Nokia 6600, e.g., this parameter has the value $r = 1.64177$.

$$
\begin{aligned}
i &= \text{image coordinates of the image center point} \\
c &= \text{CodeCoordinates}(i) \\
\\
x &= \text{ImageCoordinates}(c + (1,0)) - i \\
y &= \text{ImageCoordinates}(c + (0,1)) - i \\
u &= x/|x| \\
v &= y/|y| \\
\\
l &= |\text{CodeCoordinates}(i - hu) - c| \\
r &= |\text{CodeCoordinates}(i + hu) - c| \\
t &= |\text{CodeCoordinates}(i - hv) - c| \\
b &= |\text{CodeCoordinates}(i + hv) - c| \\
\\
t_x &= l/r \\
t_y &= t/b \\
\\
t_x^\alpha &= \arctan\left(r\frac{t_x - 1}{t_x + 1}\right) \\
t_y^\alpha &= \arctan\left(r\frac{t_y - 1}{t_y + 1}\right)
\end{aligned}
$$

Code distance.

If the real code size s_{real} (the real distance between the centers of the upper left and the upper right cornerstones) and the camera's focal distance f are known, the metric distance from the camera to the untilted visual code can be computed from s_{image} (the pixel distance between the centers of the upper cornerstones in the camera image) using the pinhole model as (w_{image} is the pixel width of the image)

$$D_{camera,code} = \frac{s_{real} \times f}{s_{image}/w_{image}}.$$

Since s_{real} and f are typically not known and we want to use the code distance for interaction purposes rather than measuring its exact value, we define the distance in terms of the size of the visual code in the image. We set $d_{camera,code} := 100$ for the farthest distance at which a code is recognized in view finder mode. For the Nokia 6600 this is the case when $s_{image} = 25$ pixels, which amounts to 15.625% of the image width. Hence the distance is computed as

$$d_{camera,code} = \frac{15.625}{s_{image}/w_{image}}.$$

Should s_{real} and f be known, the metric distance $D_{camera,code}$ can still be computed from $d_{camera,code}$ as

$$D_{camera,code} = d_{camera,code} \times \frac{s_{real} \times f}{15.625 \times w_{image}}.$$

For the Nokia 6600, the range of distances at which codes are recognized in view finder mode are: $11-46$ cm for $s_{real} = 69$ mm, $3.5-14$ cm for $s_{real} = 21$ mm, $2.3-9$ cm for $s_{real} = 13.6$ mm.

Reading the encoded bits.

Once the positions of the guide bars and corner stones have been identified and a suitable projective mapping has been computed, reading the encoded bits is simply a matter of testing the appropriate pixels (x, y) of the black-and-white image. There are two encodings defined within the general layout provided by the larger and the smaller guide bar and the three corner stones. The small version has a raw capacity of 83 bits. The large version stores $83 \times 4 = 332$ bits. The large version is obtained from the small version by subdividing each quadratic bit cell into four quadrats. For the small version, we use code coordinates (u, v) with $u, v \in \{0, ..., 10\}$. For the large version, the encoded bits are located at code coordinate positions $(\frac{1}{2}u-\frac{1}{4}, \frac{1}{2}v-\frac{1}{4})$ with $u, v \in \{0, ..., 21\}$. In both cases, these code coordinates correspond to image coordinates $(x, y) = \text{ImageCoordinates}((u, v))$.

Figure 3.6 shows the positions within the marker at which the individual bits of a code word are stored. The highest index corresponds to the most significant bit. The left part shows the encoding order in a small code. The right part shows the encoding order in a large code.

Error detection.

In order to detect pixel errors and false orientation features, the data bits are protected by error detecting codes. For real time marker recognition it is important

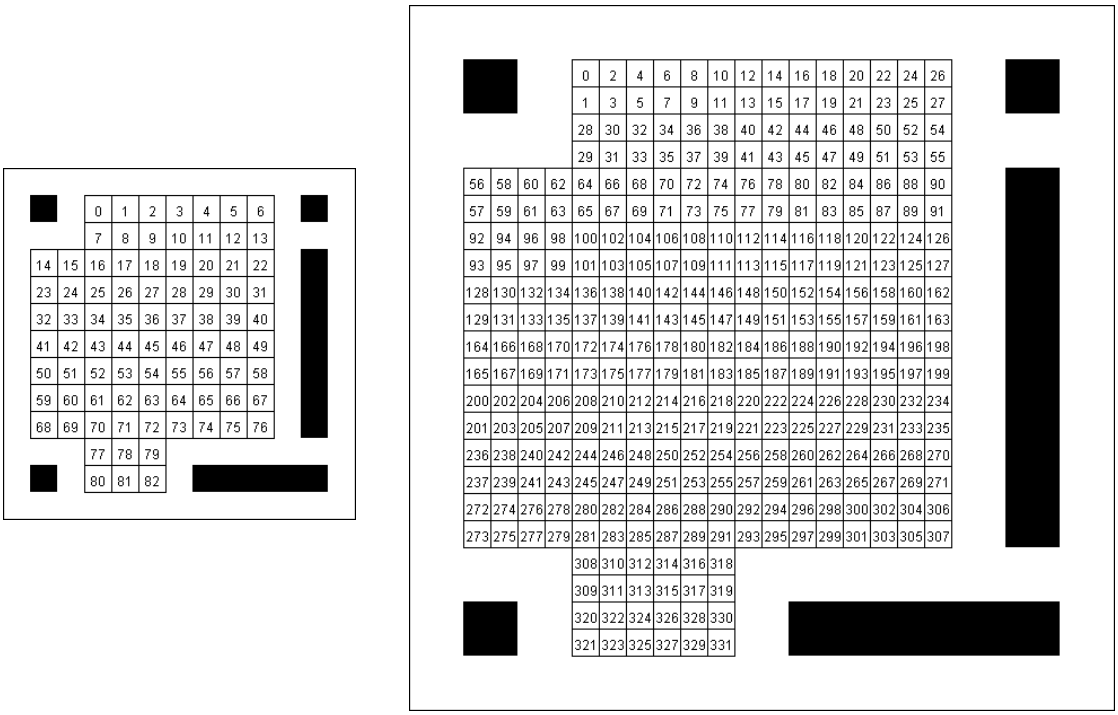


Figure 3.6: Encoding order in small (left) and large (right) visual codes. The numbers indicate the bit index within the encoded bit string. The least significant bit has index 0.

to choose error detecting codes that can quickly be decoded. The limited size of the markers requires a low space overhead for the redundancy bits. In general, there is a tradeoff between the error detection and correction capabilities of a code and the overhead in terms of storage space and computational complexity.

For the small version of the marker we chose a simple (83,76,3) linear code that generates an 83-bit code word x from a 76-bit value m and that has a Hamming distance of 3. The code word is computed as

$$x = mG$$

with code word $x[1 \times 83]$, message $m[1 \times 76]$, and generator matrix $G[76 \times 83]$. All matrixes have elements of $\mathcal{Z}_2$ and the operations are executed in the field of integers modulo 2 (logical *and* and addition modulo 2).

$$G = (I_{76}|A)$$

with identity matrix $I_{76}[76 \times 76]$ and $A[76 \times 7]$. The rows of A are taken from the sequence [3, 5, 6, 7, 9, ..., 83] (omitting the integers 2^i, $i \in \{0, 1, ..., 6\}$).

The decoder is implemented as class `CCodeChecker` of the visual code system. It allows to correct a single bit error in an 83-bit code word x using a parity check matrix H in the following way:

$$Hx^T = s^T$$

with code word $x^T[83 \times 1]$, syndrome $s^T[7 \times 1]$, and parity check matrix $H[7 \times 83]$.

$$H = (A^T|I_7) =$$

$$\left(\begin{array}{l}
00000000000000000000000000000000000000000000000000000000000011111111111111111\ 1000000\\
00000000000000000000000000111111111111111111111111111111111000000000000000000\ 0100000\\
00000000000111111111111111100000000000000011111111111111111000000000000000111\ 0010000\\
00001111111000000011111111000000011111111000000001111111100000001111111100000\ 0001000\\
01110001111000111100011110001111000011110000111100001111000011110000111100000\ 0000100\\
10110110011011001100110011011001100110011001100110011001100110110011001100110011\ 0000010\\
11011010101101010101010101101010101010101010101010101010101101010101010101010101\ 0000001
\end{array}\right)$$

with identity matrix $I_7[7 \times 7]$ and $A^T[7 \times 76]$. If there is no error, syndrome $s = (0000000)$. If s indicates an error, the corresponding code is discarded. No error correction is attempted in this case.

The large visual markers are protected by a shortened (63,55) Reed-Solomon code [160] with a symbol size of 6 bits. The chosen Reed-Solomon code consists of 55 6-bit data symbols plus 8 6-bit parity symbols. The large version of the marker has a raw capacity of 332 bits. The parity symbols require $8 \times 6 = 48$ bits. This leaves $332 - 8 \times 6 = 284$ bits $= 71$ hexadecimal digits $= 35.5$ bytes for data. The code can correct errors in up to four symbols. The implementation (class `CReedSolomon` of the visual code system) is based on Phil Karn's Reed-Solomon codec.[5]

3.3.3 Phone Movement Detection

The code recognition algorithm is combined with a phone movement detection algorithm that solely relies on image data obtained from the camera. It does not require any additional hardware components, such as accelerometers. It is integrated with the visual code recognition algorithm in such a way that the latter only examines images for visual codes when the detected phone movement is below a certain threshold. If the phone is quickly moved, it is very unlikely that the user aims at a specific code. Trying to locate codes in the image in such a case would not be sensible.

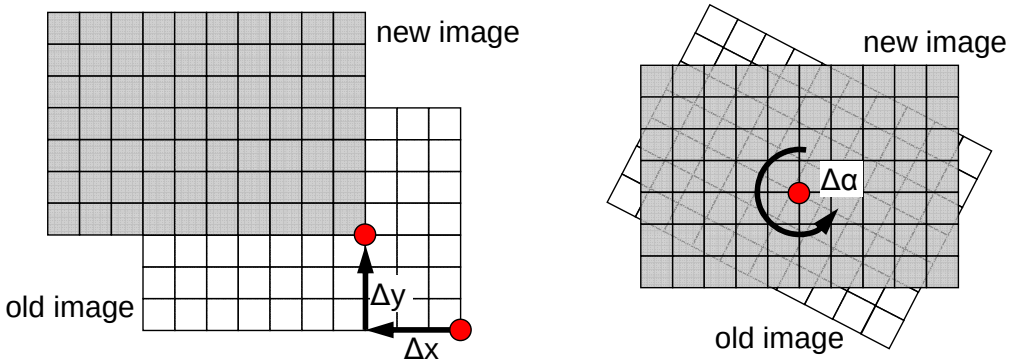


Figure 3.7: Checking different displacements in two successive block images to compute the most likely movement. Translational (Δx, Δy) and rotational ($\Delta\alpha$) displacements are checked.

The algorithm provides the relative x, y, and *rotational* motion of the phone, representing three degrees of freedom (see Figure 3.7). With the movement detection, the camera phone thus becomes an optical mouse. The algorithm works as follows: Successive images from the camera are dispatched to the view finder to

[5]Phil Karn: Reed-Solomon codec. Available under GPL at: `www.ka9q.net/code/fec`

render them on the device display. Every n-th frame (depending on the performance of the phone) is used for phone movement detection. The image is divided in 16×16 pixel blocks. For each block, 16 pixels are sampled (out of the 256 available pixels in each block) and their average gray value is computed. Then, the blocks of the current image are compared to the blocks of the previously sampled frame. The block arrays are displaced against each other in x and y direction using values for Δx and Δy from $\{-3, \ldots, 3\}$. The difference values are computed and normalized with the number of compared blocks (which depends on the amount of displacement) and the minimal difference is chosen as the most likely amount of $(\Delta x, \Delta y)$ movement relative to the image before. Relative rotation θ is computed in a similar fashion, but rotating the block images against each other. The current block image is rotated by $\Delta\alpha$ values between $-24°$ and $24°$, with a step width of $6°$. The rotational coordinate mappings are precomputed and stored in tables for performance reasons. Again, the differences of the resulting block images are compared to the previous block image and the minimal difference is chosen as the most likely amount of relative rotation.

This simple algorithm works quite reliably and detects the relative motion even if the sampled backgrounds only have a limited number of features, like a wall or a floor. Because only a few pixels are sampled, the algorithm performs quickly and leaves enough time for doing the actual code recognition. On a Nokia 6600, it produces about five (x, y, θ) triples per second.

3.4 Implementation and Performance

The code recognition and motion detection algorithms have been implemented for various platforms. There are C++ implementations for Symbian OS (v6.1, v7.0s, and v8.0a) and Windows Pocket PC Phone Edition [13]. There is a Java Micro Edition (J2ME) implementation for mobile phones [85] and a Java Standard Edition (J2SE) implementation for PCs. The J2ME implementation requires MIDP 2.0 and the J2ME camera API (JSR 135). The J2SE implementation requires the Java Media Framework (JMF) or Windows DirectShow and operates with a connected camera.

The recognition algorithm was first developed on the J2SE platform on a PC using images that were originally captured by a camera phone. This Java implementation is now available for infrastructure-supported visual code recognition. It has been used in a number of projects and diploma theses (such as the stationary recognition of product packages that is described in Chapter 8). Infrastructure-supported recognition opens up visual code recognition for non-programmable camera phones.

On the Symbian OS platform, we experienced the following performance figures. Replacing floating point operations by shifted integer operations reduced the time consumption of the thresholding phase from 2000 ms to less than 400 ms on a Nokia 7650 for a 640×480 pixel camera image. The total execution time of the recognition algorithm on the same device amounts to about 700 ms if less than 5 codes are present, and up to 1500 ms if 30 codes are present (see Figure 3.9) – which is rather uncommon in typical applications. The picture-taking process for 640×480 pixel images takes about 850 ms, resulting in an overall average delay of about 2000 ms. Low resolution 160×120 pixel images that are generated during the view finding process are recognized much faster, i.e. in close real-time as the

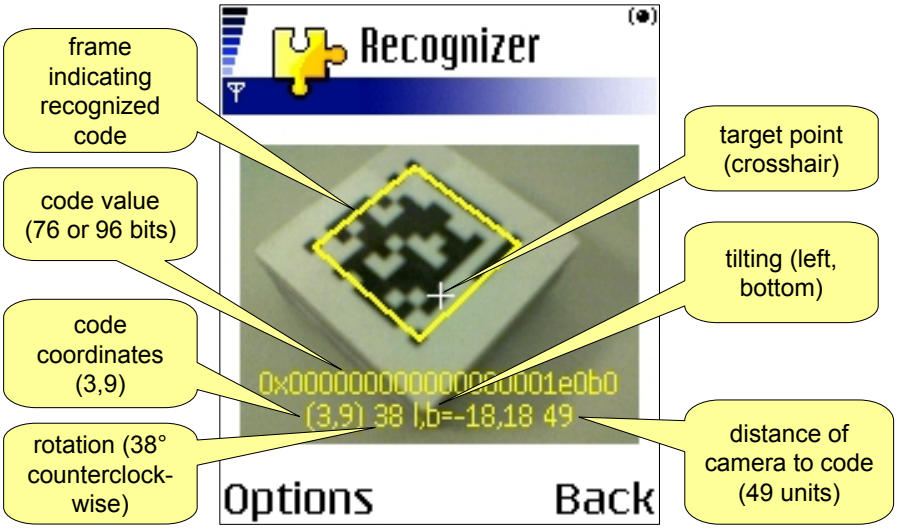


Figure 3.8: Visual code parameters as shown in the Symbian recognizer.

device moves relative to the detected code(s). Figure 3.8 shows a screenshot of the Symbian recognizer application with the parameters of a recognized code.

Figure 3.9: Multiple recognized code: (left) all codes have been correctly recognized (yellow frames), (right) some codes have bit errors (red frames).

The version based on Windows Pocket PC Phone Edition performed slightly better [13], which is mostly due to the higher performance processor of the MDA III we used. The performance of the recognition algorithm is more dependent on the performance of the main processor and the availability of a floating point unit than on features of the operating system. The J2ME implementation [85] showed significantly lower performance. Recognition of a single code on a Nokia 7610 took about 500 ms for a QQVGA image (160 × 120 pixels). The Nokia 7610 implements CLDC 1.0, which does not include floating point numbers. All floating point numbers had to be emulated in Java with a special class `Real` by

Roar Lauritzsen.[6] The additional 800 ms for taking a QQVGA image in J2ME ruled out the implementation of a view finder mode. CLDC 1.1 comprises the Java types `float` and `double`, which should result in a significant performance increase.

Visual codes are reliably detected even if they are tilted by 45°-60° from the central view. As Figure 3.9 shows, multiple codes are detected in a single image. For this, the time-consuming thresholding and labeling steps have to be performed only once. In a VGA image (640 × 480 pixels), up to 70 codes can be recognized.

The reading distances for a small code of 15 × 15 mm, which was printed on recycled paper and detected by a Nokia 6600, are as follows:

- View finder images (QQVGA, 160 × 120 pixels): 3 to 9 cm
- Still images (VGA, 640 × 480 pixels): 3 to 28 cm

With a larger code of 40×40 mm, printed on white paper, the maximum reading distance increased to 80 cm. Larger codes, as could be placed on posters, can be read from a distance of multiple meters. All tests have been performed without any macro lenses. With macro lenses, the printed size of the codes can be much smaller.

The codes are detectable in normal daylight, as well as by artificial light. They are suited for printing on white paper, on recycled paper, for display on electronic screens, as well as for projection with LCD projectors. With projectors, the recognition rate can be diminished by direct sunlight or very bright artificial light.

Since the recognition algorithm is executed on the phone, no infrastructure support for the actual code recognition is required. In order to retrieve actual content, wireless connectivity via GSM/GPRS or Bluetooth is necessary. We set up a Web server with a PHP script to map code values and parameters to content. The visual code recognizer application returns resolved content as HTML pages with links to embedded images.

3.5 Item Selection using Visual Code Parameters

In this section we show how the orientation parameters that the code recognition algorithm provides can be used to realize various applications that include novel interaction patterns.

For testing the implementation we developed the *visual code recognizer*, which is shown in Figure 3.8. It is a basic demo application[7] that allows to take images, recognize them, and optionally send the identifiers and parameters to a backend server. The backend server resolves the supplied identifiers, parameters, and the user identifier to actual content. As backend servers, we have used simple PHP scripts as well as the ETHOC system [173]. When aiming the phone camera at the target item, the image of this target item appears on the display. It is continuously updated as the phone is moved. The center of the display contains a crosshair to facilitate precise selection, as can be seen in Figure 3.8. To further facilitate item selection the display contains a magnified portion of the area around the display center. The level of magnification can be adjusted with the joystick.

[6] `gridbug.ods.org/Real.html`

[7] This and other applications as well as example codes are available at `www.inf.ethz.ch/personal/rohs/visualcodes`.

Figure 3.10: Cut-out of a cinema magazine with embedded visual codes.

A primary application area of the visual code system is adding online content and operations to printed documents, like flyers, magazines, etc. An example online operation is selling tickets for events, theaters, or movies via the mobile phone. Figure 3.10 shows a page of a printed television magazine, which is equipped with embedded visual codes. By aiming at the printed visual codes, movie plot outlines can be shown, a list of the cast can be given, movie ratings can be displayed or submitted, movie trailers can be started, and tickets can be directly booked via the mobile phone.

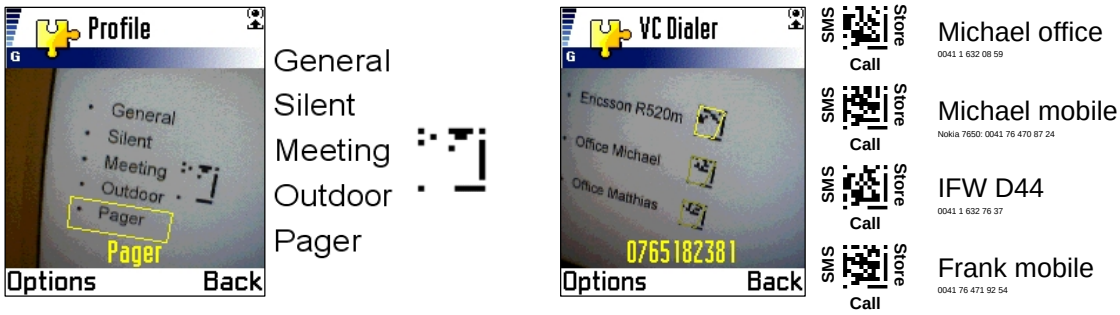


Figure 3.11: Application screenshots: Profile selector and Dialer.

To simplify interaction with the mobile phone itself, parts of its user interface can be externalized on paper or on a large screen by using visual codes. Deeply nested menu hierarchies – that are a consequence of the tiny display dimensions – are difficult to deal with. Such menu hierarchies can be unfolded, laid out on paper, and recognized by embedding associated visual codes. The profile selector (see left part of Figure 3.11) application illustrates this scenario. It allows the selection of the current phone profile ("outdoor," "meeting," "pager," etc.) by aiming the crosshair at the desired item. Note that only a single code is necessary in this application. The selected item is identified by using the projective mapping described above. It maps the coordinates of the crosshair at the image center to the code coordinate system, which is independent of the camera orientation.

The mapping from image coordinate system to code coordinate system enables the precise selection of items in the image, requiring just a single code element for

multiple items. Image items may be menu entries, arbitrarily shaped regions in a picture, or the cells of a table. Further input parameters comprise the rotation of the code tag in the image and the amount of tilting of the image plane relative to the code plane. The tilting parameter identifies the viewing position (from left, from right, from top, from bottom). Both parameters can be used to associate multiple information aspects with a single point in the code coordinate system. As an example, Figure 3.12 shows the selection of multiple items in a table that is associated with a single visual code. The code coordinates determine the table row, the camera rotation specifies the concrete information aspect to display.

Figure 3.12: Selection from a table: the code coordinates determine the table row, the camera rotation specifies the concrete information aspect to display.

For an effective interaction, the user has to be provided with indications about the possible interactions. This can be achieved by superimposing visual cues on the display image that indicate at what rotation angles and viewing positions what kind of information is to be expected. Chapter 4 details different kinds of symbols that guide the user in his or her interactions with visual codes; we present a conceptual framework for the combination of such visual cues. An indication of user interaction normally consists of a symbol denoting the kinds of physical interaction – like movement, rotation, or tilting – and a set of symbols for the associated actions that are triggered as a consequence of the interaction. The latter comprise symbols for typical functions of a mobile phone, such as placing a phone call or starting the WAP browser. Another possibility is to print interaction cues next to the code. This was realized with a visual code dialer application (see right part of Figure 3.11). The printed code contains a phone number and is surrounded by words indicating the function that is triggered when the phone is tilted in that direction: Just below the code it says "Call," to the left it says "SMS," and to the right the word "Store" is printed. Scanning from a central position immediately places a call, scanning from the left opens the phone's SMS editor with the number already entered into the appropriate field, and scanning from the right looks up the contact information on a server and stores it on the phone.

In newspapers, online background information to articles, advertisements, or information which quickly gets obsolete, like weather forecasts or stock quotes, can be linked via visual codes. Figure 3.13 shows a cut-out of a newspaper page containing a geographic map with the current weather data and a table containing the snow conditions for various regions. The dotted lines drawn on the newspaper

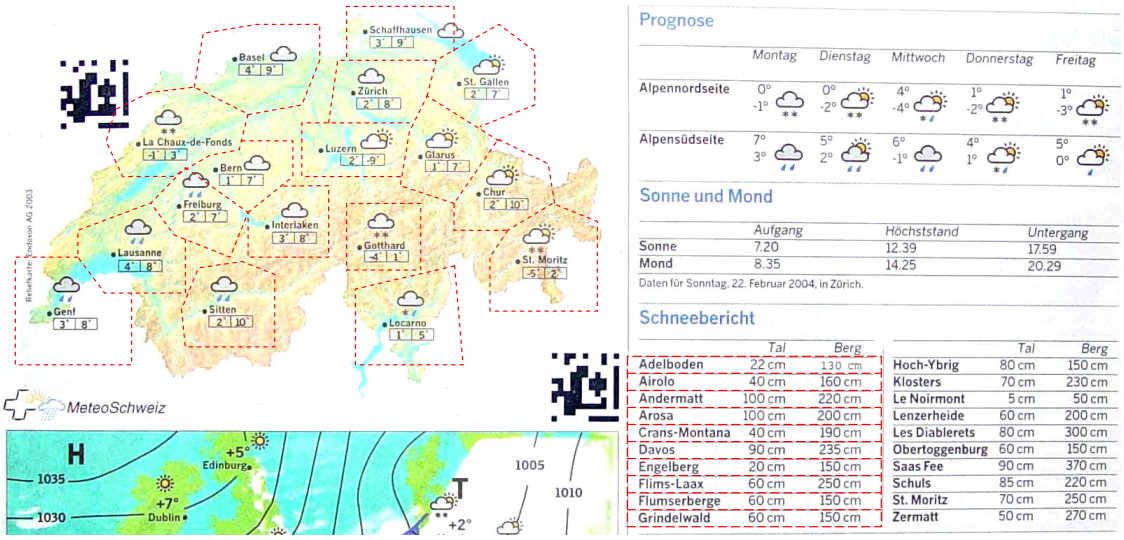


Figure 3.13: Example of a weather forecast newspaper page containing visual codes. The 17 regions on the map and all entries in the table are individually mapped to different URLs and thus hyperlinked to specific online content.

page indicate sensitive areas that are individually linked to online content. Such a mapping can be easily created with suitable content creation software. As a prototype, we developed a mapping tool for drawing the areas in the image and specifying the associated URL for each region. The tool computes the coordinates of these areas in the coordinate systems of the codes present in the image, and stores this data as an XML file. Multiple URLs can be specified for each region by taking into account rotation, distance, and tilting. As shown in Figure 3.13, a single code suffices to select any one of the multiple areas and table entries, respectively. By rotating the mobile device, different aspects of the online information can be chosen: In the example, vertical orientation shows the snow depth for the selected area, while a slight rotation shows the current temperature. Other conceivable operations include showing the currently open skiing trails, calling the local tourist information office, and booking rail and lift tickets. The current weather data is retrieved from a server and the display of the phone is updated in real time as the crosshair is moved across the table entries and geographic regions and as the phone is rotated clockwise and counterclockwise. A video that demonstrates this type of interaction is available at `www.vs.inf.ethz.ch/personal/rohs/visualcodes`.

The ability to link multiple items to a single code based on their code coordinates and to associate multiple information aspects to a single point depending on rotation and tilting has a number of usability advantages. In the example above, it would of course be possible to present a table of the current snow conditions of all regions on the map to the user once the code has been recognized. But it is difficult to effectively show a table containing all the attributes on the small amount of available display space. It also requires the user to scroll through the – possibly lengthy – table and locate the data of interest in a second step. The presented approach avoids both of these problems. It gives direct and immediate feedback to the user and presents exactly the scanned item and selected information aspect.

Visual codes can also be combined with printed forms for the simplification of form input, in which the mobile device provides a wireless communication channel

for conveniently and cheaply sending back the entered data. The fields of these forms can be check boxes, rulers, and canvas fields for free-hand textual or graphical input. Using the frame surrounding the form as a marker, the individual form elements can be precisely localized. The projective mapping of the recognition algorithm allows to "unwarp" parts of the image as well as the image as a whole. Figure 3.14 shows an example form for the entry of calendar events. The form processing algorithm first recognizes the code and locates the edges of the frame. Then the checkboxes located in the form are scanned for check marks. Finally, the canvas fields are "unwarped," i.e. their tilting is removed as shown in the lower part of Figure 3.14, and stored. To define the forms, a markup language is beneficial, which describes the structure of a form, including the positions and kinds of input elements it contains. The code is first used as a key to retrieve the form description from an online server. After interpreting the retrieved markup, user input can be categorized according to field type and sent back to the server.

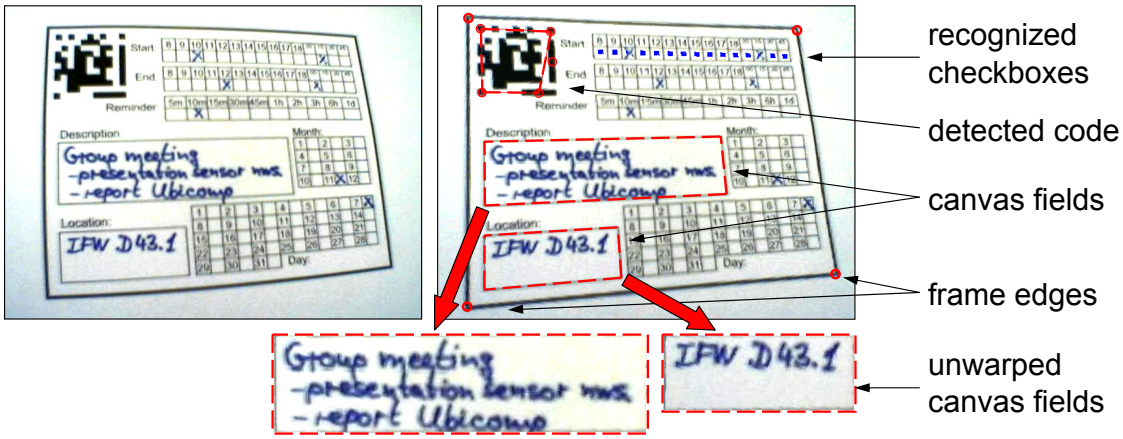


Figure 3.14: Calendar entry form: original image with barrel distortion (left), corrected image with recognized code and frame edges (right), unwarped canvas fields (bottom).

The CAM document processing framework [154] uses camera phones and our visual codes to link digital functionality to paper-based forms. The application scenario is to support people in rural India to capture information they have entered on paper forms with camera phones in order to allow for digital processing. The CAM framework consists of CamForms, CamShell, CamBrowser, and CamServer. CamForms are paper documents that contain visual codes arranged in a grid (cf. Figure 3.5). CamShell is a scripting language that specifies data capture instructions within CamForms. The scripting language is encoded within the visual codes of a CamForm. There are instructions for variable assignment, dialogs, control and arithmetic, application-launching, multimedia instructions, and networking. Arranged within a script, these instructions enable the collection of data from printed documents. CamShell scripts control the phone's dialog with the user and thereby the user's data capturing activity. Users are guided in entering the contents of paper-based forms and tables. CamBrowser is the application on the camera phone that recognizes codes and interprets forms. CamServer operates in the backend and processes uploaded data.

In the CANVIS system [143], visual codes are used for real time monitoring and visualization of wide area networks (WANs). Camera phones and visual codes

attached to networking hardware and infrastructure cabling allow field engineers to get access to real time network traffic and statistics. The rotation parameter is used to access different information aspects. Holding the phone upright retrieves information about the device or connection to which the visual code is attached, such as its IP address, management port number, zone within the WAN, historical information, and hyperlinks to existing fault tickets. Holding the handset upside down shows a graph with the current network traffic passing through that node at that point in time. Holding the phone left or right shows the upstream and downstream node statistics page, respectively. These features support field engineers in locating faults in the network.

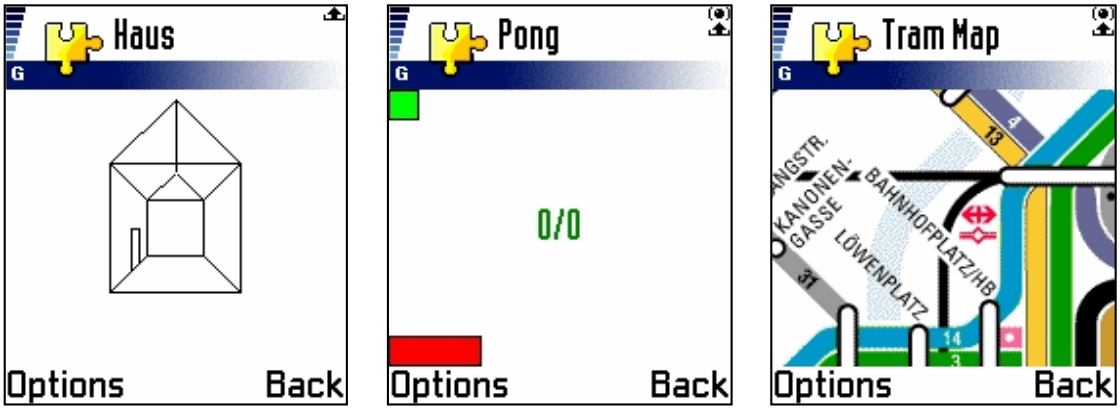


Figure 3.15: Wireframe model (left), pong game (middle), and tram map (right), all controlled by the visual detection of phone movement.

We have explored the integrated phone movement detection features in a number of ways. As the camera detects phone movement relative to the background, the content of the phone display is continuously updated. No visual code needs to be present in the view of the camera. With this technique we have built a camera-controlled wireframe model of a house, a pong game whose slider can be controlled by tilting the wrist left and right, and an application showing a large tram map that is scrolled in response to phone movement. These applications are shown in Figure 3.15. In Chapter 6, we describe techniques for interacting with large-scale displays that are based on the phone movement detection algorithm.

3.6 Visual Codes Sequences

The data capacity of a visual code is fixed to a relatively small number of bits. However, if visual codes are shown on electronic screens, such as large-scale public displays or on mobile devices, they need not be fixed, but can be dynamically presented in a repeating sequence of codes. The data capacity of a sequence depends on the number of codes it contains, which in turn is limited by the time the user can be expected to wait for capturing the whole sequence. *Visual code sequences* [115] provide an anonymous unidirectional communication channel between the capturing camera phone and the displays that show the sequence. Other than in Bluetooth or WLAN communication, the capturing device does not need to reveal its presence, since no device identifier needs to be announced to the sending device. The anonymity has advantages in terms of privacy. Visual code sequences

can be used to announce the privacy policy of a smart environment without a need for the receiver to reveal its identity or even its presence. Other metadata, such as the cost for service provision, can also be transferred via this channel. Finally, it can be used to associate mobile devices by bringing them close together. This would allow for dispensing with the usual Bluetooth discovery and pairing processes, for example. In their "seeing-is-believing" approach, McCune et al. [141] use our visual codes and the visual channel between two devices for intuitive authentication. The approach relies on the assumption that the visual channel is sufficiently secure for device authentication.

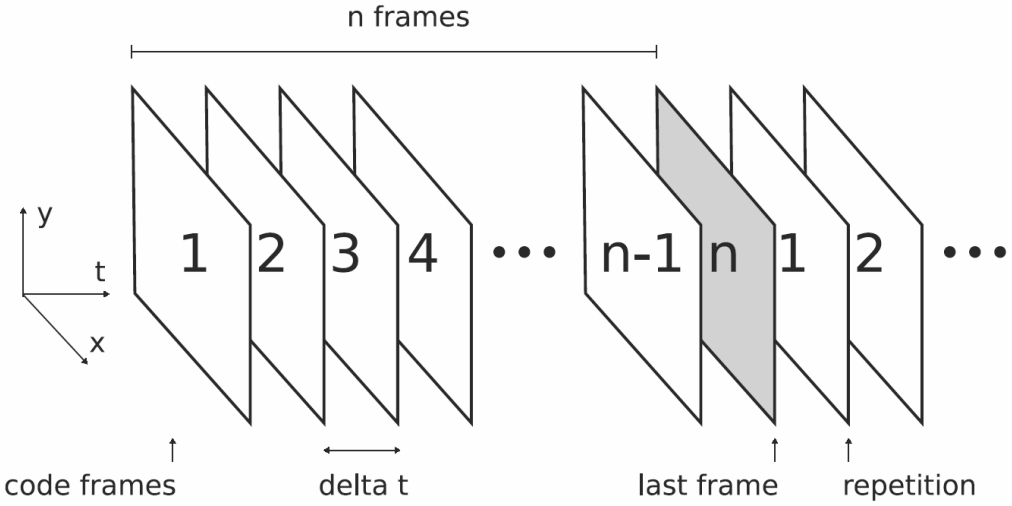


Figure 3.16: A visual code sequence of n frames (from [115]).

Figure 3.16 shows a visual code sequence of n frames. Each frame is composed of either one or four codes and shown for an interval time of Δt seconds. After the last frame has been shown, the first one is shown again. Figure 3.17 depicts the communication scheme. The sender is shown on the left side of Figure 3.17. It splits the application data, appends a frame header to each frame, encodes the raw data using a linear code, and stores the code words into a buffer. On the physical layer, each code is displayed on the screen for the interval time Δt. The receiver is shown on the right side of Figure 3.17. The receiver's camera continuously captures its input in view finder mode. Visual codes are detected, decoded, and stored in a buffer. Once the whole sequence is complete, the frame headers are stripped and the application data is reassembled.

The time users are willing to capture visual code sequences is limited. Moreover, code recognition in view finder mode is not perfect. If a code within a frame is not recognized, the whole sequence has to pass through again, until there is another chance to capture the missing code. The overall waiting time and error probability are determined by the parameters n (the size of the sequence) and Δt (the time interval each frame is shown). During performance tests we found minimal frame interval times of $\Delta t = 0.25$ s for the Nokia 6600 and $\Delta t = 0.15$ s for the Nokia 6630. Since no feedback is possible during anonymous communication, a longer interval time has to be used in order to accommodate for devices with low frame rates. Maximal sensible sequence sizes are $n = 10$ frames for one code per frame and $n = 8$ frames for four codes per frame. This results in data capacities of 80 and 270 bytes, respectively. Based on our measurements, the expected waiting time to

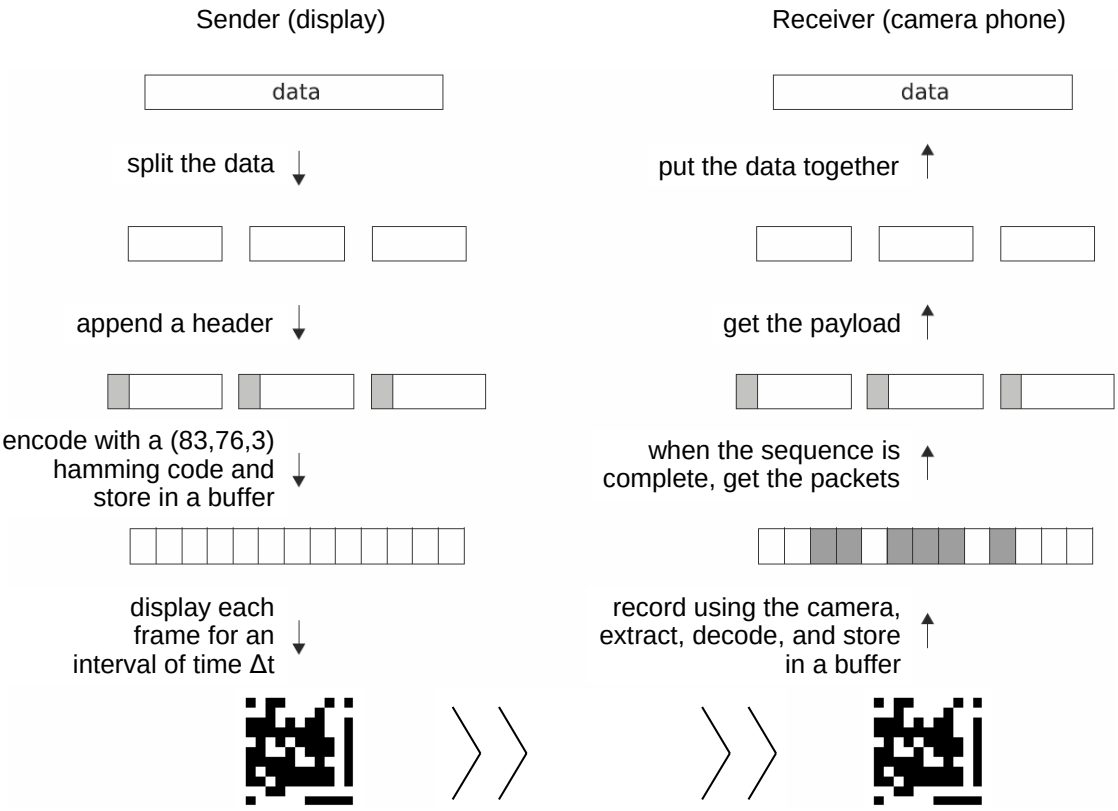


Figure 3.17: Unidirectional communication via visual code sequences (from [115]).

capture a full sequence of maximum length is about 3 s for one code per frame and 5 s for four codes per frame, respectively. These values are strongly dependent on the recognition reliability, which is lower if four codes are present rather than a single one, and on the frame rate of the visual code system on the particular device.

3.7 Summary

In this chapter we have presented extended features of a visual code system for camera equipped mobile phones. It performs well on resource constrained phone devices with low to medium resolution cameras. Besides detecting visual codes in the user's vicinity and thus providing a basis for linking physical objects to online content, it has a number of supplementary features. It provides the code coordinates, the distance between the code and the camera, the code rotation angle, and the tilting of the image plane relative to the code plane as additional input parameters. These parameters can be determined without prior calibration. Phone movement detection is integrated with the visual code system. It provides (x, y, θ) motion parameters and turns the mobile phone into an optical mouse.

We have shown how these input parameters can be used and combined to provide novel interaction patterns with objects in the user's local environment. The user can pick up multiple information items which are located at known code coordinate positions relative to a single code tag by aiming the camera focus at the appropriate location. By slightly rotating or tilting the phone, the user has the opportunity to select between different information aspects.

In the future, camera phones might play a prominent role as ubiquitous personal image recognition devices and for local interaction with physical objects that their users encounter in everyday settings. New services can be associated with printed documents, wall displays, TV programs, or general consumer products when they are made interactive using techniques as described in this chapter.

Chapter 4

A Conceptual Framework for Marker-Based Interaction

In this chapter we propose and evaluate a conceptual framework of marker-based interaction techniques for camera-equipped mobile phones. The visual code system described before forms the basis for the proposed interaction techniques. Our conceptual framework defines a set of fundamental physical gestures that form a basic vocabulary for describing interaction when using mobile phones capable of reading visual codes. These interaction primitives can be combined to create more complex and expressive interactions. A stateless interaction model allows for specifying interaction sequences, which guide the user with iconic and auditory cues. In using the parameters of the visual code system as a means of input, our framework enhances the currently limited input capabilities of mobile phones. Moreover, it enables users to interact with real-world objects in their current environment. We present an XML-based specification language for this model, a corresponding authoring tool, and a generic interpreter application for Symbian phones.

4.1 Introduction

In our conceptual framework, we propose and evaluate a number of physical gestures that form a basic vocabulary for interaction when using mobile phones capable of reading visual codes. These fundamental *interaction primitives* are based on camera input and simple image processing algorithms. The primitives can be combined to form more expressive interactions that provide rich input capabilities and effectively structure the output space. An interaction specification language defines rules that associate conditions of certain phone postures to actions, such as textual, graphical, and auditory output, which are performed by the mobile device. As described in detail below, these interaction primitives can be used in visual code image maps, augmented board games, product packaging, posters, and large public displays.

The following section gives a brief overview of related work. Section 4.3 outlines a number of application scenarios. Section 4.5 discusses interaction primitives, their combinations, and how they are indicated to the user. In Section 4.6, we define our user interaction model, which describes how to create visual code image map applications. Also, we describe an XML-based specification language, an authoring tool for visual code image maps, and a corresponding parser and interpreter on the

phone. In Section 4.7, we report about a usability study in which we evaluated our interaction techniques. We wrap up with a number of conclusions and ideas for future work.

4.2 Related Work

Several projects have investigated linking physical objects with virtual resources, using technologies such as RFID tags or infrared beacons [119, 211]. However, these projects were mainly concerned with physical linking per se or with the infrastructure required for identifier resolution. They were limited in the richness of user interactions and typically allowed just a single physical gesture (for example presence in the reading range of an RFID reader) and thus just a single action per object. We, in contrast, allow the semantics to be a function of both the *object* and the *gestural sequence*.

A number of projects focused on improving interaction with the device itself rather than with the user's environment. No objects in the environment were integrated in the interaction. In 1994, Fitzmaurice et al. [73, 77] prototyped a spatially aware palmtop device with a six degrees-of-freedom tracker to create a porthole window into a large 3D workspace. The actual processing was done on a graphics workstation. The palmtop sensed its position and orientation in space and combined input control and output display in a single unit – thus integrating "seeing" and "acting". One could explore the 3D workspace with the palmtop using an *eye-in-the-hand* navigation metaphor. The goal was to step out of the "claustrophobic designs and constraints" imposed by the form factor of handheld devices. While this was a vision in 1994, similar interfaces can be built today with handheld devices and interaction techniques as presented here.

Our work can be seen as an instance of an *embodied user interface* [72], in which the user directly interacts with the device itself – for example by tilting it – and both the manipulation and the virtual representation are integrated within the same object. Tilting of a handheld device has been explored as an input parameter for menu selection, scrolling, navigation, text entry, and 3D object manipulation [16, 91, 99, 100, 163, 222]. Readability problems of tilted displays, which we also experienced in this work, are described in [91] and [16].

In [70] Fishkin analyzed the idea of a physical grammar, and in [72] he addressed the issue of multi-gesture command sequences. Bartlett [16] used gestures – like tilting, snapping, shaking, tapping, and fanning – as a vocabulary of commands for a handheld electronic photo album. Our work tries to build compound interactions from a vocabulary of interaction primitives.

Bartlett [16] commented on some of the limits of embodied user interfaces: "perceived motion on the screen is the sum of the motion of the embodied device and the changes made to the display by the device. As you interact with the device by moving it, the orientation of the display to the user changes, then in response to that motion the display contents move on the display." This effect is especially severe for fast movements; however such movements are not required in our design. Rather, our work is more concerned with subtle yet easily and manually controllable changes.

Our interaction model allows us to define state spaces of phone postures in 3D. These issues are similar to those experienced with augmented reality environments

and with 3D input devices [100, 226]. In our case however, our 3D environment is the physical world perceived through the camera lens. The VideoMouse [100] is an optical mouse for 3D input based on video sensing and although it shares some similarities with our system, it is different in that it provides only very limited height sensing of just a few centimeters.

4.3 Application Scenarios

As outlined in the introduction, our system enhances the general input capabilities of mobile devices and provides a way to access mobile information services related to physical objects within a user's vicinity. Our system allows fine-grained control and access to various information items and services that can be physically hyperlinked [119] to objects in the environment. Typical objects that a mobile user might encounter include product packaging, vending and ticketing machines, posters and signs, as well as large electronic displays [10]. A few application scenarios are outlined below.

Figure 4.1: A tram stop (left and middle) and a vending machine (right) equipped with visual code image maps.

- **Tram stop.** The left and middle sections of Figure 4.1 show a tram stop information panel tagged with a visual code which allows users to access tram arrival times and to obtain further information by rotating the phone. To obtain information about the route of interest, users focus on the corresponding route number.
- **Vending machine.** The right part of Figure 4.1 shows a vending machine tagged with visual codes. To purchase products and confirm the purchase, users aim at the desired object. Of course in this scenario, a payment method needs to be in place.
- **Campus map.** Visual code image maps can help to find the location of an event on a campus map. A visitor to the campus could focus on an area labeled "current events" to get information about conferences, talks, and other events. The location of the event could then be highlighted on the mobile phone screen.

- **Augmented board games.** Computer-augmented board games are another good candidate for using visual code image maps since such games could benefit from a wide range of interaction possibilities that do not tie the user to a desktop computer.

  We have developed a mobile phone-supported memory game [130] that uses a number of the interaction techniques presented. The phone offers jokers to players to find the direction of a matching card, provides statistics about how many times particular cards have been looked at, etc. For the memory game we used an upside-down operation of the mobile phone. The phone is lying on the table with the camera facing upwards. Cards are swept over the phone camera in a particular way to trigger various jokers.

4.4 Visual Code Parameters

In this section, we review the orientation parameters of the visual code system that are used within the framework.

Code coordinate system. An essential feature of the visual code system is the mapping of points in the image plane to corresponding points in the code plane, and vice versa. With the help of this mapping, the pixel coordinates of the camera focus, which is the point the user aims at and which is indicated by a crosshair during view finder mode, can be mapped to corresponding code coordinates. As shown in Figure 3.4, each code defines its own local coordinate system that has its origin at the upper left edge of the code and that is independent of the orientation of the code in the image. Areas that are defined with respect to the code coordinate system are thus invariant to projective distortion.

Rotation. The recognition algorithm provides the rotation of the code in the image in degrees counterclockwise in the range of 0° to 359°.

Horizontal and vertical tilting. Another parameter is the tilting value between the image plane and the code plane. Horizontal tilting is defined as the angle between the x axis of the code coordinate system and the image plane. Analogously, vertical tilting denotes the angle between the y axis and the image plane. A tilting value of 1 means that the axis and the image plane are parallel. A tilting value less than 1 means tilting to the left/top, a value greater than 1 denotes tilting to the right/bottom.

Distance. The reading distance between a visual code and the camera is defined in such a way that the value 100 is assigned to the distance at which the code is just recognized in view finder mode. This distance is reached with current phone cameras if the code occupies about 3% of the image area. Defining the distance in terms of the size of the code in the image, instead of using the actual metric distance, keeps this parameter independent of the camera parameters. It is still adequate for the interaction purposes we envision.

Relative movement. Finally, the recognition algorithm is combined with a visual movement detection algorithm that solely relies on image data provided by the camera. The movement detection provides (x,y,θ) triples for relative linear and relative rotational movement. The movement detection does not rely on the presence of a code in the image.

All of the described parameters can be computed without knowing the optical characteristics of the camera used, like the focal distance. If these parameters are known, the tilting angles and the metric distance can be computed. Figure 3.8 shows all of the code parameters as seen in the Visual Code Recognizer application.

4.5 Interaction Techniques

In the following, we introduce the interaction techniques that are used in visual code image map applications. These techniques rely on sensing visual codes from different perspectives. We describe how interactions are combined from basic building blocks and how interaction cues guide users in the interaction process.

4.5.1 Interaction Primitives and Interaction Cues

Combined interactions are constructed from basic building blocks, called *interaction primitives*. *Static interaction primitives* require the user to aim their camera phone at a visual code from a certain orientation and to stay in that orientation. We defined two kinds of *dynamic interaction primitives*, which either involve "sweeping" the camera across a visual code or simply moving the phone relative to the background.

To facilitate information access and guide users in their interaction flows, each interaction primitive is associated with one or more *interaction cues* in the form of an icon. They appear on the mobile device's display and provide users with an indication of the possible interaction postures. Visual cues can optionally be combined with auditory icons.

For instance, the leftmost *rotation* interaction cue in table 4.1 indicates to users to rotate the mobile phone either clockwise or counterclockwise in order to access more information. The rightmost cue for the *distance* primitive means that more information can be obtained by moving the phone closer to the code – relative to the current posture.

An interaction cue should be both intuitive when encountered for the first time and easy to remember. Interaction cues should also be unambiguous so that it is easy to distinguish between different interaction primitives. In our design of interaction icons, we use color extensively since it facilitates distinguishing between different interaction primitives, and color displays are available on all camera phones. We restrict icon size, since the interaction cues must occupy only a small part of the phone display. They have to be rather simple and plain in order not to unnecessarily distract the user or clutter the interface.

4.5.2 Input and Output Capacity

Static interaction primitives map certain orientations of the mobile phone, also called *postures*, to individual information aspects. The posture of the device is determined with respect to a visual code in the camera image. With the term *input capacity* we denote the number of discrete information aspects that can be sensibly encoded in each of the interaction primitives. The input capacity is a measure of how many discrete interaction postures can be easily and efficiently located and selected by users. An important performance aspect is the time it

takes a user to locate an individual information item among a set of available information items. This time depends on the kind of interaction primitive, the number of available postures, as well as the quality of feedback that is provided to the user. For static interaction primitives, discrete postures are possible, like focusing a particular information area, as well as more fine-grained forms of input, like the continuous adjustment of a value by moving closer to or away from a code. For each interaction primitive, we will give an estimation of its input capacity, which has been obtained experimentally and during user testing. In this work, discrete postures activate associated information aspects. It would also be conceivable to use voice commands for this purpose. In addition, voice commands can be taken as a way to get further input once a certain posture has been reached. This combination of postures and audio input would realize a multi-modal user interface.

The output capabilities of mobile devices are limited due to the small screen size. Thus, the amount of textual and graphical information that can be shown on a single screen is limited. Fortunately, the interaction postures are very well suited for structuring the presentation of data by distributing it across several related interaction postures. With the proposed approach, text and graphics can be overlaid over the camera image as known from augmented reality applications. Graphical elements can be registered with objects in the image, i.e. shown at the same position and resized and adapted to the viewing perspective. This makes the connection between the physical object and the information shown on the display more obvious to the user and avoids its isolated presentation without any other context.

Output can also be used to realize a feedback loop to the input side, which has an impact on the input capacity. To create the feedback loop, characteristic icons can represent interaction primitives and indicate interaction possibilities to the user. Mobile devices typically have an audio output channel which can be also used for establishing a feedback loop. Characteristic audio cues ("earcons" [33]) can be permanently associated to different information or interaction types. Auditory cues have the advantage that they do not take up any space on the device display. Designing audio feedback needs to be done with care, because it has privacy repercussions or might be distracting to some users. Another interesting option to support the feedback loop between output and input is available with the phone's vibration feature ("tactons" [32]).

4.5.3 Static Interaction Primitives

Static interaction primitives are based on the parameters of the visual code system, as well as the focused area, key presses, and the time stayed in a given interaction posture. For the user, this means finding a posture in view finder mode guided by the interaction cues, staying in that posture, and optionally taking a high-resolution picture to "freeze" the current posture. The "information freezing" feature stops the view finder mode and shows the information related to the last captured phone posture. The available static interaction primitives, their estimated input capacity, and the associated icons are shown in table 4.1.

Pointing. The *pointing* interaction primitive requires targeting an area with the crosshair shown on the device display in view finder mode. The area is defined in terms of code coordinates. The input capacity is only limited by the number

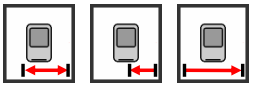
Static interaction primitive	Input capacity	Interaction cues
pointing	number of information areas	information area is highlighted
rotation	7	
tilting	5 (+4 if using NW,NE,SW,SE)	
distance	8	
stay	unlimited (time domain)	 (icon has a highlighted display)
keystroke	12 (keypad) + 5 (joystick)	 (icon has a highlighted keypad)

Table 4.1: Input capacity and interaction cues of static interaction primitives.

of areas that can be reached with the crosshair while the associated visual code is in the camera view. Section 4.6.5 presents techniques for extending the scope of reach. The borders of an area are highlighted when the associated visual code is recognized and the focus point is inside that area.

Rotation. The *rotation* interaction primitive associates rotation regions with discrete information items. For usability purposes, the rotation of the phone should be limited to ±90° from the upright position. To improve legibility, text should be aligned vertically if the rotation is greater than 45°. Users can complete up to 7 discrete postures, which correspond to regions that cover about 30° each, centered at 0°, ±30°, ±60°, and ±90°. *Rotation* is also usable as a continuous input control for a parameter value, such as the volume of a hi-fi system.

Tilting. During user testing, *tilting* turned out to be the most challenging interaction primitive for users since it requires turning the head in order to follow the device screen. We therefore do not use precise tilting values, but only an indication of the direction (“north”, “west”, “south”, “east”, and central position). This results in an input capacity of five postures. It is straightforward to extend this by “north-west”, “north-east”, “south-west”, and “south-east”, resulting in an overall input capacity of 9 postures.

Distance. The *distance* is measured during view finder mode and has an input capacity of 8 easily distinguishable distances. Distance is another a good candidate for continuous input.

Stay. The *stay* interaction primitive requires the user to stay in a certain posture. It automatically changes the provided information after a certain amount of time. The time interval can be freely specified, but should depend on the amount of information shown on the device screen. For a few lines of information it would typically be a couple of seconds. This primitive can be combined with the *keystroke* primitive described next, in order to realize a “timeout kill” mechanism as used for

multi-tap text entry [222]. The input capacity is unlimited in principle, requiring the user to wait.

Keystroke. Finally, the *keystroke* interaction primitive consists of pressing a button on the device's keypad or using the device's thumb-operated joystick. Our target device has a 12 button numeric keypad and a non-isometric joystick with five states (left, right, up, down, press). The input capacity of this interaction primitive is obviously limited by the number of available keys.

The numbers given for the discernible input capacity of each interaction primitive decrease, if the basic primitives are combined with each other, as shown in the next sections.

4.5.4 Dynamic Interaction Primitives

There are two kinds of dynamic interaction primitives. With the first, the phone is moved ("swept") across the code in a certain direction while the camera is in view finder mode. The direction of movement is sensed by the mobile device and used as the input parameter. Interaction symbols for this kind of dynamic interaction primitive are not shown on the device display, but printed next to the code. For each possible direction of movement, a label is given, informing the user about the operation that will be triggered when the code is "swept" in the indicated direction. These interaction primitives are suitable for "blind" operation, in which a single operation is selected and immediately triggered after the movement. Sweep primitives can be regarded as the equivalent of a crossing-based interface for visual codes [3]. The input capacity amounts to 4 for both horizontal and vertical movement as well as for diagonal movement. A combination of both movement types seems to be too complex. With current phone hardware, the movement must not be too fast, in order for the codes to be reliably detected at multiple positions in the image. The input capacity and interaction cues are depicted in table 4.2.

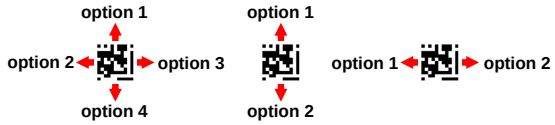
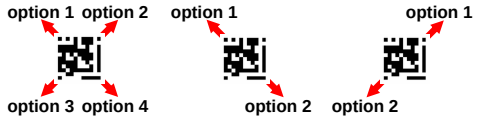
Dynamic interaction primitive („sweep“)	**Input capacity**	**Interaction cues (printed next to the code)**
horizontal or vertical movement	4	
diagonal movement	4	

Table 4.2: Input capacity and interaction cues of *sweep* interaction primitives.

The second kind of dynamic interaction primitives is based on the optical movement detection algorithm that does not require a visual code in the camera image. It provides relative linear movement and relative rotation. It is not suited for discrete input, but for continuous adjustment of parameter values or for direct manipulation tasks. The corresponding interaction cues can be shown on the device display, printed next to a code, or shown on an electronic display to indicate that

its objects can be directly manipulated by movement detection. Table 4.3 contains the capacities and interaction cues of these interaction primitives.

Dynamic interaction primitive (relative movement)	**Input capacity**	**Interaction cues**
relative linear movement	4 (continuous)	
relative rotation	2 (continuous)	

Table 4.3: Input capacity and interaction cues of relative movement interaction primitives.

A *clutching mechanism* is required to prevent incidental motions of the phone from triggering unwanted dynamic interaction primitives. In our system, the relative movement tracking is active while the phone's joystick button is held down. Releasing the button exits the relative movement detection state. This is also known as a *quasimode* as defined by Raskin in [159]. In Buxton's model [40], pressing the joystick button down corresponds to a state transition between state 1 ("tracking") and state 2 ("dragging"), releasing the button again transitions back to state 1.

4.5.5 Combinations of Interaction Primitives

The basic interaction cues are designed in such a way that they can be combined to form more complex interaction cues. Table 4.4 shows the possible combinations of two static interaction cues. When the mobile display shows a combination interaction cue, this means that the user has a choice to select between more than one interaction primitive to reach further information items. The usability of such combinations is discussed in Section 4.7. Combinations of more than two interaction cues should be avoided in order not to confuse the user. Even with combinations of only two static interaction cues, a large number interaction possibilities results.

Some of the static interaction primitives can be combined with the dynamic *sweep* interaction primitives. Each of the eight directions of movement can be combined with the following static interactions: rotation, tilting, and distance. The idea is to move the camera across the code in the chosen direction while keeping a certain rotation, tilting, or distance. In the case of rotation, for example, it should be easy to hold the phone rotated 90° counterclockwise from the upright position.

Combinations of static primitives and dynamic primitives that sense relative movement seem to be more practical. Even if they cannot be executed simultaneously, performing a dynamic after a static interaction primitive is useful. A user first selects a certain parameter using a static interaction primitive – like tilting – and then uses relative linear movement to adjust the value. The relative movement detection is activated while the user is holding the joystick button down. This kind of combination resembles a "point & drag" transaction in classical GUI interfaces [40].

Combination	Interaction cue	Combination	Interaction cue
pointing & rotation	 + highlighted area	**rotation & stay**	
pointing & tilting	 + highlighted area	**rotation & keystroke**	
pointing & distance	 + highlighted area	**tilting & distance**	
pointing & stay	 + highlighted area	**tilting & stay**	
pointing & keystroke	 + highlighted area	**tilting & keystroke**	
rotation & tilting		**distance & stay**	
rotation & distance		**distance & keystroke**	

Table 4.4: Combinations of two static interaction primitives with example interaction cues.

4.6 Visual Code Image Maps

In this section, we describe how combinations of interaction primitives can be applied in entire *visual code image map* applications. Visual code image maps consist of a number of areas, which are located close to a visual code and associated with multiple information aspects or specific operations. Areas can cover a certain region in the vicinity of a code, occupy the same space as the code, or even be defined as infinitely large. Area locations and shapes are defined in terms of the coordinate systems of the visual codes located near them. Area-related information is accessed by varying the input parameters provided by the visual code system. The input parameters are abstracted to a set of postures that are easily discoverable and applicable by users. The postures are specified as combinations of interaction primitives in an image map definition.

Figure 4.2 shows an example interaction flow for a simple image map. To the left of the screenshots, the enlarged interaction cues are drawn. An elliptical region next to a visual code is associated with six information items. At a farther distance (depicted in the upper three screenshots), three different information items are presented. The user just has to stay at that distance. The *stay* user interaction symbol indicates that more information will be displayed by waiting. Moving closer to the code plane, the interaction cue changes and another information aspect is displayed (depicted in the lower three screenshots). In the near distance posture, more information can be accessed by rotating the phone to the left (counterclockwise) and to the right (clockwise). The underlying user interaction model is discussed in more detail below.

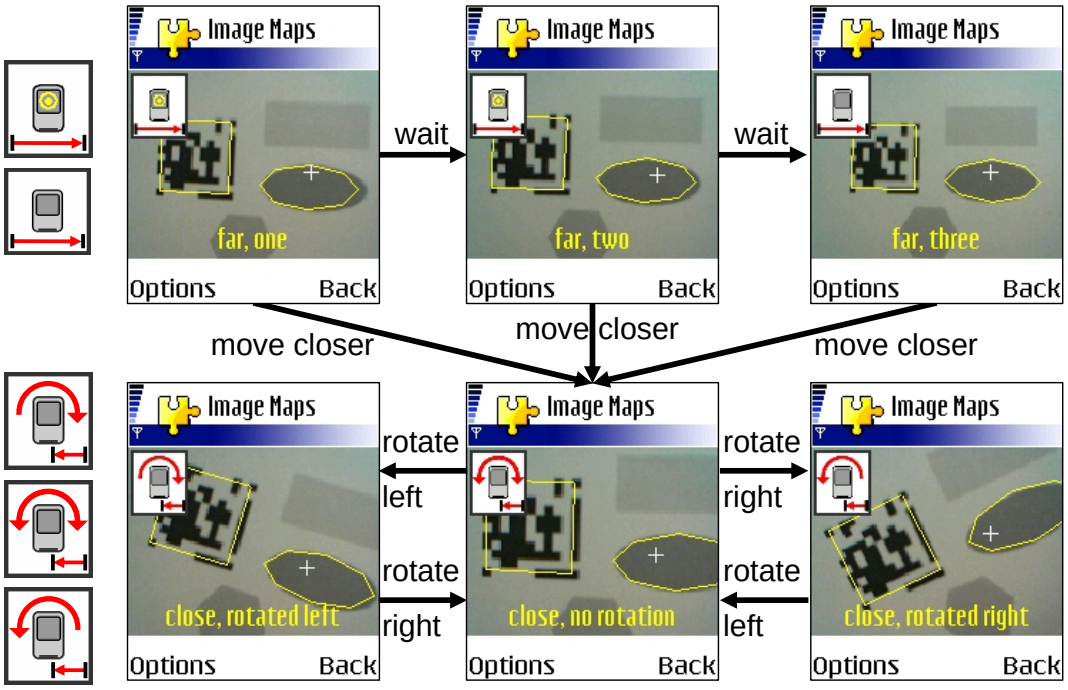


Figure 4.2: An example interaction flow in a visual code image map.

When designing overlays over the camera image, design guidelines as described in [200] should be taken into account. The visual context given by the camera image should be maintained as far as possible. A graphical representation should be chosen such that the underlying context is revealed. This avoids the issue that the user has to split visual attention between the camera image (the "context") and the generated graphical overlay. It enables *dual attention*, which is characteristic of see-through tools. In addition, unnecessary information, i.e., information that is not part of the currently pointed area should be hidden. This is especially important for small displays.

4.6.1 User Interaction Model

The user interaction model determines how a user can browse information or trigger actions in a visual code image map. We use a stateless model that only considers the currently sensed parameters. Each interaction posture is associated with a rule. A rule consists of a condition and a result that is activated when the condition is met. A condition is made up of a set of constraints. A constraint restricts the valid range of a single input parameter. The rules are continuously checked and their results activated if their conditions are met. The visual code image map designer has to ensure that conditions are mutually exclusive. If they are not, the order of execution is undefined. For non-idempotent functions, it is important that they are not activated multiple times. Checking such constraints is part of the semantics of each action result and not specified in the interaction model. The stateless model is easy to understand for users, since they always see the same result if the same input posture is chosen. For image map designers, image map applications are easy to specify in this model. In a completely stateless model, some of the proposed

combinations of static and dynamic interaction primitives cannot be realized. It is therefore slightly extended as described below.

State-based models are inherently difficult to understand for users since the system can behave differently on the same input parameters if it has different states. We therefore limited the notion of input state in the system to the relative movement interaction primitives. In order to activate relative movement detection, the user has to hold the joystick button down. The user's last posture receives relative movement updates while the joystick button is held down. Releasing the button exits the relative movement detection state. This *quasimode* scheme ensures that the user is not inadvertently locked in a state. The second notion of state is introduced with the *stay* static interaction primitive. It becomes true when the time stayed in a certain posture is within a predefined time range. The state is thus defined by the set of other constraints of a condition, without the *stay* interaction primitive. A rule containing a *stay* interaction primitive fires when all other constraints have been true for the specified amount of time. The timeout is reset when the rule becomes invalid again.

4.6.2 Visual Code Image Map Specification Language

Based on the interaction model described above, we developed a visual code image map specification language. The specification language is XML-based. Depending on the visual code value, different measures are taken to retrieve the specification of an unknown image map. The XML description is loaded from the local file system, obtained via Bluetooth or the mobile phone network. It is parsed and the extracted information is used to present information and provide functionality according to the image map. We assume that up-to-date information is inserted into the XML file on the server, for example via PHP scripting.

The image map specification of the example in Figure 4.2 is given below. The XMLSchema description is detailed in Appendix A. An `ImageMap` consists of one or more named `Area`s whose extent is defined in `Rectangle`, `Ellipse`, or `Polygon` elements. These elements specify the boundaries of an `Area` with respect to the coordinate system of a particular visual code, whose value is given in `coordinateSystem` attribute. If an area can be seen from multiple visual codes, multiple `Rectangle`, `Ellipse`, or `Polygon` elements are present.

`Rule` elements first list the constraints – like `Rotation`, `Tilt`, and `Distance` – and then specify a result, either as an `Information` result or an `Action` result. Information and action results optionally contain interaction cues. Compound `IconicCue`s can be created by combining the basic cues shown in table 4.1. Other possibilities are `AuditoryCue`s, which provide audio feedback, and `TactileCue`s, which use the device's vibration alarm feature for generating feedback. The `Line` element shows a single line of textual output. Action results contain the attributes `functionName`, `arguments`, `body`, and `phoneNumber`.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<ImageMap imageName="test.jpg">
<Area name="ellipse">
```

```
<Ellipse coordinateSystem="0x0000000000123128abcabc">
  <Point x="14.42" y="5.26"/>
  <Point x="28.33" y="11.49"/>
</Ellipse>

<Rule name="Far1">
  <Distance start="70" end="100"/>
  <Stay start="0.0" end="2.0"/>
  <Information>
    <IconicCue name="Stay"/>
    <IconicCue name="DistanceCloser"/>
    <Line value="far, one"/>
  </Information>
</Rule>

<Rule name="Far2">
  <Distance start="70" end="100"/>
  <Stay start="2.0" end="4.0"/>
  <Information>
    <IconicCue name="DistanceCloser"/>
    <IconicCue name="Stay"/>
    <Line value="far, two"/>
  </Information>
</Rule>

<Rule name="Far3">
  <Distance start="70" end="100"/>
  <Stay start="4.0" end="6.0"/>
  <Information>
    <IconicCue name="DistanceCloser"/>
    <Line value="far, three"/>
  </Information>
</Rule>

<Rule name="CloseRotationNone">
  <Distance start="0" end="70"/>
  <Rotation category="absolute" start="345" end="15"/>
  <Information>
    <IconicCue name="DistanceFarther"/>
    <IconicCue name="RotationBothDir"/>
    <Line value="close, no rotation"/>
  </Information>
</Rule>

<Rule name="CloseRotationCW">
  <Distance start="0" end="70"/>
  <Rotation category="absolute" start="270" end="345"/>
  <Information>
    <IconicCue name="DistanceFarther"/>
    <IconicCue name="RotationCCW"/>
    <Line value="close, rotated right"/>
  </Information>
</Rule>

<Rule name="CloseRotationCCW">
  <Distance start="0" end="70"/>
  <Rotation category="absolute" start="15" end="90"/>
  <Information>
```

```
      <IconicCue name="DistanceFarther"/>
      <IconicCue name="RotationCW"/>
      <Line value="close, rotated left"/>
    </Information>
  </Rule>

</Area>
</ImageMap>
```

4.6.3 Information Results

Information results can consist of auditory cues, textual overlays over the camera image, bitmap overlays, and overlays of graphical shapes. Textual overlays can appear at a constant position in the mobile's display, which is the default in the current implementation. The text position can also be tied to specific code coordinates and thus appear as an overlay of an element in the image map. Bitmap overlays[1] can either appear at a constant display position or located at specific code coordinates. As with textual output, free rotation of images is an expensive operation for current mobile devices and can thus not be performed in real time on current devices. The Symbian operating system, for example, only provides functions for rotating text in steps of 90°, which is sufficient for legibility in the case of rotation. An example information result specification and its presentation on the device are shown in Figure 4.3.

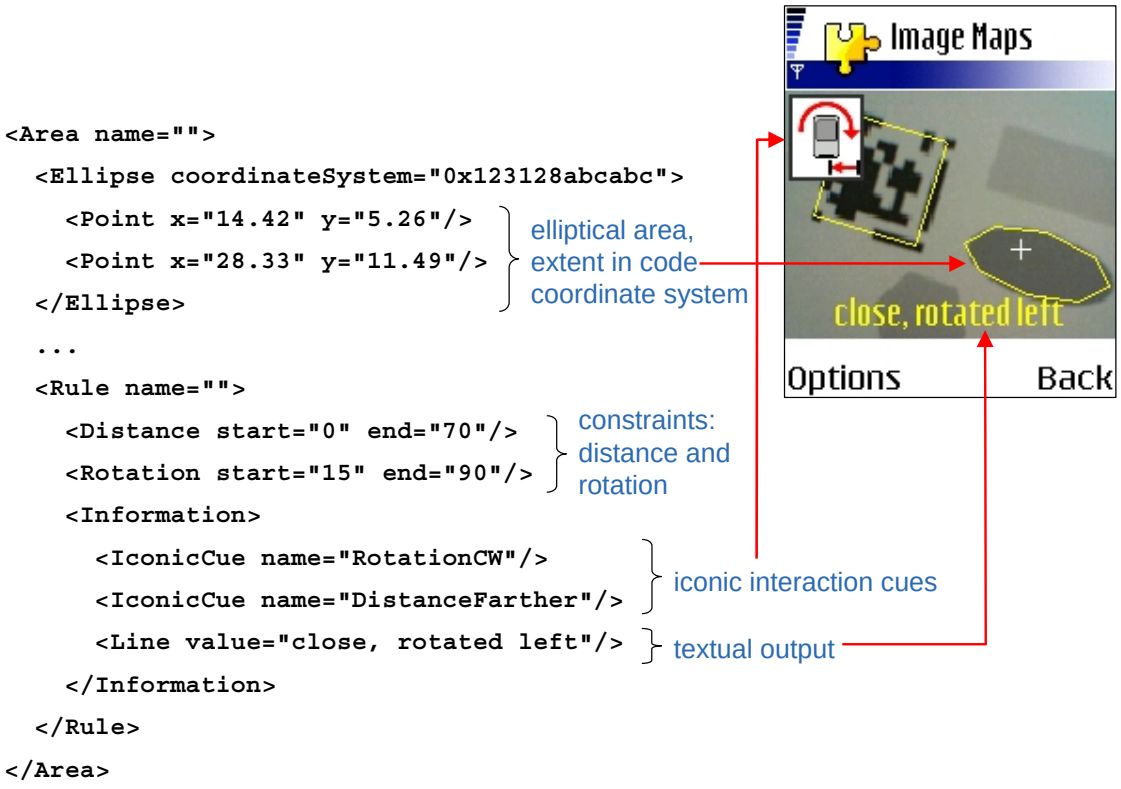


Figure 4.3: Specification and display of an information result.

[1] Bitmap overlays are not implemented.

Graphical overlays, such as rectangles, ellipses, and polygons are automatically adapted to perspective distortion by using the code coordinate system mapping. Graphical overlays can be specified using the `DrawPolygon` element. The attributes `penColor` and `brushColor` specify the corresponding colors in (blue, green, red) format. If no `penColor` is given, black is used; if no `brushColor` is given, the interior of the polygon is transparent. The attribute `penSize` defaults to 1. Figure 4.4 shows an information result with two polygons. It is possible to specify multiple textual and graphical outputs in a single information result.

```
<Information>
    <IconicCue name="RotationBothDir"/>
    <DrawPolygon penColor="0000ff"
     penSize="3" brushColor="000080"
     coordinateSystem="0x10000">
        <Point x="16" y="16"/>
        <Point x="22" y="16"/>
        <Point x="22" y="22"/>
        <Point x="16" y="22"/>
    </DrawPolygon>
    <DrawPolygon ...> ... </DrawPolygon>
    <Line value="2 houses"/>
    <Line value="7 field workers"/>
</Information>
```

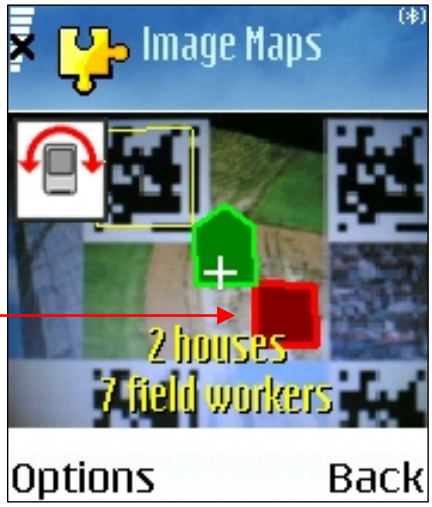


Figure 4.4: Graphical overlays within an information result.

Information results have the optional attribute `dynamic`. If this parameter is set to `true`, the contents of the information result are dynamically retrieved via Bluetooth. In this case, the name of the activated `Rule` is given as an argument. Each time a dynamic rule is activated – i.e. all of its constraints are satisfied – a special request is made via Bluetooth. Textual as well as graphical output can be dynamically created in this way. In addition, as long as a dynamic information result is active, all keypad input is reported via Bluetooth. We used this feature for dynamic content loading in an augmented board game [31].

4.6.4 Action Results

Triggering an action result consists of starting the requested application on the device and dispatching the provided arguments in the format the application requires. The semantics of the arguments depend on the given application. In the simplest case, the argument string provided in the XML description is simply passed on to the application. In a more complex case, it requires parsing the argument string and calling multiple methods on the phone application. The action result needs to define whether it has to be executed on each image update while the corresponding condition is valid, once as the rule first becomes active, or only when the joystick is additionally held down. Example action results are starting the WAP browser

with a specific URL as an argument, storing vCard and vCalendar entries, placing a phone call to the number given in the argument string, invoking the SMS editor, or sending a predefined text message without invoking the SMS editor. Other action results include opening a Bluetooth connection to report relative movement updates and visual code sightings.

The examples below illustrate the implemented actions. For all actions, we assume that the corresponding rule expects a keystroke input to actually trigger the application. The information lines are displayed as soon as the corresponding rule is activated. The first two actions start the HTML and the WAP browser, respectively. The internal WAP browsers of current devices are often capable of rendering simple HTML as well. The action `sendStaticSMS` automatically sends a predefined text message to a predefined number, without invoking the editor. The other variants allow the user to edit the text message by invoking the editor with pre filled text (`sendStaticTextSMS`) or with a pre filled phone number (`sendStaticNumberSMS`).

```
<Action functionName="startHTMLBrowser"
  argument="http://www.inf.ethz.ch">
    <IconicCue name="Keystroke"/>
    <Line value="Start HTML browser."/>
</Action>

<Action functionName="startWAPBrowser"
  argument="4 http://wap.ethz.ch">
    <IconicCue name="Keystroke"/>
    <Line value="Start WAP browser"/>
</Action>

<Action functionName="sendStaticSMS"
  phoneNumber="0791234567"
  body="This is an SMS with a predefined body and number.">
    <IconicCue name="Keystroke"/>
    <Line value="Send predefined SMS."/>
</Action>

<Action functionName="sendStaticTextSMS"
  body="This is an SMS with a predefined body.">
    <IconicCue name="Keystroke"/>
    <Line value="Send predefined SMS."/>
</Action>

<Action functionName="sendStaticNumberSMS"
  phoneNumber="0791234567">
    <IconicCue name="Keystroke"/>
    <Line value="This is an SMS to a predfined number."/>
</Action>
```

Interesting – but not yet implemented – options include storing personal information and status information. Personalized information comprises sending the user's vCard, sending the phone number, or the current location as GSM cell information. This information could then adapt the image map results. The disclosure personal information needs to be configurable by the user, of course. Status information could be implemented by storing cookies corresponding to individual image

maps and sending them back to the information provider when the user encounters the same image map again.

4.6.5 Focus Point Adaptation

A problem with visual code image maps comes from the fact that at least one code needs to be present in the camera view in order to compute the mapping from the image to the code coordinate system. This restricts the radius of action for moving the focus point. This situation is shown in the left section of Figure 4.5. The focus point is typically indicated to the user with a cross hair that is located in the middle of the display. The reachability problem can be solved in a number of ways. First, multiple codes can be dispersed throughout an image map. This raises aesthetical concerns and restricts the designer of an image map, because more space is occupied by visual markers. This might be mitigated in the future with higher-resolution cameras, allowing much smaller codes to be used. Additionally, with zoom cameras, a wide angle setting can be used to cover a larger part of the image map. Second, there is no reason why the focus point has to be located in the middle of the screen. One option would be to include the most suitable position of the cursor as a parameter in the image map specification. If a visual code is located to the left of a vertical arrangement of areas, for example, the focus point might be set horizontally to the right for easier targeting.

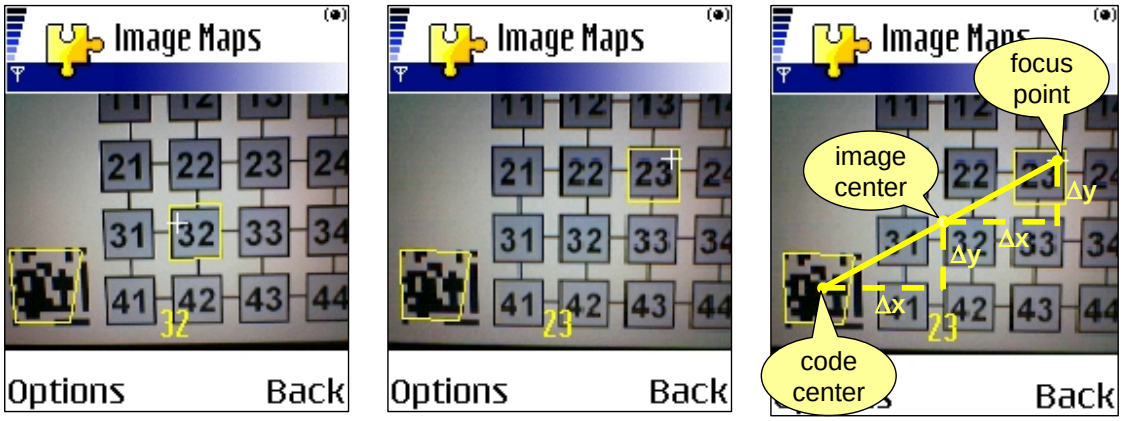


Figure 4.5: Central focus point (left) and adapted focus point (middle and right).

A third option is to dynamically adapt the position of the focus point depending on the position of the code center on the screen. This is shown in the middle and right of Figure 4.5. The focus point is computed as the mirror point of the code center point through the image center point. In usability tests, this smooth adaptation style seemed to be more predictable than another adaptation style, in which discrete focus point positions had been used. The smooth adaptation of the cursor position requires more dexterity than a fixed cursor position, but is manageable after a short time. If no code is present in the image for a certain time, the focus point is repositioned to the display center. If multiple codes are available, the nearest one is chosen for adaptation. If multiple codes are visible in the camera image, the adaptation can be disabled, because the reachability problem is no longer given. With dynamic adaptation, the reachable radius is increased by up to 100% compared to a focus point which is centered on the display.

4.6.6 Visual Code Image Map Editor and Interpreter

We have developed a visual code image map editor in Java (see Figure 4.6). The editor produces image map specifications from jpg, gif, and png images (typically a picture of the visual code in the real world) so that users can draw areas and specify constraints, interaction cues, and results. The resulting output is an XML file that can be stored on a server and which can be downloaded to the mobile phone.

On the device side, we have developed a generic visual code image map interpreter in C++ for Symbian devices. For each detected code, the interpreter tries to locate the corresponding image map and continuously checks for satisfied conditions in the available rules. As long as the conditions of a rule are satisfied, the corresponding information or external application is shown.

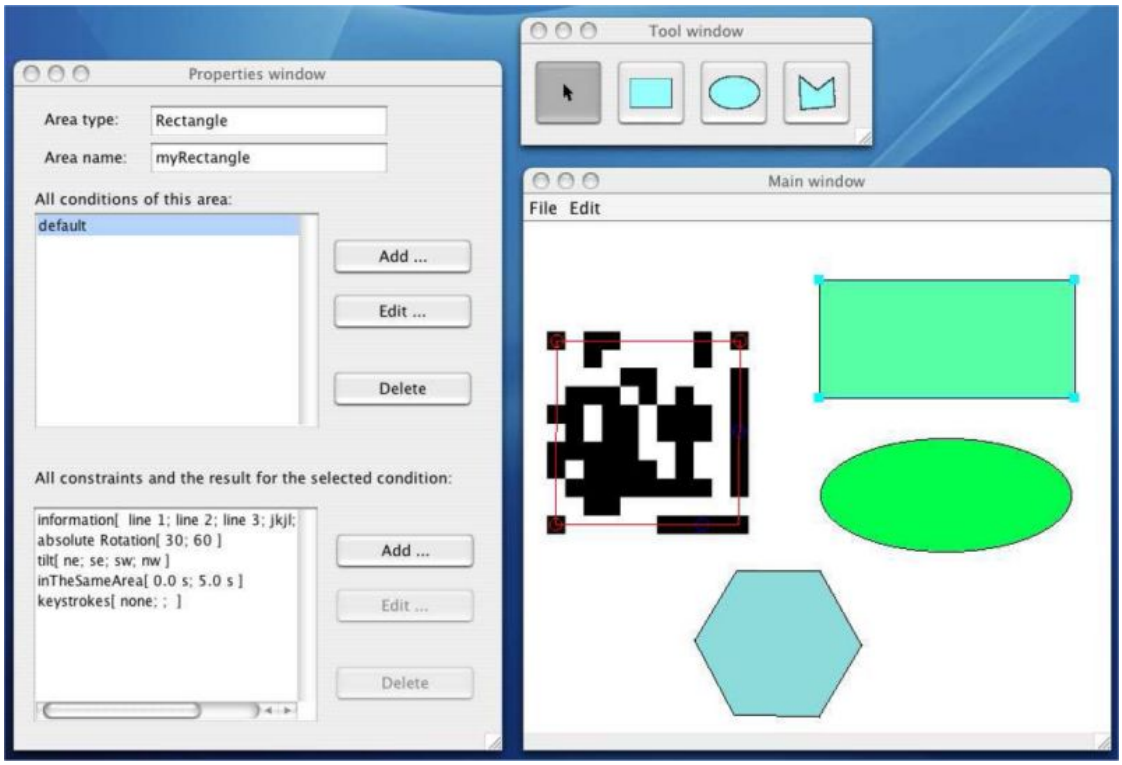


Figure 4.6: Visual code image map editor.

4.7 Usability Evaluation

4.7.1 Goals and Design

To understand the strengths and weaknesses of the individual interaction techniques, as well as the approach as a whole, we designed a qualitative usability study. It consisted of a questionnaire, two task execution parts, and a final interview. The material that was presented to test users is given in Appendix B. The questionnaire (see B.1) covered basic biographic data and asked about users' level of experience with mobile phones, text messaging, and playing computer games. The two task execution parts served different purposes. The aim of the first one was

to evaluate individual interaction primitives and their combinations independently from the semantics of a specific application. The second one used the campus map scenario outlined in Section 4.3 to help users understand the implications of using the interaction concepts in a broader context. The dynamic interaction primitives have not been evaluated in this study.

A number of technical factors influence the users' satisfaction with the interaction techniques, such as the size and quality of the display and the response time and reliability of the visual code system. However, we can expect that most of these technical factors are likely to improve. The usability evaluation thus tried to focus on issues that are inherent to the design of the proposed interaction techniques.

The first part of the study consisted of 15 individual tasks (see B.2.1 and B.4). Before the actual tests started, we demonstrated the interaction primitives to users and gave them an image map to get familiar with the interaction. Users employed the various interaction primitives to try and find a secret number and a secret letter in a particular image map. The first few tasks tested the dexterity required for the basic interaction primitives as well as how easily users were able to remember and distinguish between various interaction cues. The remaining tasks tested combined interactions. The second part of the study allowed users to get a feel for a possible real-word application (see B.2.2 and B.5). Lastly, in post-test interviews, users were asked to express their opinion about the overall system, rate the individual interaction primitives, and to give feedback about the presented scenario (see B.3).

When observing tasks, we used the think-aloud technique. Tasks were performed under quiet, laboratory-like conditions. Our evaluation procedure was adapted from the guidelines proposed in [84].

The execution of the study, including the initial questionnaire and the final interview, lasted approximately one hour per user. Eight users took part in our study, with ages ranging between 17 and 35. All of our users had some experience with personal computers, and all regularly used mobile phones for making phone calls and writing text messages. Some of them were heavy phone users, who often played mobile phone games and accessed information via WAP.

4.7.2 Findings and Discussion

Our results indicate the most challenging interaction primitive for users to do was *tilting*. The *pointing*, *distance*, and *stay* primitives were rated the best, followed by *keystroke* and *rotation*. For combinations which used *pointing* with other static primitives, we found two groups of people who preferred different user interactions. The first group, consisting of five participants, preferred user interactions that demand less manual dexterity. They favored the *pointing & stay* interaction, followed by the *pointing & keystroke* interaction. The second group, consisting of three participants, preferred the *pointing & distance* and *pointing & rotation* combinations. One possible explanation for this difference is dexterity. Since the first group seemed to have more problems with manual dexterity, they preferred passive user interactions, like *stay*, and well-known interactions, like *keystroke*. The second group, which had less problems with manual dexterity, liked combinations which used *distance* and *rotation* primitives the best because this gave them immediate control over the visual code application. With the *stay* primitive, the system forces the user to pause. This can be problematic, since the user cannot control

the duration of each state. The *pointing & tilting* combination was by far the most difficult interaction. The reason seems to be that this combined interaction which asks users to simultaneously focus on an area while keeping a visual code in the camera image and tilting the phone to the required position is very demanding for first time users.

During view finder mode, camera images are continuously sampled and the display is updated accordingly. We provided an "information freezing" feature that stops the view finder mode and shows the information related to the last captured phone posture. Some users used this feature as soon as they reached the correct phone posture. The feature is extremely important in that it provides users with some sense of permanence and stability.

We observed some learning effects during the usability test. In general, participants managed the *rotation* interaction primitive in task 13[2] more easily and more rapidly than in task 3[3] although task 13 was more difficult. Evidently, participants had improved their skills in handling the interaction techniques during the performed tasks and, moreover, the tasks did not seem to exhaust them.

All user interaction cues seemed to be easy to learn and remember. However, two participants first confused the *tilting* and *rotation* interaction cues. But after this first mistake they had no further problems. The interaction cues have been redesigned in the meantime and now use different colors for indicating "rotation" (red) and "tilting" (blue), which should improve distinguishability.

In the second part of the study, users had to look up a building on a campus map that was printed on a poster and attached to the wall. Observation showed that the application was not self-explanatory. Most users needed some instructions on how to use it. The following observations have been made during the second part:

- If the information areas are not clear, users tend to focus on the visual codes since they assume that they contain information items. The observation of this behavior offers two conclusions: first, a visual code image map designer should pay attention to design obvious information areas. Second, in a complex image map application, focusing directly on the visual codes should trigger a help menu or a description of the application.

- Most users tended to read printed information that was captured by the camera and shown on the phone display by looking directly at the printed information. They seemed to avoid the display since the size and quality is not yet good enough. However, users had no problems to read generated textual information and graphical overlays over the camera image directly on the display.

- Two users spontaneously remarked that they find it easier to access information aspects with the information map application on the wall than with the newspaper-like tasks on the table.

- Reading distances differed between users, but all users managed to find a suitable distance between a printed code and the camera after a short time.

[2] *rotation* plus *pointing* and only one visual code in range
[3] *rotation* plus *pointing* and two visual codes in range

4.8 Summary

We have used our visual code system to augment camera phones with physical gestures in order to turn them into versatile interfaces to real-world objects. The proposed conceptual framework allows constructing rich interaction sequences by combining a few basic interaction primitives in a flexible way. Even though the input capacity of each individual interaction primitive is limited, their combination results in a large number of input postures. The chosen stateless interaction model, or rather its realization as an XML-based specification language, adequately describes visual code information maps, including input postures, phone movements, information results, and action results. An authoring tool and a generic interpreter application for Symbian phones enable the creation and usage of visual code image map applications.

The user evaluation showed that most of the postures are easily and quickly discoverable with the help of graphical and auditory cues. It also showed that users generally like the proposed interaction paradigm. A few undesirable combinations of interaction primitives have been revealed that should be avoided by visual code image map designers.

Chapter 5

Visual Code Widgets as Building Blocks for Marker-Based Interaction

In this chapter we take a step beyond the work described before in that we add an additional layer of abstraction – a set of generic widgets – to the concept of marker-based interaction. We present a set of graphical user interface elements for 2-dimensional visual codes. The proposed widgets are suitable for printing on paper as well as showing on electronic displays. They define basic building blocks for creating applications that incorporate mobile devices as well as resources in the user's environment, such as paper documents, posters, and public electronic displays. In particular, we present visual code menus (vertical menus and pie menus), check boxes, radio buttons, sliders, dials, text entry widgets, and free-form input widgets. We describe the associated interaction idioms and outline potential application areas.

5.1 Introduction

Widgets are generic, reusable, directly manipulable, self-contained visual screen idioms. They are the primary building blocks for creating graphical user interfaces. Examples are buttons, check boxes, edit fields, and tooltips. Operating systems typically come with a default widget toolkit, which defines a set of basic interaction metaphors across all applications. Each widget solves a certain input or interaction problem and offers familiar affordances to the user. Cooper [47] classifies widgets[1] into the following four categories:

- *Imperative widgets*, such as buttons, initiate a function.
- *Selection widgets*, such as check boxes or radio buttons, allow the user to select some option or data.
- *Entry widgets*, which can either be bounded, like sliders, or unbounded, like edit fields, enable the input of data.

[1] Cooper actually uses the term "gizmos."

- *Display widgets*, which will not be discussed here, serve the manipulation of the appearance of an application itself.

Figure 5.1: A user interacting with a printed visual code pie menu. On the phone screen, the focused pie slice has a yellow frame.

With the use of such widgets the expressivity of marker-based input becomes richer, which enhances the overall user interface capabilities of mobile devices. The code coordinate system and the ability to create precisely aligned graphical overlays in the live camera image are essential features of the visual code system, which enable visual code widgets. In this way, printed documents and large-scale displays serve as extended user interfaces of mobile devices, which are not subject to the size restrictions of handheld device screens. Moreover, paper is permanently available, always ready for interaction, and – when used with visual code widgets – can help to quickly establish the context of an application.

Interaction would typically take place as follows (see Figure 5.1): The user finds a visual code widget, for example in a magazine, on a doorplate, on a poster, or on a survey form. She starts the generic recognizer application on her phone or PDA and aims at the widget. The widget appears on the device screen in view finder mode and is updated in real time as the user moves the device relative to the widget. The state of the widget is superimposed over the camera image, for example by drawing a current slider position. The user knows how to interact with the widget, since the familiar layout of the widget offers clear affordances. It allows her to quickly enter data or to make selections. The context of the interaction is implicitly established and takes almost no time or effort for the user to set up. The

proposed widgets together with their interaction idioms thus enable the connection of mobile devices to objects in the real world.

In the following section, we discuss related work. Section 5.3 describes the widget encoding scheme. Section 5.4 presents visual code menus. Section 5.5 describes selection and data entry widgets. Section 5.6 outlines application areas and a summary is given in Section 5.7.

5.2 Related Work

In comparison to traditional graphical user interface widgets [47], visual code widgets provide similar functionality, but are more tangible and permanent and require different sensory-motor skills. With their unique identification number, they can automatically identify their target application, such that any input is directly sent to the target application. The user does not have to find and start the target application, but can use a generic widget interpreter application.

Visual code widgets operate on two layers of information – the device screen, which is "transparent," and the printed marker surface. Thus similar issues arise as for transparent layered user interfaces and see-through tools. Harrison et al. [92] discuss the "switching costs" for shifting attention between the two layers and the visual interference of background and foreground objects. See-through tools or toolglasses [24] are widgets that are layered between application objects and the cursor. They can be positioned over application objects in order to modify their appearance or set the context for command execution. With visual code widgets the printed widgets form the background layer and their state is superimposed over them on the foreground layer. Since there are no application objects in the view of the camera there is less visual interference. Still, textual output might interfere with the camera image in the background. Harrison et al. [93] describe a font enhancement technique, called *anti-interference font*, that mitigates this issue.

The proposed widgets are generic in the sense that they could also be implemented with other types of markers, as long as these markers provide the required orientation parameters. In particular, the code coordinate system described below is essential for detecting the focused point and for precise registration of graphical overlays. In addition, the markers and orientation parameters have to be recognized in real time in the live camera image without a perceptible delay. The markers also need to have a capacity of at least 64 bits to encode the widget type.

Other 2-D marker technologies include *CyberCode* [165], *TRIP* [53], *SpotCodes*[2], and *SemaCode*[3]. CyberCodes and TRIP tags do not operate on mobile phone class devices. Even if they would, CyberCodes encode only 2^{24} and TRIP tags just 3^9 possible values, making it difficult to store the widget type information in the code itself. A few camera phones can read *QR Codes* [108]. These devices do not compute any orientation parameters, which are necessary for visual code widgets.

SpotCodes are a derivative of *TRIP* tags for camera phones. The system recognizes the rotation of the tag in the image, but does not seem to support the precise registration of virtual graphics with the camera image. Some interesting interaction possibilities, such as rotation controls and sliders for large displays, are described

[2] `www.highenergymagic.com`

[3] `www.semacode.org`

in [133]. In contrast to visual code widgets, these controls are generated on the large display and are thus not usable with non-electronic media, such as paper.

5.3 Visual Code Widget Encoding

To enable a smooth style of interaction, visual code widgets are recognized in view finder mode and the overlay graphics are updated in real time as the device moves. In order not to delay spontaneous interaction, the type and layout of each widget is stored in the code value itself. This means that there is no initial resolution step in which the code value would have to be sent to a server, mapped to the widget description, and sent back to the device. Even with communication over Bluetooth this would incur a noticeable delay. Via the mobile phone network the delay would be much higher and would moreover entail connection fees.

We devised a very compact way of encoding for storing the type and layout of each widget within a visual code (see Figure 5.2). The description of most widgets takes only 12 bits of the code, leaving 64 bits of the 76-bit value for an identifier. Application designers can freely choose this identifier and use it to establish the global application context. For use with visual code widgets, the address space is divided into four classes. The code class is stored in the two most significant bits of the code. Code class 3 is used for visual code menus, code class 2 for selection and data entry widgets, and code classes 0 and 1 are unused. The detailed layout of the encoding is given in the individual subsections.

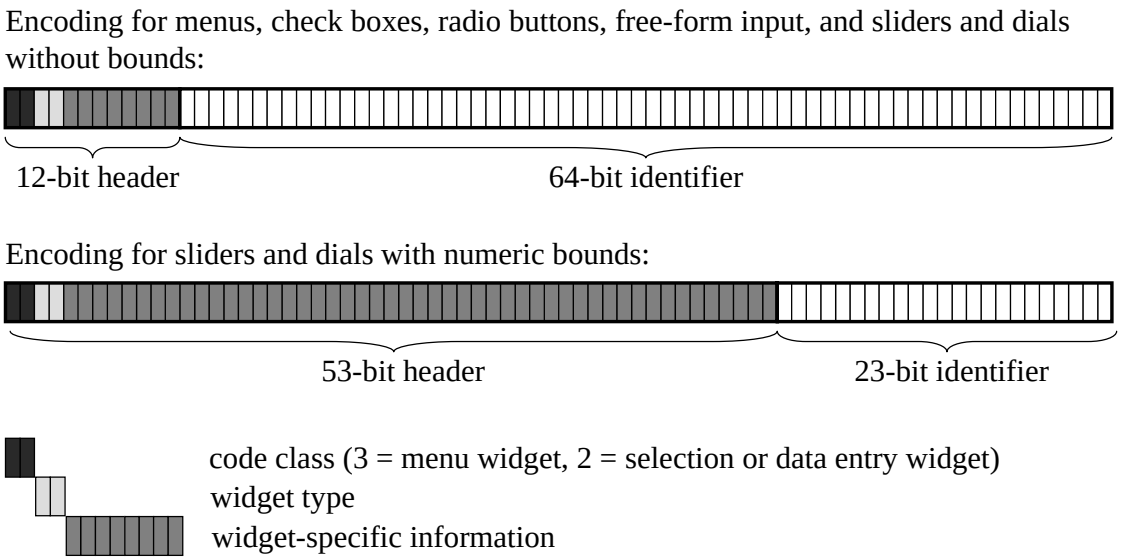


Figure 5.2: Bit-wise encoding of widget layout information within a 76-bit visual code.

Ideally, each widget would be universally unique. This would ensure that its semantics could be established without any other frame of reference. A person walking up to a poster with a printed widget would rely on the accompanying graphical and textual presentation to understand the semantics of the widget. A generic widget interpreter would suffice, since the unique identifier contained in the widget would then trigger a well-defined action. Unfortunately, the more complex widgets, like sliders with numeric bounds, leave less than 64 bits for the identifier.

Therefore, it is difficult to encode a globally unique identifier in the remaining bits. Either more bits need to be encoded in the tag or the application context has to be established explicitly, for example, by starting a special purpose application on the phone, instead of the generic widget interpreter.

5.4 Visual Code Menus

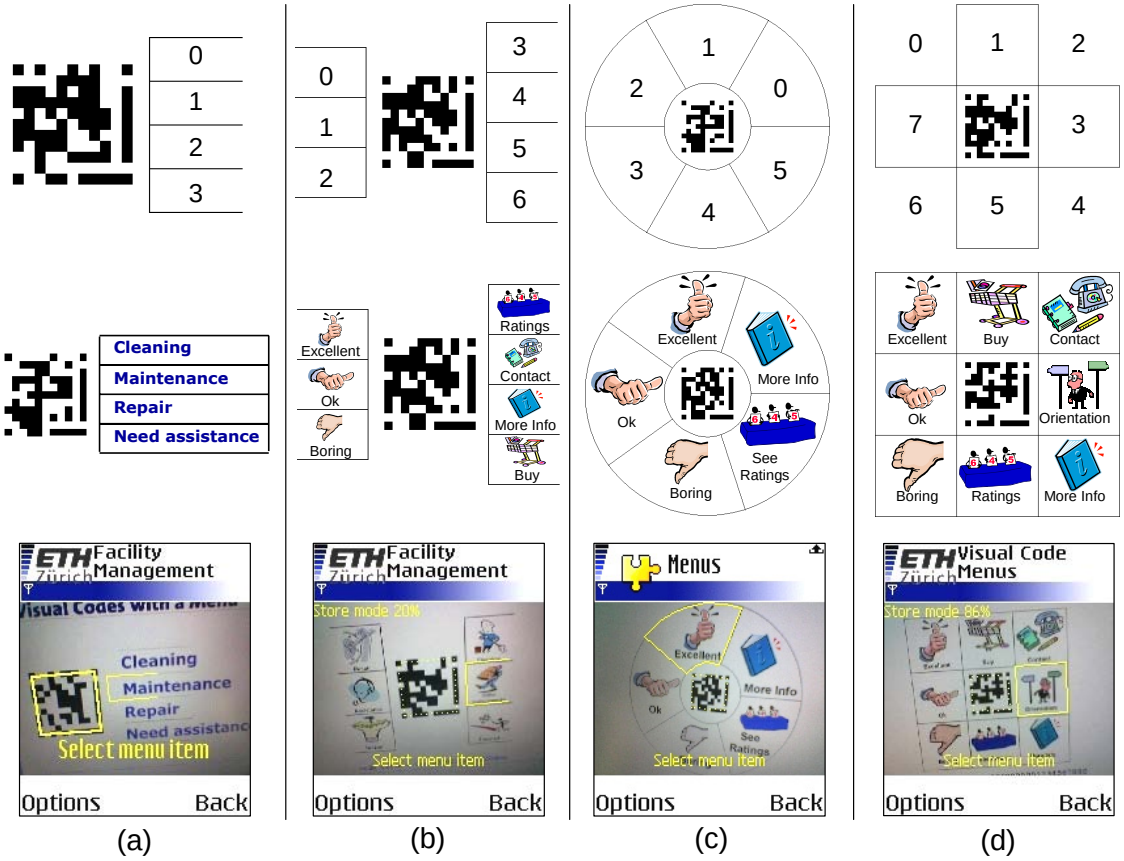


Figure 5.3: Visual code menus: (a) single vertical menu, (b) double vertical menu, (c) circular pie menu, and (d) square pie menu.

Menus offer the user a list of choices which are all visible at once. Menus are imperative widgets that trigger a function upon item selection. Visual code menus encode the menu type, the number of menu items, the height of each menu item, and a 64-bit identifier. There are four types of visual code menus as shown in Figure 5.3: (a) single vertical menus, (b) double vertical menus, (c) circular pie menus, and (d) square pie menus. Each type has a standard layout, which means that only very few parameters need to be specified in the code. Single vertical menus, for example, are always right-aligned to a visual code and vertically centered with the code.

The height of the menu items is given in units of the code coordinate system (*ccu*). One code bit element corresponds to a single unit. The location of the individual menu items can thus be computed in terms of code coordinates directly from the code. The mapping between code coordinates and pixel coordinates allows to

quickly compute the corresponding points on the screen, regardless of the perspective distortion of the code in the image. This way, the borders of the selected menu item can be superimposed over the camera image at the correct location.

The menu rendering component is generic in the sense that it just needs to interpret the menu layout that is encoded in the visual code. No information about the contents of the individual menu items is required. The menu item semantics are implicitly conveyed to the user through the camera image. The lower row of Figure 5.3 shows example screenshots of the generic rendering component in different applications. The currently selected item is highlighted with a yellow frame. When the user completes the interaction, the resulting parameters consist of the 64-bit identifier and the index number of the selected menu item. The assignment of index numbers to menu items is shown in the upper row of Figure 5.3.

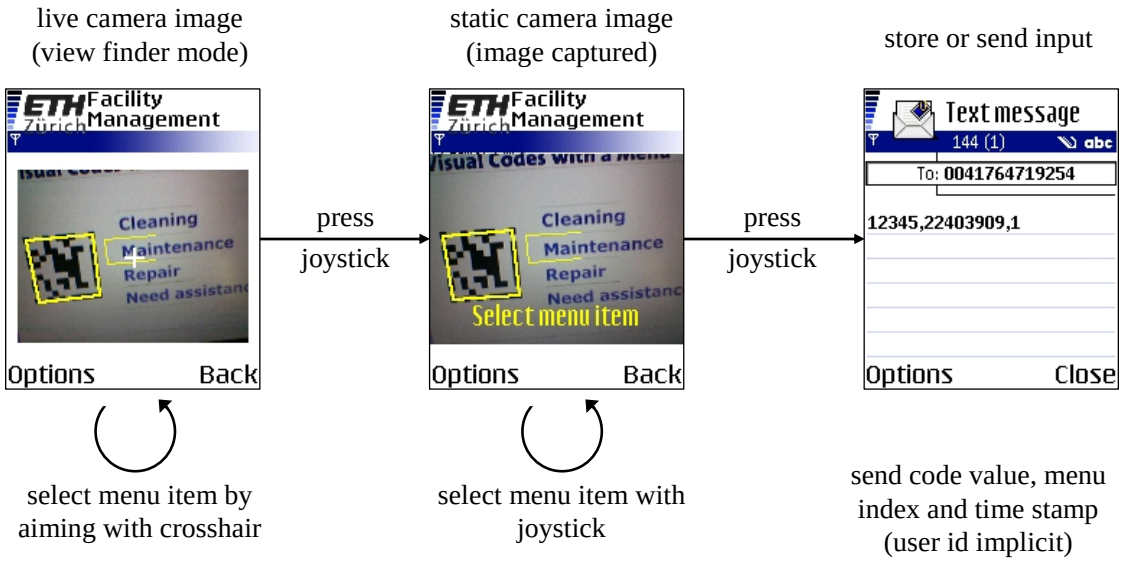


Figure 5.4: Interaction state machine for visual code menus.

For devices without pen-based input, such as the Symbian phones we used, interaction with visual code menus is a two-step process (see Figure 5.4). During view finder mode menu items will be selected depending on the focus point indicated by the cross-hair. Aiming at the visual code itself could trigger a special function, such as a help screen explaining the use of the widget (this feature is not implemented yet). Pressing the joystick button on the device will take a final high-quality picture and stop the view finder process. The camera image now freezes, the selection from the first step is retained, and the user has the opportunity to cycle through the menu selection using the joystick. One more click will submit the selected menu item.

We also implemented visual code menus on a smartphone running Pocket PC, which has a pen-based user interface. The device we used (a T-Mobile MDA III) does not provide a way to superimpose graphics over the camera image during view finder mode. We thus implemented the interaction in a slightly different way. The user first takes a picture of the menu as a whole. The picture then freezes and is shown on the display. All menu items are framed by a yellow border. The user can now select a menu item by tapping into the appropriate area.

For convenience, we have developed a Java command line tool to generate "empty" visual code menus and other types of widgets. Its input arguments are the type, number and size of the items, and the 64-bit identifier of the menu. Its output is a PNG image containing the code and the borders of the menu items (shown in the first row of Figure 5.3). The idea is that an application designer uses the image in any graphics editor of her choice to label the menu items with textual or graphical content (shown in the second row of Figure 5.3). A reference of the visual code widget creation tool is given in Appendix C.

5.4.1 Vertical Menus

In *vertical menus*, the menu items are stacked vertically upon each other. In *single vertical menus* (see column *a* in Figure 5.3), the items appear either to the left or to the right of the code. The table below shows the encoding of menu parameters for single vertical menus.

bits	**parameter**	**values**
75..74	code class	3 (visual code menus)
73..72	menu type	0 (single menu)
71	menu sub type	0 = left, 1 = right
70..68	item height	h = 2(x + 2) = 4..18 ccu
67..64	item count	n = x + 1 = 1..16
63..0	identifier	2^{64} values

Double vertical menus (see column *b* in Figure 5.3) have menu items on both sides, whose number can be specified independently.

bits	**parameter**	**values**
75..74	code class	3 (visual code menus)
73..72	menu type	1 (double menu)
71..70	item height	h = 2(x + 2) = 4..10 ccu
69..67	item count left	n_l = x + 1 = 1..8
66..64	item count right	n_r = x + 1 = 1..8
63..0	identifier	2^{64} values

5.4.2 Circular Pie Menus

In pie menus – also referred to as a "radial menus" – menu items are arranged as several "pie slices" around a center. As opposed to conventional linear menus, the distance to all menu items is the same and the area of an item is infinitely large, since selection only depends on the angle of the target point relative to the menu center. The location of pie menu items is easier to remember than the location of menu items in a linear list, which allows for fast access to commonly used items [41, 105]. These advantages also apply to *visual code pie* menus (shown in column *c* in Figure 5.3). The shape of the selected pie slice may be perspectively distorted depending on the orientation. The frame that is overlaid over the camera image is thus drawn using a polyline.

The parameters are encoded as:

bits	parameter	values
75..74	code class	3 (visual code menus)
73..72	menu type	3 (circular pie menu)
71..68	outer circle radius	r = 20 + 2x = 20..50 ccu
67..64	item count	n = x + 2 = 2..17
63..0	identifier	2^{64} values

5.4.3 Square Pie Menus

Square pie menus (see column *d* in Figure 5.3) are an alternative to circular pie menus. They have up to 8 items at fixed positions around the code. As a special feature, menu items can be individually included or removed from the menu. The generated pie menu at the top of column *d*, for example, only contains items, 1, 3, 5, and 7. On the device, only the available items are selectable. A bitmap of the available menu items is stored in the code:

bits	parameter	values
75..74	code class	3 (visual code menus)
73..72	menu type	2 (square pie menu)
71..64	bitmap of used menu items	1 = used, 0 = unused
63..0	identifier	2^{64} values

5.5 Selection and Data Entry Widgets

Check boxes and radio buttons are selection widgets. They enable the selection of multiple options or one of a set of mutually exclusive options. Selection widgets have state and change their visual appearance depending on that state. Unlike with menus, no immediate action is associated with selection widgets. Sliders, dials, and edit fields are data entry widgets. They enable the user to provide new information to an application, rather than merely selecting information from an existing list. Here, we present visual code equipped versions of these widgets. Again, the layout of each widget is described in the code value itself.

5.5.1 Check Boxes and Radio Buttons

Figure 5.5 shows *visual code check boxes* and *radio buttons*. In a check box, multiple options can be selected, whereas radio buttons only allow for the selection of a single choice. In both cases, the list of choices is vertically stacked and appears to the right of the code. Selections are indicated by superimposing a red cross or a red bullet, respectively, over the appropriate boxes and circles in the camera image. The box or circle that is currently focused is indicated by a light yellow frame. Pressing the selection button on the handheld device selects an option, pressing it again deselects it (in the case of check boxes).

The widget rendering component on the phone remembers the state of a check box widget or radio button widget using the widget identifier that is stored in the code. When the user targets the widget again, her choice of selected items will become visible again. The state will initially be empty (for radio buttons, the first option will be selected by default) – that is, when the widget is encountered for

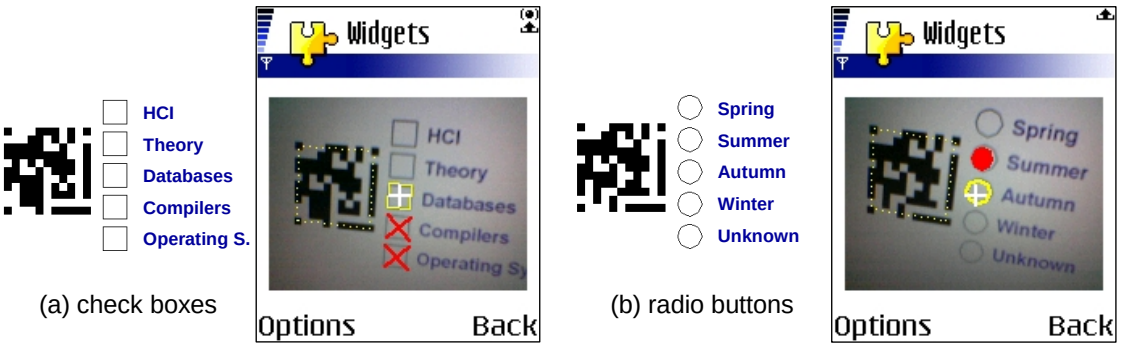


Figure 5.5: Visual code selection widgets: (a) check boxes and (b) radio buttons.

the first time – unless its global state is retrieved via some other communication channel.

Interaction with check boxes and radio buttons can completely take place in view finder mode. As the device is moved, the live camera image changes and the virtual overlays are updated in real time. Pressing the selection button changes the state of the current option, but the module stays in view finder mode. Pressing the joystick button freezes the current image. On the captured widget, further selections can be performed by moving the joystick up and down. This is consistent with the behavior of visual code menus.

The parameter encoding is defined as:

bits	**parameter**	**values**
75..74	code class	2 (visual code widgets)
73..72	widget type	1 (selection widget)
71	widget sub type	0 = check box
		1 = radio button
70..68	item height	h = 2(x + 2) = 4..18 ccu
67..64	item count	n = 1..16 for check boxes
		n = 2..17 for radio buttons
63..0	identifier	2^{64} values

5.5.2 Free-form Input

Free-form input fields have no direct counterpart in traditional graphical user interfaces. These widgets define a rectangular area next to a visual code in which the user can draw or write (with a pencil on paper). The user will then take a picture of the widget containing the drawing. The coordinate system mapping of the visual code is used for removing any perspective distortion from the captured image. The user gets a warning if the input area is not completely contained in the camera image. This is done by displaying red bars at edges which intersect the input area. As indicated with the arrow in Figure 5.6, the right edge intersects the input area; a red bar is thus displayed. The captured free-form input – unwarped and clipped to the bounds of the area – is then converted to a JPG image and stored on the device. It can later be submitted to a server using Bluetooth or the mobile phone network. The camera of the mobile device thus acts as a mobile scanner for selected areas of a printed document.

Free-form input widgets are encoded as:

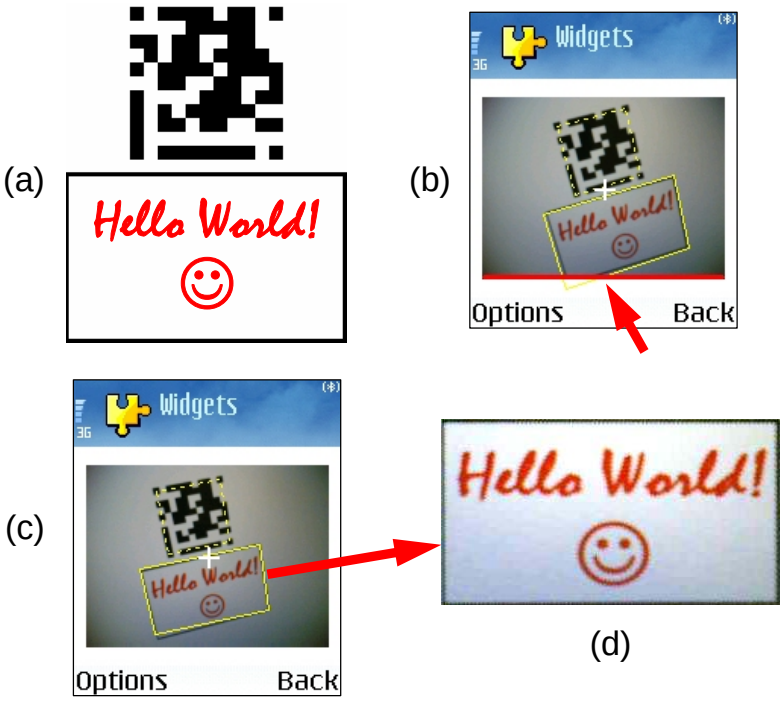


Figure 5.6: Free-form input widget: (a) area with free-hand drawing to be captured, (b) red bar at bottom indicates error, (c) perspectively distorted frame is captured, (d) captured area without perspective distortion.

bits	parameter	values
75..74	code class	2 (visual code widgets)
73..72	widget type	2 (free-form input or button)
71..70	code position	0 = left, 1 = top
69..66	frame width	w = 4 + 4 x = 4..64 ccu
65..62	frame height	h = 4 + 4 x = 4..64 ccu
61..0	identifier	2^{62} values

5.5.3 Sliders

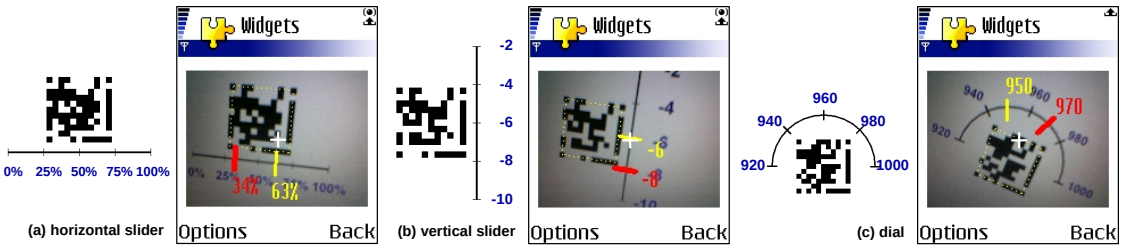


Figure 5.7: Visual code data entry widgets: (a) horizontal slider, (b) vertical slider, and (c) dial.

Unlike free-form input widgets, which provide "unbounded" input, *sliders* are "bounded" data entry widgets. The slider can be moved across a certain range, the selected value being proportional to the current slider position. As shown in Figure 5.7 there are horizontal and vertical sliders. Input can either be continuous

or discrete. For a continuous slider, the sliding thumb can take any position on the scale. For a discrete slider, the thumb can only be placed at the positions of the tick marks that are indicated on the scale. There are four different options for the kind of output that is given as feedback: no textual output, percentage value (increasing or decreasing), and numeric output using the bounds and step width given in the code. Sliders are encoded as:

bits	parameter	values
75..74	code class	2 (visual code widgets)
73..72	widget type	0 (slider)
71	direction	0 = horizontal, 1 = vertical
70	mode	0 = continuous, 1 = discrete
69..68	output	0 = none, 1 = percent (inc)
		2 = percent (dec), 3 = numeric
67..62	tick distance	h = 2x + 2 = 2..128 ccu
61..55	tick count	n = x + 2 = 2..129

For the output modes 0, 1, and 2, the remaining bits store a 55-bit identifier. For numeric output (output mode 3), the remaining bits are structured as:

bits	parameter	values
54..39	bound	16-bit signed integer
38..23	step	16-bit signed integer
22..0	identifier	2^{23} values

Visual code sliders are most conveniently used in view finder mode. The current position is indicated with a light yellow slider thumb ("63%" and "-6" in Figure 5.7). The yellow thumb always immediately follows the current cursor position and moves along the scale as the camera focus changes. When the user presses the selection button, the actual value of the widget changes and the thick red slider is moved to the current slider position. The position of the red slider ("34%" and "-8") determines the value that is returned by the widget as the final result of the interaction.

As an alternative to absolute sliders with a printed scale, we have defined relative sliders. This category of sliders uses the recognition algorithm's relative camera movement detection. It does not require the presence of a visual code in the camera's field of view, but detects device movements by looking for changes in the background image. This gives more freedom in moving the device to continuously adapt parameters. We have used this kind of sliders in a large-scale display application [20].

5.5.4 Further Data Entry Widgets

Other widgets we have explored are text entry widgets, dials, and submission buttons. Clicking a *text entry widget* opens a standard GUI single-line or multi-line edit control, depending on the widget options. The widget rendering component stores the state of the widget, i.e. the already entered text. It also automatically sets the input mode to numeric or alphanumeric mode.

Visual code dials (see *c* in Figure 5.7) are a kind of circular slider that is controlled by the amount of rotation of the phone relative to the code. For ease of

use, the scale is mapped to a semi circle above the code, instead of using the full 360°. The slider is positioned in the middle of the scale when the phone is held upright. It is positioned at the right end of the scale when the phone is rotated right by 90° and at the left end when it is rotated left by 90°. The interaction proceeds in the same way as for linear sliders.

Dials are encoded as:

bits	**parameter**	**values**
75..74	code class	2 (visual code widgets)
73..72	widget type	3 (dial)
71	mode	0 = continuous, 1 = discrete
70..69	output	0 = none, 1 = percent (inc)
		2 = percent (dec), 3 = numeric
68..63	tick count	n = x + 2 = 2..65
		for output = 0, 1, 2
62..0	identifier	2^{63} values
		for output = 3
62..47	bound	16-bit signed integer
46..31	step	16-bit signed integer
30..0	identifier	2^{31} values

Submission buttons belong to the imperative widget category. They are necessary to actually submit the data that was input or modified before, using the data entry and selection widgets. The widget encodes the method used for transporting the data, method-specific parameters, such as the target address, and a prefix of identifiers it is relevant for. All data that was collected from widgets with the specified prefix will be submitted to the given destination.

bits	**parameter**	**values**
75..74	code class	2 (visual code widgets)
73..72	widget type	2 (free-form or button)
71..70	widget sub type	2 = submit button
69..67	method	0 = RFCOMM
69..67	method	1 = HTTP POST
	method	2 = HTTP GET
	method	3 = SMS, 4..7 unused
66..65	prefix length	l = x + 1 = 1..4 bytes
64	flag	method-specific
63..56-32	prefix	1-4 bytes
55-31..0	address	4-7 bytes

Relative movement detection is encoded as shown below. This widget type operates without a visual code in the camera view during movement detection. The visual code is only sampled initially to retrieve its value. This value is then reported together with the movement updates as (Δx, Δy, $\Delta\alpha$, value), for example via Bluetooth. This widget type is used in the large display interaction example applications.

bits	parameter	values
75..74	code class	2 (visual code widgets)
73..72	widget type	2 (free-form, button, or relative movement)
71..70	widget sub type	3 = relative movement detection
69..64	reserved	0
63..0	identifier	2^{64} values

5.6 Applications

Visual code widgets can be used in a wide range of application areas. We outline two examples here: facility management and interactive television.

In the facility management and maintenance work scenario, mobile workers have a camera-equipped smartphone. At a location, machine, or other object, they find a visual code menu that contains a menu item for each step in the procedure they need to perform, such as "cleaning," "repair," or "routine maintenance." The basic facility management application we have developed just stores the selected index, the menu identifier, and a time stamp. Upon request, the application sends the stored data via SMS to a predefined phone number. Beyond this, it would be beneficial to integrate services, such as ordering spare parts, calling for assistance (based on the identifier the call could automatically be routed to an expert), and getting background information on an object. To differentiate between different phone functions – such as storing (index, identifier, and timestamp), calling, text messaging, and browsing – the identifier could be further structured: If the identifier indicates storing, the data would be stored internally; if it indicates browsing, the identifier and selected index would be used as arguments for a backend server to retrieve the actual content. A standard set of icons for these different functions could indicate to the user what happens when a menu item is selected, showing, for example, a disk icon for storing and a phone icon for calling. This would, of course, use up most of the capacity of the remaining 64 bits of the identifier.

In many television programs, like in breaks of sports event broadcasts, there are quizzes in which users can select their answer from a menu of choices and send an SMS to a certain phone number. In return, they can win tickets or other prizes. Instead of requiring the user to quickly jot down a phone number that is only shortly displayed, the camera phone can be used as an interaction device for the television program (see Figure 5.8): The television screen shows a visual code menu that is captured by the user with a single click. The screen contents are then frozen on the handheld screen and the menu is available for interaction. As the television program continues, the user can take her time to cycle through the menu choices and select an answer. The result is then automatically sent as an SMS to a phone number that is encoded in the menu identifier.

The described widgets are generally useful for paper-based interaction. An example are surveys and opinion polls which could be carried out with paper forms that contain visual code widgets. The widgets are also suitable for interaction with large-scale displays for which traditional mouse and keyboard input is not feasible. This is particularly relevant for displays at public places that do not provide an input device on their own, but rely on devices that users carry with them [10].

Mobile annotation services are an active area of research. The main problem is the limited input capacity of mobile devices. It is very inconvenient to annotate

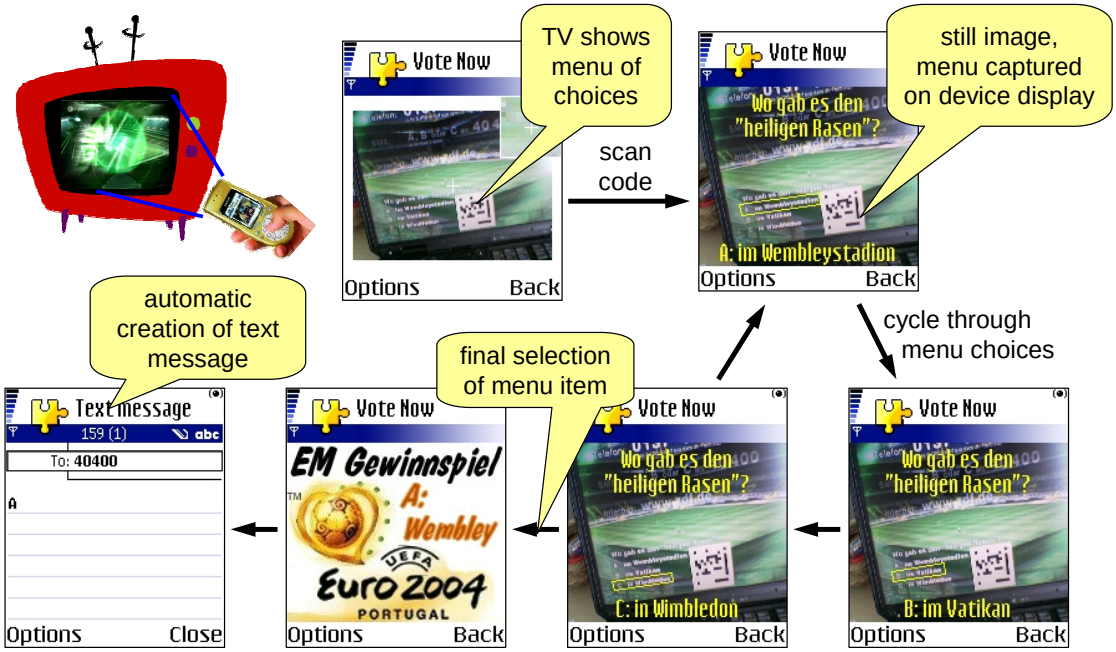


Figure 5.8: Visual code menus for interactive television.

an object by using the tiny keyboard on the handheld device. Most users just do not take the effort to create annotations this way. Using visual code widgets, it becomes much easier to enter selections, ratings, and free-form annotations.

5.7 Summary

Visual code widgets are a versatile mechanism to interact with elements in printed documents or on electronic screens using mobile devices. Visual code widgets could become important building blocks for marker-based interaction. For programmers, they encapsulate generic functionality and thus simplify application development. For users, they provide clear affordances and predictable behavior that is consistent across different applications.

A key feature of the proposed encoding scheme is that no initial communication is necessary. The semantics are implicitly available and conveyed to the user through the captured camera image that is augmented by virtual widget state information. A single generic widget interpreter and rendering component is sufficient to recognize all of the discussed widgets. A creation tool allows for the simple generation of different types of widgets.

Widgets could also be completely virtualized, i.e. all of their components except for the visual code itself could be virtually overlaid. This would be interesting for applications for which printing space is scarce or needs to be flexibly used, such as with virtually enhanced product packaging. For some widget types, like menus, this would require that the semantics are explicitly provided to the user.

Chapter 6

Interaction with Large Public Displays

In this chapter, we propose and evaluate direct manipulation techniques for large public displays.[1] We introduce two interaction techniques that are based on camera phones, visual codes [169], and optical movement detection. The *point & shoot* technique uses visual codes to set up an absolute coordinate system on the display surface, instead of tagging individual objects on the screen. The *sweep* technique turns the phone into an optical mouse with multiple degrees of freedom and allows interaction without having to point the camera at the display. Prototypes of these interactions have been implemented and evaluated. They serve as a proof of concept, provide a performance baseline, and give insights to guide future research and development.

6.1 Introduction

Large-scale electronic displays are increasingly found in public places like airports, train stations, shopping malls, and museums. Typically, their content – like current train schedules, information on local events, or advertisements on products to be sold – is related to the environment in which they are situated. Unfortunately, most of today's public electronic displays are not interactive, making it difficult to capture interesting information and impossible to influence the display's content. In addition, large displays in public places are often inaccessible for direct touch-based interaction, since they need to be protected from vandalism, and installing dedicated hardware for interaction can be prohibitively expensive.

Electronic large displays can be categorized into three different classes of social settings, in which they are embedded: personal, semi-public, and public. *Personal* large displays allow a single user to visualize and quickly process large amounts of information [18]. *Semi-public* large displays are situated in closed environments such as an office building or a meeting room, where a small community of people

[1] Parts of this chapter are the result of collaborative work with Rafael Ballagas, RWTH Aachen, and Jennifer G. Sheridan, Lancaster University. These parts have been published in [10, 11, 12, 177]. The interaction techniques and the applications have been designed and developed by myself. The design of the usability study and the device classification is mostly the effort of Rafael Ballagas and Jennifer G. Sheridan. The design considerations have emerged from joint discussions.

interact regularly (see Figure 6.1, left). Typically, semi-public displays are used collaboratively in single display groupware [23] applications. Large *public* displays are commonly placed in publicly accessible locations, usually in environments with high pedestrian traffic and extended wait times such as train stations, airports, or theme parks (see Figure 6.1, right). In this work, we focus on large public displays that are not accessible for direct touch-based interaction.

Figure 6.1: (Left) Semi-public displays in the *iRoom*, an interactive conference room at Stanford University. (Right) A large public display in a subway stop in Vienna, Austria (from [12]).

As an example of a non-interactive large public display, we observed the content of an "infoscreen" at a subway station in Bonn, Germany. It is mounted at a wall on the opposite side of the pedestrian platform and viewable across the railway tracks. During the observation period it featured the following repeated content:

- advertisements
- current movies in local cinemas
- museum and concert hall information (exhibitions, concerts)
- city event information (city festival, "rock'n'roll revival party")
- city weather forecast
- famous quotes
- short comic cartoons
- fashion and style tips
- health and fitness tips
- charity organization, donation requests
- headlines on prominent people
- news headlines (5 lines of text with 20 characters each)
- stock quotes

- historic events ("this happened 50 years ago")

This is a wide range of content and not only consists of advertisements. It appeals to a diverse audience – the "community" of city inhabitants and visitors who use local transportation – and it is adapted to the locality by showing locally significant information. The content is suited to foster the city community and to promote awareness of events that relate to it. Interestingly, at the time of observation, few people seemed to look at the display and pay attention to its content. Maybe this lack of attention is rooted in a kind of information overflow. People are trained at ignoring much of the information that surrounds them in public space, like advertisement posters. Probably, people expect advertisements to be the only content of electronic displays as well. Notably, the observed screen is context-aware in that it shows a line of text saying "attention, train approaching," shortly before a train enters the station on the track in front of the screen. Yet, there is no way for users to interact with the screen, to adapt its content to their interests, to vote on the content, or just to capture interesting content for later replay.

Camera-equipped mobile phones and similar personal devices open up new possibilities in this domain, since the camera provides a powerful input channel and the phone can spontaneously connect to the situated display with wireless technologies, such as Bluetooth. Additionally, people are familiar and comfortable with using their own devices and they usually have their mobile phones with them. Potential applications include interactive art, collaborative games, bulletin boards, and interactive advertising. Applications will be discussed in more detail in Section 6.3.

6.1.1 Design Considerations

In [12], Ballagas et al. have identified a number of design considerations that are specific to interaction with large public displays. They include serendipity (requiring as little effort as possible to initiate the interaction), multi-user simultaneity (allowing multiple users to simultaneously interact with a single display), dexterity (requiring just a single hand for operation, since users might carry bags), and privacy (since complete strangers can observe the interaction).

- **Serendipity** corresponds to users' ability to spontaneously interact with a large display. High serendipity means a low threshold of use with any arbitrary display. High serendipity prohibits preliminary setup and registration steps.

- **Learnability and memorability** denotes the amount of effort required by the user to learn and remember the interaction.

- **Dexterity** refers to the physical difficulty of the interaction, including both hand posture and how many hands are required for operation. This is an important consideration for public environments where users might need their hands to carry bags or other personal belongings.

- **Portability** refers to users' ability to transport the tools needed to interact with the display. High portability systems require only what users typically carry with them; low portability systems require users to carry bulky equipment.

- **Interruptability** means that interaction has to be designed with interruption in mind. Many interactions with large public displays tend to be short-term, and are often interrupted by external events (such as a bus arriving). Interruptability must be supported by potential interaction techniques.

- **Multi-user compatibility** corresponds to the ability of an interaction technique to support multiple users simultaneously and to arbitrate access to the display. This is important because large public displays are typically surrounded by many people arriving at varying times.

- **Social acceptability** refers to the acceptability of an interaction technique in the presence of others, who passively observe the interaction. The interaction technique might be disturbing to observers, embarrassing to the user. Vice versa, it might raise the social status of a person.

- **Intentional vs unintentional interaction** denotes two modes of interaction initiation. The first case, on which we focus with our interaction techniques, requires the user to actively initiate the interaction. In the second case, the large display might initiate interaction upon sensing the presence of a user.

- **Interaction distance** is dependent on the placement of a large display in public space. It determines the physical accessibility and the required visual faculty of the users.

- **Physical security** is required, since large public display systems must be protected from vandalism and theft. For example, in the subway station in Figure 6.1, the projector is protected in steel casing, and the display area is located across the rail tracks, making it inaccessible to vandals.

- **Information security and privacy** refers to the degree of security and privacy that a user can expect when interacting with a large display in public. The interaction technique has to ensure, for example, that sensitive data, like names or phone numbers, are not accidentally shown on the large public display.

- **Maintenance** refers to the amount of regular service the system needs to stay operational and maintain an appearance that attracts user interaction.

- **Sanitation** corresponds to the cleanliness and health considerations associated with an interaction technique. The physical condition of the display will affect interaction. In some cases, a well-used display can indicate popularity and have a positive effect on usage, such as with arcade games. Conversely, a dirty public kiosk will have a negative effect, likely repelling potential users.

We use these design considerations in our following discussion of different existing interaction techniques for large public displays.

6.1.2 Existing Interaction Techniques for Large Displays

Direct touch-based interaction with the display surface is the most widely used interaction paradigm for large-scale displays. Touching involves users' fingers, special active pens, or other passive pointing devices. The touched point can be determined with capacitive surfaces, with optical recognition as used in *Barehands* [168] or *Digital Vision Touch (DViT)*,[2] or it can be done with active ultrasonic pens such as *Mimio Virtual Ink*.[3] Touch based interaction has many advantages in terms of usability. It realizes a direct one-to-one mapping between the point of physical interaction and virtual on-screen objects. However, this incurs limitations in spatial scalability. If a large display covers the whole wall of a room, which is expected in the future for technologies such as OLED [188], the display will extend beyond human reach. In addition, direct touch requires a very short distance of interaction, at which only a small part of the display can be conveniently observed and other parts appear at strong perspective distortions. Interaction would thus require a constant change between close-distance interaction and far-distance viewing, to get an overview of the broader context. Touch-based interaction presumes physically accessible displays, which lowers physical security. If users's fingers are used for interaction, serendipity is very high, but sanitation is low. Touch-based input has only limited multi-user compatibility, since standard capacitive coupling cannot distinguish between multiple users. The *DiamondTouch* [58] technology addresses this problem for stationary settings. Still, with close-distance interaction a single user obstructs a large part of the view of other users.

Multimodal gesture and speech interaction provides an alternative to direct touch-based interaction. Yet these technologies are not well-suited for the public domain. It is difficult to distinguish between intentional interaction and random visual and audible noise. This problem is especially severe in public environments, where multiple people are around. The Barehands [168] project simplifies the gesture recognition task by limiting the recognition range to a very small area close to the display surface. Yet this solution has the same problems as other close-distance interaction techniques. Moreover, special gestures and speech commands are hard to learn and memorize. Speech is not always socially acceptable, since it has the potential to disturb others. It also scores badly in terms of privacy.

Sometimes, stationary input devices are installed with public displays. However, this solution is problematic in terms of cost, physical security, sanitation, maintenance, and multi-user compatibility.

An interaction paradigm that resolves many of these issues is to employ the personal devices that users carry with them for large display interaction. Most of today's PDAs and mobile phones have short distance wireless communication capabilities, which can be used for connecting to nearby large displays. They score high in terms of interruptability, since the current interaction state can be stored on the device. If the display allows multiple mobile devices to be connected at once, they are also multi-user compatible. They can be used at varying interaction distances, which means that users can interact with the display without obstructing the view of others. The personal device interaction paradigm is also physically secure and has low maintenance costs. Personal devices allow for high information security

[2] www.smarttech.com/dvit

[3] www.mimio.com

and privacy, since sensitive information can be restricted for presentation on the device and personal devices can execute cryptographic algorithms to authenticate users to the display or vice versa. Finally, users are comfortable and familiar with using their own devices.

6.1.3 Camera Phones as Interaction Devices for Large Public Displays

Mobile phones are well suited as interaction devices for large public displays. They are ubiquitously available and are widely adopted. Camera phones are continuously gaining market share.[4] In terms of usage patterns, users carry camera phones with them throughout the day and also in public places. Camera phones are smaller than PDAs and are designed for single handed input, which means that they exhibit higher dexterity and portability. We are thus adopting camera phones for enabling direct manipulation capabilities for large public displays.

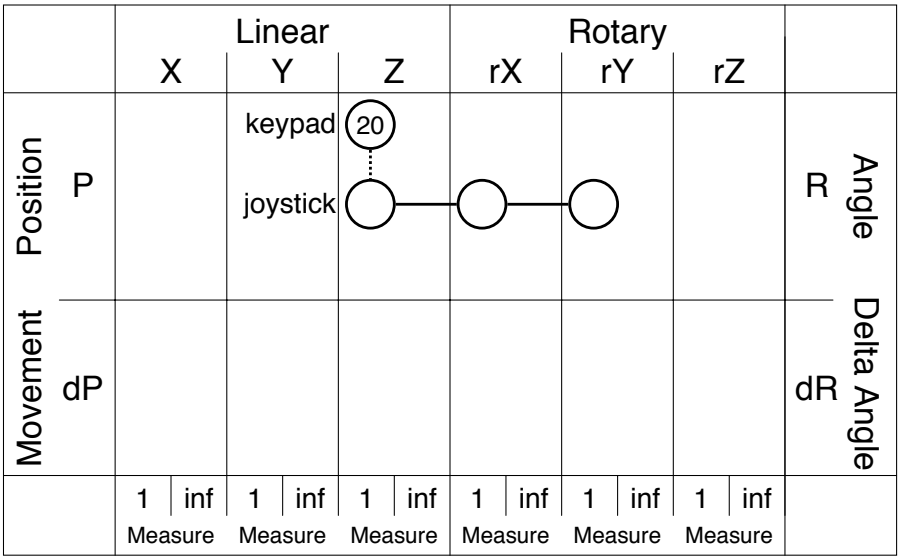


Figure 6.2: Naive classification of a joystick-equipped mobile phone as an input device following the design space presented in [42].

To implement our interaction techniques, we used camera phones that run the Symbian operating system. The Nokia 6600 camera phone that we initially used can be classified as an input device that has a 20 button keyboard combined with a joystick. The joystick has five possible states (up, down, left, right, or pressed). Figure 6.2 shows its properties in a Card [42] style design space. Additionally, the low-resolution integrated camera found in today's mobile phones can function as a special optical sensor and therefore provide an additional input channel. Our direct manipulation techniques for large public displays extend the design space, as shown in Figure 6.6 and discussed below.

[4]Of the 625 million mobile phones that are expected to be sold in 2004, more than a quarter have built-in cameras (in Japan 60%). Sources: CNET News, iSupply, Wikipedia.

6.2 Phone-Based Interaction Techniques

We present two complementary interaction techniques for camera phones: one for absolute positioning, which is based on visual code sensing, and one for relative positioning based on relative movement detection. Both techniques enable selection, dragging, and rotation of objects on a large display. These techniques can be combined into *compound targeting*, where the technique that is used is chosen by the user, depending on factors such as the distance between the user and the display or on the current task. Different tasks require different pointing precisions and different distances over which on-screen objects need to be dragged.

6.2.1 Relative Positioning: Sweep

The *sweep* technique utilizes visual movement detection, which involves rapidly sampling successive images from a camera phone and sequentially comparing them to determine relative motion in the (x, y, θ) dimensions, as described in detail in Section 3.3.3. Movement detection enables the camera to be used as a three degrees of freedom (DOF) input device. In particular, we use it here as an optical mouse for a nearby large display (see Figure 6.3). No visual code has to be present in the view of the camera, since the relative movement detection solely relies on the comparison of camera images. Also, no additional hardware – like an accelerometer – needs to be attached to the phone. In our implementation, relative movement detection is performed directly on the phone rather than on the computer driving the display. One advantage of this strategy is scalability; the interaction technique easily scales to a high number of users. A disadvantage however, is the high latency with current hardware (about 200 ms with a Nokia 6600) that occurs when calculating the (x, y, θ) changes from successive images. Yet mobile computing trends indicate that in the not too distant future mobile phones will have the processing power necessary to create a fluid interaction experience.

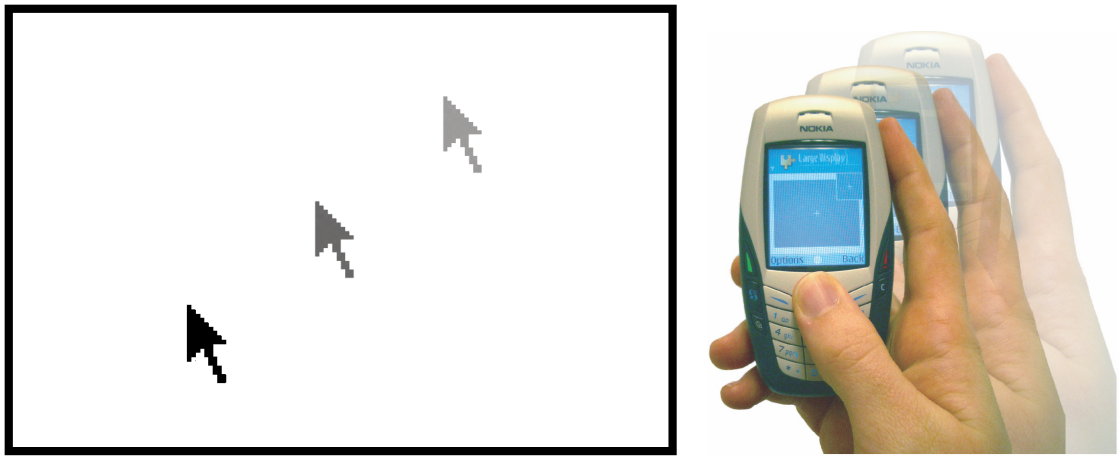


Figure 6.3: The sweep technique turns the camera phone into an optical mouse and can be used to control a cursor on a nearby large display.

To invoke the *sweep* function, users vertically push and hold the joystick button, which acts as a clutch, to indicate to the system that they are actively controlling

the cursor. Then they wave the phone in the air to control the (x, y, θ) input. Users can release the clutch button to reposition their arm, which is similar to the way a mouse can be lifted to be repositioned on a desktop surface. This means that the camera need not be pointed directly at the display but can be pointed at the floor to allow users a more comfortable arm posture. In the *sweep* mode, the user can ignore the display on the phone and focus attention on the large display to observe the cursor movement.

6.2.2 Absolute Positioning: Point & Shoot

The *point & shoot* interaction technique is illustrated in Figure 6.4. Users aim the camera phone at the target on the large display. The large display contents appear on the camera phone screen, which acts as a view finder. As users move the phone, the screen is continuously updated with a live camera image. Aiming is facilitated by a crosshair in the center of the device screen and a magnification of the area around the center. The magnified part is shown in the upper right corner of the phone screen. In *point & shoot* interactions, the user's locus of attention switches between the phone screen and the large display.

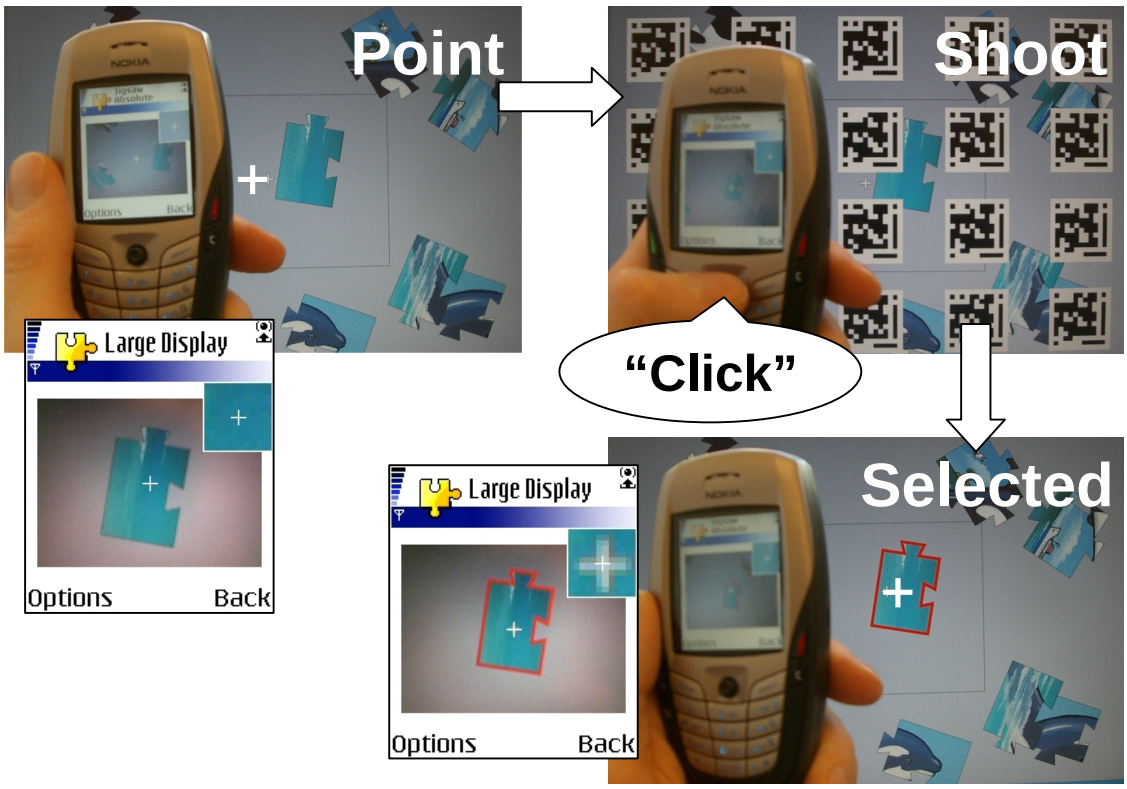


Figure 6.4: Point & shoot technique: (Left) The phone display is used to aim at a puzzle piece on a large display. (Middle) Pressing the joystick indicates *selection* and a visual code grid flashes on the large display to compute the target coordinates. (Right) The grid disappears and the targeted piece highlights to indicate successful selection.

Point & shoot is triggered by horizontally pushing and releasing the joystick button ("shooting"). When users shoot, a grid of visual codes flashes on large

display, as seen in the middle Figure 6.4. The grid is then used to compute the precise target point. A "selection" is then issued on the large display at the target point.

Absolute positioning is accomplished by using the coordinate systems of the grid of visual codes that are located at known positions on the large display. As soon as *point & shoot* is triggered, this event is reported via Bluetooth to the large display. The display reacts by showing the visual code grid. The phone now recognizes the visual codes and reports their values and the target point in the coordinates of the different code coordinate systems to the large display. Upon receiving this data, the display hides the grid and transforms the coordinates into pixel coordinates on the large display. The operation at the target pixel, such as a selection, can then be executed. The minimum requirement is that at least one visual code must be in the camera image during selection. In our experience, during interaction via Bluetooth both latency and jitter (variance of delay) are very low.

With the current implementation, the grid remains visible for about 0.5 seconds. In future implementations, the grid might be displayed for a shorter amount of time and would thus be barely visible to human users. Alternatively, future display technologies may allow the codes to be displayed in infrared so that they are recognizable by the camera, but invisible to humans.

6.2.3 Input Device Classification

Figure 6.5: Phone input interaction: *point & shoot* is mapped to horizontal joystick push-and-release, *sweep* is mapped to vertical push-and-hold.

To enable selection, dragging, and rotation, the *point & shoot* and *sweep* techniques are mapped to the phone's joystick button as shown in Figure 6.5: Absolute movement (*point & shoot*) is invoked by pushing the joystick in a horizontal direction (see Figure 6.5). Pushing it to the left and releasing it again triggers absolute movement of the cursor only, whereas pushing and releasing it to the right in addition also drags the object that is currently located underneath the cursor to the new cursor position. Relative movement (*sweep*) is invoked by pushing the joystick in a vertical direction. Holding it upwards invokes relative cursor movement only, whereas holding it downwards additionally drags the current object. Relative

dragging includes rotation of on-screen objects. It is accomplished by rotating the phone around the z-axis. Absolute dragging includes rotation as well. Holding the phone upright while "shooting" just moves the current object to the target point without changing its current rotation state. Holding it rotated to the left/right rotates the object to the left/right correspondingly, i.e., by the amount the phone is rotated relative to the upright position. Pressing the joystick key inwards (along the z-axis into the phone) is used for explicit selection. Mapping the interaction techniques to the joystick button in this way preserves simple one-handed operation and does not impinge on dexterity as users are not required to reposition their fingers to different buttons. Our initial hypothesis was that users tend to use absolute positioning to cover large distances and use relative movement for shorter distances or dragging tasks that require more precise control.

As shown in the classification in Figure 6.6, the interaction properties of the device become richer by adding the camera as a relative and absolute movement sensor. Given a camera resolution that is fine enough, in principle arbitrarily small relative motion updates can be sensed. The three horizontally connected circles labeled *sweep* correspond to the 3 DOF and map to the (x, y, θ) dimensions. Although it is possible to also detect relative Z movement and relative X and Y rotation, we excluded it here in order to focus on the most important aspects. In our implementation, relative rotation around the X axis (dR:rX) is equivalent to linear Y motion and relative rotation around the Y axis (dR:rY) is equivalent to linear X motion. This means that for the *sweep* technique, bending the wrist is equivalent to moving the whole arm. (Extension and flexion for linear Y movement, pronation and supination for linear X movement, as defined in Section 3.5 of [107].) In addition, relative Z movement (dP:Z) could be mapped to a further input dimension. The three horizontally connected circles labeled *point & shoot* represent absolute position sensing. It provides the X and Y position and the state of rotation around the Z axis.

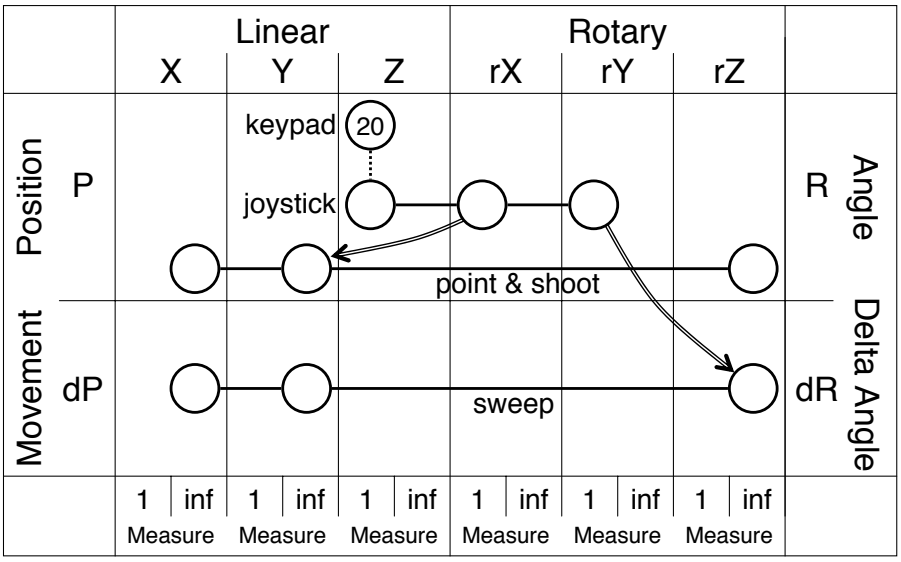


Figure 6.6: Card design space classification [42] of our camera-based interaction techniques.

When multiple users interact with a large display simultaneously, multiple cursors are required. This can be achieved by shaping or coloring the cursors differently. The cursor color could match the shape and color of the cursor on the mobile phone to help users identify which large display cursor they are controlling. Additionally, to help users locate their respective cursor on the large screen, a press on a special phone button could shortly flash or highlight their cursor.

6.2.4 Designing for Serendipity

In addition to establishing a coordinate plane, we use visual codes to encode the public display's Bluetooth address information thus enabling a communications channel to be rapidly established between the mobile phone and the large display, similar to [186]. Users merely take a picture of a visual code associated with the display and the phone will automatically connect to send (x, y, θ, text) information via Bluetooth. The latency to establish the channel is fairly low and the amount of jitter (variance of delay) during interaction is negligible.

The same connection can be used to authenticate the user, to send user profiles for adapting the content shown on the large display to personal preferences, to transfer sensitive information to the personal display, and to copy and store information and the current state of interaction on the phone.

This creates a low threshold of use and allows for highly serendipitous interactions. In order to do visual code recognition and visual movement detection, our proposed device interactions require that users install special software on their mobile phone. However, this software could potentially be installed during manufacturing, via the mobile phone network using over-the-air provisioning, or users could retrieve it via Bluetooth directly from the computer that is driving the display. Fortunately, this software only needs to be installed once and therefore only slightly increases the threshold of use for first time users.

6.3 Application Areas

Application areas for interactive large public displays include collaborative games, interactive art, digital bulletin boards, and interactive advertising. In the PhotoPhone Entertainment project [201], for example, camera phones were explored to play games in public places, like bus stops and public squares. PhotoPhone uses large public displays available at these places as output. Photos are analyzed or modified by a remote server and the results are sent back to the phone again. However, our direct manipulation approach operates directly on the phones and thus enables a more interactive gaming experience between users and the public display. Other application areas include interactive art installations [61] or public digital bulletin boards like those envisioned in the Plasma Poster Network [43] and Web Wall [68], which provide an outlet for communities to share and disseminate news, announcements, and ideas.

6.3.1 Jigsaw Puzzle

As a demo application for the interaction techniques, we incorporated phone input into an existing jigsaw puzzle application (shown in Figure 6.7). The puzzle pieces

can be individually selected, moved, and rotated. As soon as two compatible pieces are brought in close proximity and their amount of rotation matches, they snap together and form a larger piece.

Figure 6.7: A user interacting with the jigsaw puzzle game via a camera phone.

The demo application implements both of the interaction techniques described above. The techniques are mapped to the joystick button as shown in Figure 6.5 and Table 6.1. Again, the joystick is only used to trigger the interactions. The actual effect depends on the point that was targeted (in the case of *point & shoot*) or on the movement of the phone relative to the background (in the case of *sweep*).

Input	**Interaction**	**Application function**
push-and-release left	*point & shoot* select	move cursor to destination point and select puzzle piece (if any)
push-and-release right	*point & shoot* drag+rotate	move the selected puzzle piece to new destination and simultaneously rotate it by the amount the phone is rotated from the upright position
push-and-hold up	*sweep* select	continuously move cursor in the direction in which the phone is moved
push-and-hold down	*sweep* drag+rotate	continuously drag and rotate the selected puzzle piece in the indicated direction

Table 6.1: Mapping of input action to interaction technique and application function.

6.3.2 3-D Menu Hierarchy

In this application, a number of menu items are shown in a 3-D model. The main menu items are placed on an invisible sphere in equal distance, as shown in Figure 6.8. Each main menu item is associated with a visual code. The model can be rotated around the x, y, and z axes using the *sweep* technique. Individual menu items can be selected, which invokes submenus. Information that is associated with a menu item will appear on the phone's screen upon selection. The user can thus copy and store information of interest on the mobile phone.

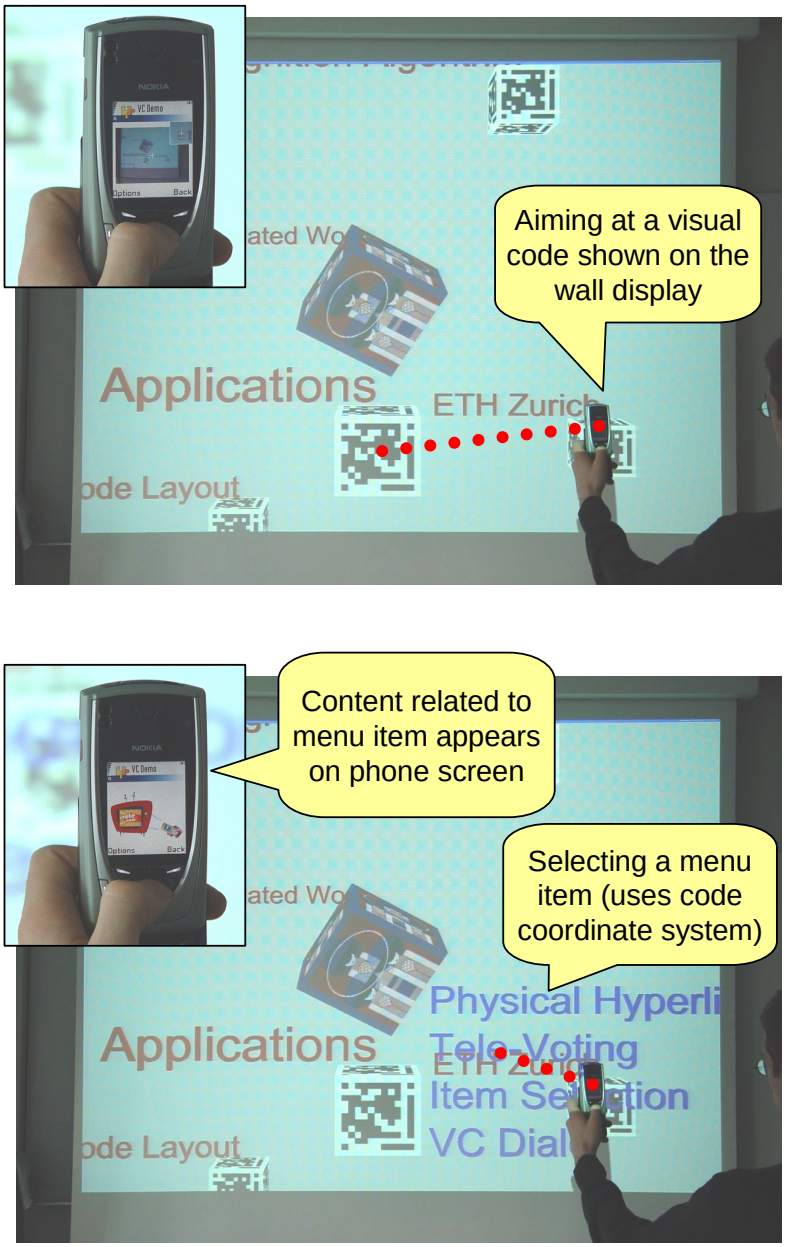


Figure 6.8: 3-D model of a hierarchical menu structure that can be manipulated using the camera phone.

The model is implemented using Java 3D [197]. The screen model's movement is controlled in the following way: Phone movement in horizontal direction results in a rotation of the sphere around the y-axis, vertical motion results in a rotation

around the x-axis, and rotating the phone results in sphere rotation around the z-axis. Because of the relatively low update rate, the movement of the sphere is interpolated between motion updates from the phone, in order to obtain a smooth visually pleasing movement. The updates are frequent enough to be able to effectively control the display contents.

This application does not use *point & shoot*. Instead, individual selectable items on the screen are associated with permanent visual codes. Aiming at a visual code shown on the large display brings up an associated menu. Individual menu items can now be selected, whereupon the related content is transferred to the phone and shown on the device screen.

The application was informally tested by a number of subjects and worked well. We provided the test subjects with different tasks, such as rotating a certain menu to the foreground and selecting a specific menu item. After a short period of practice, the subjects quickly became familiar with this type of interaction. This application was shown as a demonstration at [174].

6.3.3 PhotoWall

The "Photo Wall" [20] is a large public display for organizing photos taken with the camera phone (see Figures 6.9 and 6.10). The large-scale display is also used as an access point to photo printing services and one's online photo collection. Because of the small display size, organizing photos is an inconvenient task if done on the mobile phone itself. In addition, the processing power of current camera phones is not sufficient for ambitious image processing algorithms. Uploading them to a PC can also be awkward, because of installation efforts, and the user needs to have a PC in the first place. PhotoWalls could be installed at public places, such as shopping malls or train stations, or at semi-public places, such as office hallways.

A usage scenario of the PhotoWall looks as follows: A user approaches the PhotoWall with the camera phone and connects the phone with the wall via Bluetooth. To do this, the user merely scans a visual code on the PhotoWall, labeled "connect." Upon connecting, the user is identified with a unique login name and password. The PhotoWall stores previously uploaded photos in an online photo album and administers a personal workspace with different folders for photos. Newly taken photos on the phone – or thumbnails of those photos, respectively – are uploaded and shown on the wall. Interaction with the uploaded media elements is done with the keypad of the camera phone, with the *sweep* interaction technique, and with visual code widgets (see Chapter 5) that are displayed on the wall and associated with different application functions. With these mechanisms the user can navigate through the uploaded photos, resize, rotate, sort, and delete them. In addition, users can change brightness and contrast. Selected photos can be sent to a print service or sent to an email address as online picture postcards. Photos can also be marked as "public," which means that they are shown in a slide show on the photo wall, as long as no user is logged in and actively using it. A further option (which has not been implemented) would be to enable the download and exchange of photos between different users and thus turn the PhotoWall into a kind of online photo exchange.

Technically, the PhotoWall is a test application for the use of visual code widgets in conjunction with large-scale displays. As shown in Figure 6.9, individual

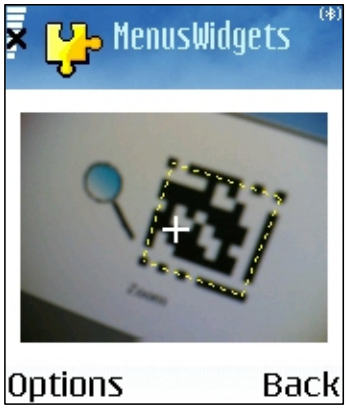
Figure 6.9: Screenshot of the PhotoWall when a user is logged in (from [20]).

interaction elements on the display are associated with visual code widgets. The current photos are shown in the center of the screen. A photo can be selected by using the joystick button on the phone. Figure 6.9a shows visual code check boxes. Focusing them allows the user to make the selected photo public, to print it, and to send it via email (see Figure 6.10a for the corresponding phone screenshot). Public photos are allowed to be shown in the slide show when the user leaves the PhotoWall. Figures 6.9b and 6.10b show a visual code menu for cutting, pasting, and deleting the currently selected photo. Figures 6.9c and 6.10d depict a text entry widget. Selecting it brings up a single-line text input box for editing the name of a photo (see Figure 6.10e). Zoom (see Figure 6.9d), brightness (see Figure 6.9h), and contrast (see Figure 6.9i) are associated with relative movement widgets. Activating these widgets involves focusing them and pressing-and-holding the joystick button. Once active, no visual code needs to be present in the view of the camera and the *sweep* technique is used to change the corresponding parameter. While active, a slider is shown on the PhotoWall next to the relative movement widget (see Figure 6.10f) and is updated in real time. In addition, the photo wall provides for the administration of a user's photos in an online photo album. Photos can be organized into different folders (see Figure 6.9 e and f).

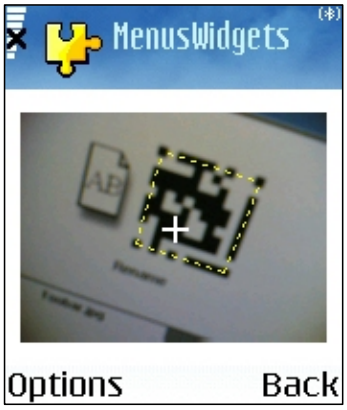
Informal tests have shown that PhotoWall is easily usable. The use of relative movement widgets, such as zoom, was confusing for first time users, because they have to switch attention between their camera phone and the large display during interaction. They fist look at the phone screen to focus the relative movement widget. But then the slider appears on the large display and forces them to switch attention to the large display. At that moment, the slider is obstructed by the

a) visual code checkboxes for making the selected photo public, printing, and sending via email

b) visual code menu for cutting, pasting, and deleting the selected photo

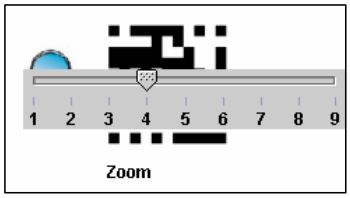


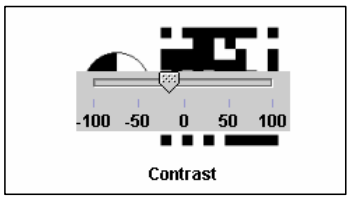
c) a relative movement widget for changing the zoom factor; active while pressing the joystick button (see below)



d) text input widget; if activated, a single-line text input box is shown (see right)

e) single-line text input box for editing the name of a photo





f) sliders are used in conjunction with relative movement widgets; only visible when activated

Figure 6.10: Phone screenshots of visual code widgets for interacting with the PhotoWall.

camera phone. Telling users that they do not keep the on-display slider in the view of the camera resolved this issue.

6.4 Usability Evaluation

In [10] Ballagas et al. have evaluated *sweep* and *point & shoot*. The evaluation of the two interaction techniques shed light on their particular strengths and weaknesses. Users performed a multi-directional tapping test in order to gather quantitative information. They played with the jigsaw puzzle game described in Section 6.3.1 in order to present the techniques in a broader application context and to elicit more subjective user opinions.

6.4.1 Goals and Design

For *sweep*, *point & shoot*, as well as the phone's joystick, users completed a multi-directional tapping test. The test was based on ISO 9241-9 [107]. The interaction

device was a Nokia 6600. As a large display, a Panasonic 60-inch plasma display was used. During interaction, users stood at a distance of one meter from the display. Users had to select highlighted targets as quickly as possible. Tapping targets were located equally spaced at the perimeter of a circle (see Figure 6.11). A single index of difficulty[5] of 3 was used (distance of 612 pixels and width of 87 pixels). A complete Fitts' Law throughput analysis was not done, since the tested interactions are more complex than simple linear movements. In addition, moving the cursor across the screen with the *sweep* technique requires multiple repositionings of the arm. Times and errors were just measured to be able to compare the different techniques. The task completion times were measured as the times between consecutive selections. A selection outside the highlighted target was counted as an error.

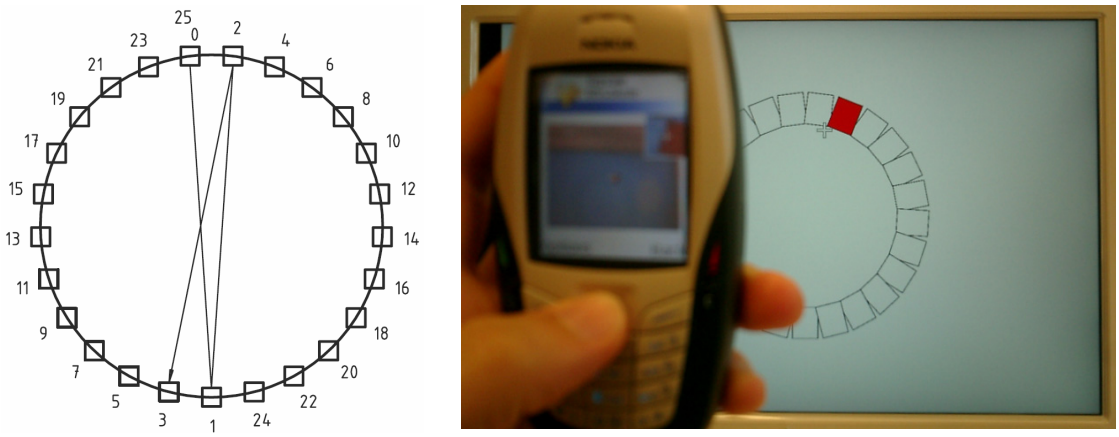


Figure 6.11: Multi-directional tapping test based on ISO 9241-9 [107].

Initially, users filled out a questionnaire that asked for experience with using phones and biographical data. Users were allowed to practice each technique until they felt comfortable with it. After testing each of the three techniques, they completed a questionnaire, which was based upon [107], with subjective rating scales for different parameters, such as comfort and perceived performance. Of the 10 participants (6 men, 4 women), 6 were from the UK, 3 from non-UK Europe, and one from outside Europe. 5 were aged 26-35 years, 4 were aged 17-25, and one was over 45. Most of the participants had no prior experience in using camera phones.

6.4.2 Findings and Discussion

The results and findings are only summarized here. For a more detailed discussion, see [10]. The results for task completion times, grouped by input technique, are shown in Figure 6.12. *Sweep* is significantly slower than both *point & shoot* and joystick input. There is no significant difference between joystick input and *point & shoot*.

Figure 6.13 shows the results for error rates. The error rate of *point & shoot* is significantly different from the other techniques. It has a larger deviation and a

[5]Index of difficulty ID = $log_2(\frac{d+w}{w})$ (see [107], Section B.5.1.2).

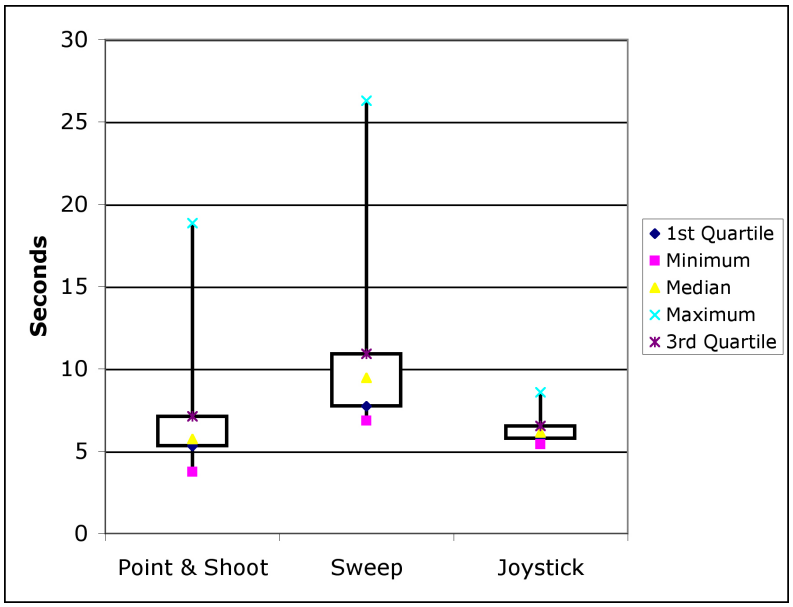


Figure 6.12: Task completion time results for multidirectional tapping test grouped by input technique (from [10]).

larger positive skew. No significant difference was found between *sweep* and joystick input. The subjective ratings with significant differences are shown in Figure 6.14.

For one older user, *point & shoot* was barely usable, since the user could not distinguish the small crosshair on the phone display and trembling hands made it difficult to aim.

Several users adopted a two-handed grasping technique, even though the techniques could be operated with a single hand. The reason seems to be that using both hands made aiming easier. Moreover, the form factor of the used device made one-handed operation akin to "trying to hold onto a bar of soap," as one user remarked. Other form factors would clearly improve the interaction experience.

At the end of the testing sessions, several users reported that their thumbs hurt. Although more experience may lead to a more relaxed interaction style, this might indicate that the interaction techniques are not suitable for prolonged times of use.

Due to the arbitrary mapping of buttons to input modes (see Figure 6.5 and Table 6.1), several users made errors during activation. If multiple input modes are to be combined, clearer affordances are required to reduce these mistakes. The mistakes slightly distorted the error rates for *sweep*.

In summary, the prototype hardware and software implementation did not provide a fluid interaction experience for the *sweep* technique. It is currently not suited for fine-grained cursor control. *Point & shoot* was as good as the joystick for task completion times, but exhibited higher error rates. *Point & shoot* is not just a pointing method, but moreover allows to identify the target object by way of an associated visual code. It is thus more powerful than simple pointing techniques.

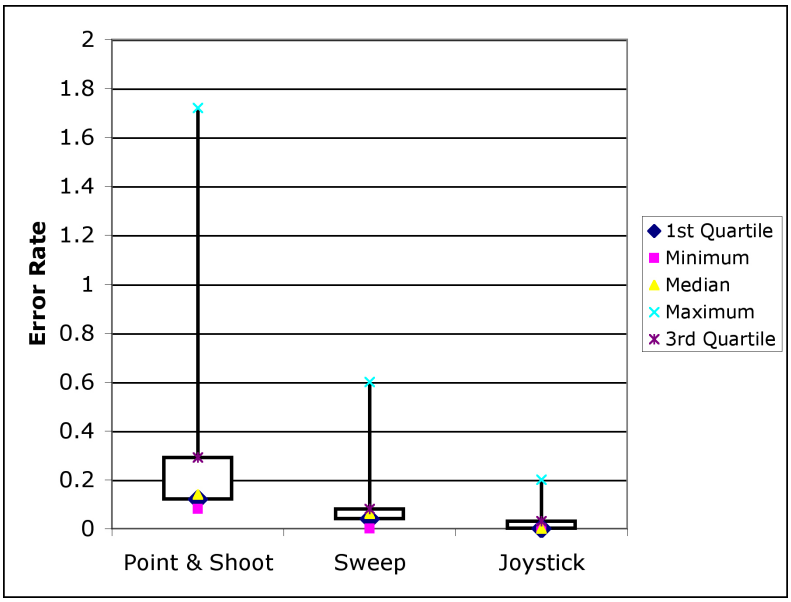


Figure 6.13: Error rates for multidirectional tapping test grouped by input technique (from [10]).

6.5 Related Work

Situated public displays have received increasing attention in recent years [153]. A number of projects have looked at supporting synchronous collaborative work [116, 196]. Other projects investigated informal social interaction in community spaces [34, 87, 106, 114]. Still other projects focused on ambient information displays in public spaces [104, 206]. Whereas most projects investigate the interaction with individual large displays in isolation, a few systems focus on the coordination of multiple displays that are dispersed in physical space (for example throughout a campus) or co-located (for example within a large office building), in order to provide a coherent and seamless mode of interaction for their users [46, 124].

The social setting is a crucial factor for any interaction within public space. Interactions with large public displays are observable by passive bystanders and thus have the potential to lead to feelings of social embarrassment. Thus, people are often reluctant to interact with large displays in public spaces. In [35], Brignull and Rogers investigate how to encourage people to take part in the interaction with large public displays. They examine the "flow" of people around public displays, the different levels of engagement and interaction, factors that cause social embarrassment, and factors that motivate people to join in. In a user study they placed their "Opinionizer" system in party-like gatherings. The system allows users to enter their opinion about a common theme on a projected display by using a stationary laptop computer. Input choices consist of text messages, cartoon avatars and speech bubbles. The avatars represent gender and mood and can be placed on different parts of the projected display to indicate different professional backgrounds. Brignull and Rogers identify three distinct *activity spaces* with increasing intensity of participation: peripheral awareness of activities (people who are aware of the presence of the system, but are engaging in other activities), focal awareness of activities (onlookers who passively observe the display), direct interaction activities

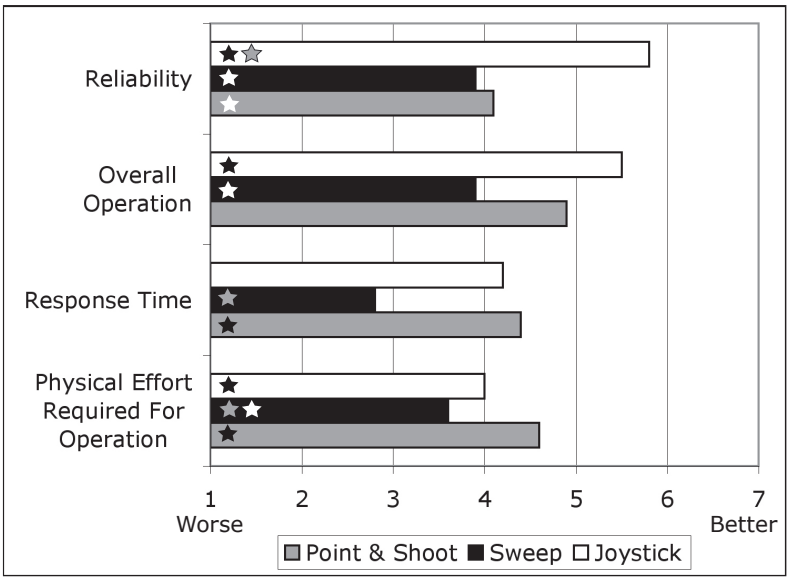


Figure 6.14: Significant results from the subjective ratings of the three input mechanisms. Insignificant results are omitted for clarity. The shading of the stars indicates the technique for which a statistically significant relationship exists (from [10]).

(individuals who enter their opinion). There are *thresholds* in both transitioning from peripheral to focal awareness and from focal awareness to direct interaction. The first transition requires that the display looks attractive when glancing from a distance. This also means that its purpose can be rapidly understood. The second transition to active participation is more difficult to overcome. To lower it, several factors need to be paid attention to. The initiation of the interaction must not take lengthy registration steps, the interaction must be simple and errors must not have potentially embarrassing effects. Before engaging in the interaction, people have to be tacitly reassured that the potential for social embarrassment is low. This in particular means that people have to know what the benefit of participation is, how long the interaction takes, what steps are involved, and if there is a quick and graceful way out.

A number of systems have used personal devices for direct manipulation interactions with large displays. The "Remote Commander" enables individuals to use a PDA to control the mouse and keyboard on a remote display using the PDA's touch sensitive display for mouse input and *Graffiti*[6] for text entry [149]. Similarly, "PebblesDraw" is a single display groupware system that captures input from PDAs and allows multiple users to draw simultaneously [149]. Both systems utilize touch sensitive displays on a PalmPilot and require two hands for operation. Since the systems focus on semi-public display environments, they do not provide any mechanisms for serendipitous interactions.

Madhavapeddy et al. [133] let users manipulate tagged GUI elements (such as dials and sliders) with a camera phone, but do not support cursor manipulation. *Point and shoot* uses visual codes to set up an absolute coordinate system on the display surface instead of tagging individual objects on the screen. The visual codes

[6]Graffiti is the pen-gesture based text input method for Palm devices.

of the grid are normally hidden to support existing applications and GUI toolkits. Our sweep technique can be operated without altering conventional applications and even without pointing the camera at the display.

The C-Blink [144] system uses the phone screen as an input device. The user runs a program on the phone to rapidly change the hue of the phone screen and then waves the phone in front of a camera mounted on the large display. This tracks the position of the phone to control a cursor on the large display.

Many researchers have recognized the opportunity of using personal devices for interactions with public displays. The "SharedNotes" system [86] employs a PDA to create and manipulate personal and public notes on a semi-public display. This solution uses replication of files across devices and synchronization of the modifications. "Web Wall" [68] and "Digital Graffiti" [43] allow users to post comments and to annotate a public digital bulletin board using PDAs. Users can post text messages to the Web Wall in order to create and interact with different media elements, such as polls, auctions, and photos.

Rukzio et al. [181] analyze personalization aspects of large public displays and propose various applicable interaction styles using mobile devices. Kruppa and Krüger [126] compare approaches for the combined use of PDAs and large displays and propose techniques for their simultaneous use. The "PEACH" project [125] employs PDAs to interact with large public displays in a museum environment. This project demonstrates how personal devices can support simultaneous multi-user interaction by providing each user a local user interface on the PDA that corresponds to the nearest exhibit. The exhibit combines personal and public content based on the visitor's interests. Magerkurth and Tandler [134] propose a text entry technique for large displays using PDAs with touch screen that can be operated "blindly," but that uses both hands for interaction. Aizawa et al. [6] propose a ubiquitous display system in which displays are placed in many public locations and operated via the keypad of a mobile phone. All of these systems allow users to issue discrete commands to the large display using the handheld device, but they do not provide support for fluid interactions.

6.6 Summary

Displays that are situated in public places are often inaccessible for direct, touch-based input. Today's large public displays are thus often limited to passive reception of the displayed information. However, if people carry their own interaction device in the form of a camera phone, this situation can be changed. Personal devices in general and camera phones in particular fulfill several of the design requirements for interaction with large-scale public displays, such as high serendipity and high portability. We have presented two interaction techniques for camera phones that rely on visual movement detection and visual code recognition, respectively. The first technique, called *sweep*, uses visual movement detection to enable relative positioning and direct manipulation of objects on the large display. The second technique, called *point & shoot*, realizes absolute positioning by shortly overlaying a visual code grid over the display that is used to establish a shared coordinate system. A usability study has shown that the performance of current prototype implementations of these techniques is not yet sufficient. Higher frame rates and increased processing power on handheld devices are likely to change this

situation. Finally, we have described a few example applications to illustrate the techniques in a broader context.

Chapter 7

Camera Phones with Pen Input as Annotation Devices

This chapter explores the use of camera phones with pen input as a platform for generating digital annotations to real-world objects. First, we analyze the client-side requirements for a general annotation system that is applicable in mobile as well as stationary settings. We outline ways to create and interact with digital annotations using the camera and pen-based input. Two prototypically implemented annotation techniques are presented. The first technique uses visual markers for digital annotations of individual items in printed photos. The second technique addresses the annotation of street signs and indication panels. It is based on image matching that is facilitated by interactively established 4-point correspondences.

7.1 Introduction

Digital annotations link user-generated digital media to physical objects. This allows users to combine the persistency and availability of physical objects with the flexibility and versatility of digital media. Physical media, like printed photographs and street signs, are tangible and permanent, but can typically store only a limited amount and type of information. Digital media, like text, graphics, audio, and video, are immaterial and volatile, but are virtually unlimited in terms of the amount and type of information they can represent. They can be automatically processed and shared across space and time. Using physical media as entry points [173] to digital annotations is a way to structure information and to embed it into the real world. In comparison to other types of augmentation the distinguishing feature of annotations is that users can freely create them and that they are not predefined by the system or any content provider. The actual content of digital annotations strongly depends on the kind of object and the interests of the user, but could answer questions like these: What are similar objects? What are complementary objects? What similar objects are better or worse? Or: Who else likes this object?

Many projects have looked into annotating physical media with online information and services [17, 121, 132, 161, 173, 193, 194, 211]. Our goal in this chapter is to explore the interaction possibilities of camera phones (or camera-equipped PDAs) with pen-based input as a platform for generating digital annotations. We present ideas of how to create and interact with annotations using phonecam-specific fea-

tures. More generally, we are interested in how a mobile user interface for a generic annotation system could be structured that allows for the creation, access, sharing, and organization of digital annotations. This system shall be usable in stationary and mobile settings and exclusively rely on mobile devices as user interfaces.

Camera phones fulfill several essential requirements of a mobile annotation device. First, the camera in combination with image processing algorithms allows for identifying annotatable objects and for determining their orientation. There are multiple options for visually identifying physical objects, including image recognition techniques and visual marker detection. RFID tagging and near-field communication (NFC) for mobile devices [151] offer non-visual alternatives. Determining the orientation of objects in the camera image enables the registration of graphical overlays in the camera image in the sense of augmented reality [7, 66]. Second, wireless connectivity allows for sharing annotations with others, persistently storing and organizing them on a backend server, and getting up-to-date information. Third, camera phones combine the ability to create annotations in multiple media types with the ability to play them back. Fourth, they are ubiquitously available in users' everyday situations.

A distinct feature of pen-based input devices is that they enable users to make more fine-grained annotations of objects captured with the camera. Users can encircle objects and create specific annotations to them, draw arrows to give directions, or put predefined icons onto the captured image. In addition to allowing for more fine-grained annotations, pen-input can also support image processing algorithms by telling the system which objects are relevant and which ones are not. In Section 7.4 we show how this can be used to accurately segment street signs in images from the background.

Digital annotations can take many forms, such as text, graphics, audio, video, hyperlinks, vCard and vCalendar items, drawings and predefined graphical objects on the captured image. All of these media types can be created and presented on camera phones. If the semantics of the annotated object or its classification in a taxonomy or ontology are known to the system, then users might be provided with forms for rating objects or widgets for entering specific parameters. This supports users in minimizing the amount of data they have to enter into their mobile device in order to create an annotation. Annotations can further be supported by context data that is automatically gathered from the mobile phone [50], such as the current location or the time of day.

In Section 7.2 we review related work. In Section 7.3 we discuss the annotation of physical media with visual codes [169] in stationary settings. In Section 7.4 we discuss the annotation of signs – or other areas with four clearly distinguishable corners – in outdoor environments, where attaching visual markers might not always be feasible.

7.2 Related Work

The *Annotated Planet* [193, 194] is a platform for annotating physical objects, such as product packages or artwork in a gallery, with online content. The client device is a PDA with an attached barcode reader. Within an art gallery deployment, users can attach text and voice annotations to artwork and can give numeric ratings with predefined feedback forms.

Decurtins et al. [55] present an information model for both informal and structured annotations of printed documents. In this model, active areas on a printed page are bound to annotation objects (text, images, or videos) or to concept objects (representing a concept of the domain of discourse). As a special annotation device, Decurtins et al. use the Anoto[1] pen and Anoto paper. The tip of the Anoto pen has an integrated CCD camera. Sheets of Anoto paper are covered by a very fine pattern that allows the pen to determine its exact position on the paper. In a prototype application, users annotate mammograms printed on Anoto paper.

The Mobile Media Metadata [49, 50] system allows users to create metadata to photos taken with a camera phone. The system gathers contextual metadata – such as time, location, and user identity – at the moment of image capture. Based on the sensed parameters it guesses the most likely annotation the user would probably make and engages the user in a dialog to interactively specify and refine the metadata. The goal is to integrate metadata capture with image capture in order to simplify later image administration and sharing.

The idea of annotated photographs was presented in the *Active Photos* project [121]. However, the annotation and viewing process is different than with our prototype as presented below: The annotation of an Active Photo is done in a Web browser and relies on the availability of a digital version of the photo. Whereas we use a standard off-the-shelf smartphone, a special lap-mounted appliance is needed to interact with Active Photos. A third difference lies in the way regions with annotations are shown in a picture. While Active Photos are placed in a transparent envelope where marking objects offers additional content, we overlay the image shown on the smartphone's display with polygons.

Yeh et al. [224, 225] have developed a mobile image-based search system. Images taken with a camera phone are used as queries and are compared to images on the Web in order to identify suitable keywords and related information. Yeh et al. also present an interactive approach for obtaining the boundaries of an object, which is called *DoubleShot*: Users take two snapshots, one with the object of interest and one without it. A simple image difference can now be used to compute the segmented image with the object only, without any background.

7.3 Digital Annotations with Visual Codes

Even though digital photography has spread rapidly over the past few years, printed photographs are still omnipresent. To explore novel ways of attaching digital content to these artifacts, we have implemented a prototype application that allows for the annotation of pictures in a physical photo album (see Figure 7.1). We use a Windows Mobile 2003 based smartphone with a touch screen and an integrated camera to enable users to attach text or multimedia content (for example voice notes) to arbitrary parts of album pictures. In our approach, we stick visual codes on every page of a conventional photo album. Each marker encodes an identifier that is unique within the physical album and (ideally) unique across different albums. The markers thus identify the album and the album page within it. The markers can either be pre-printed onto pages or they can be supplied to users as individual stickers that they can put on album pages themselves.

[1] `www.anoto.com`

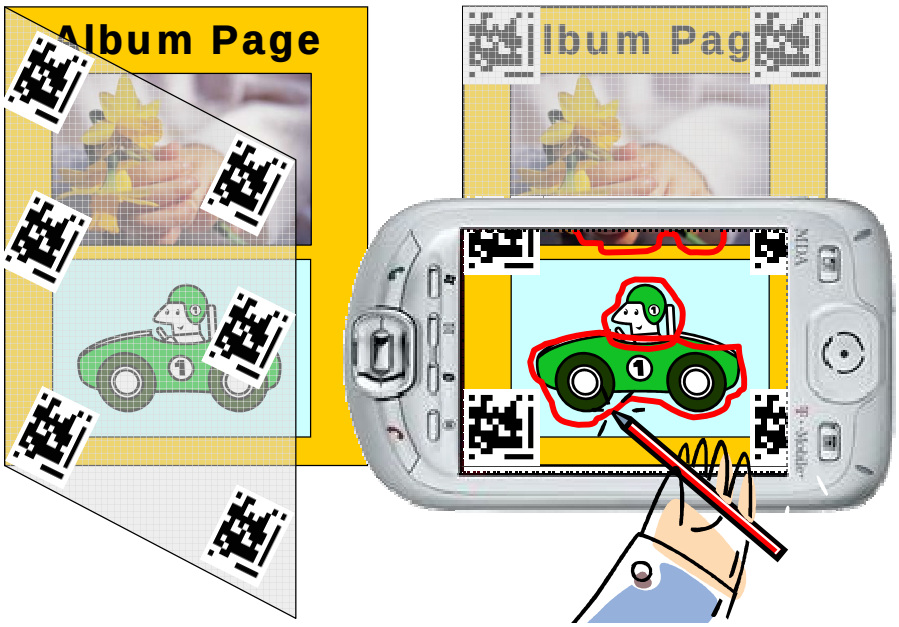


Figure 7.1: Digital annotation of a photo album with visual codes.

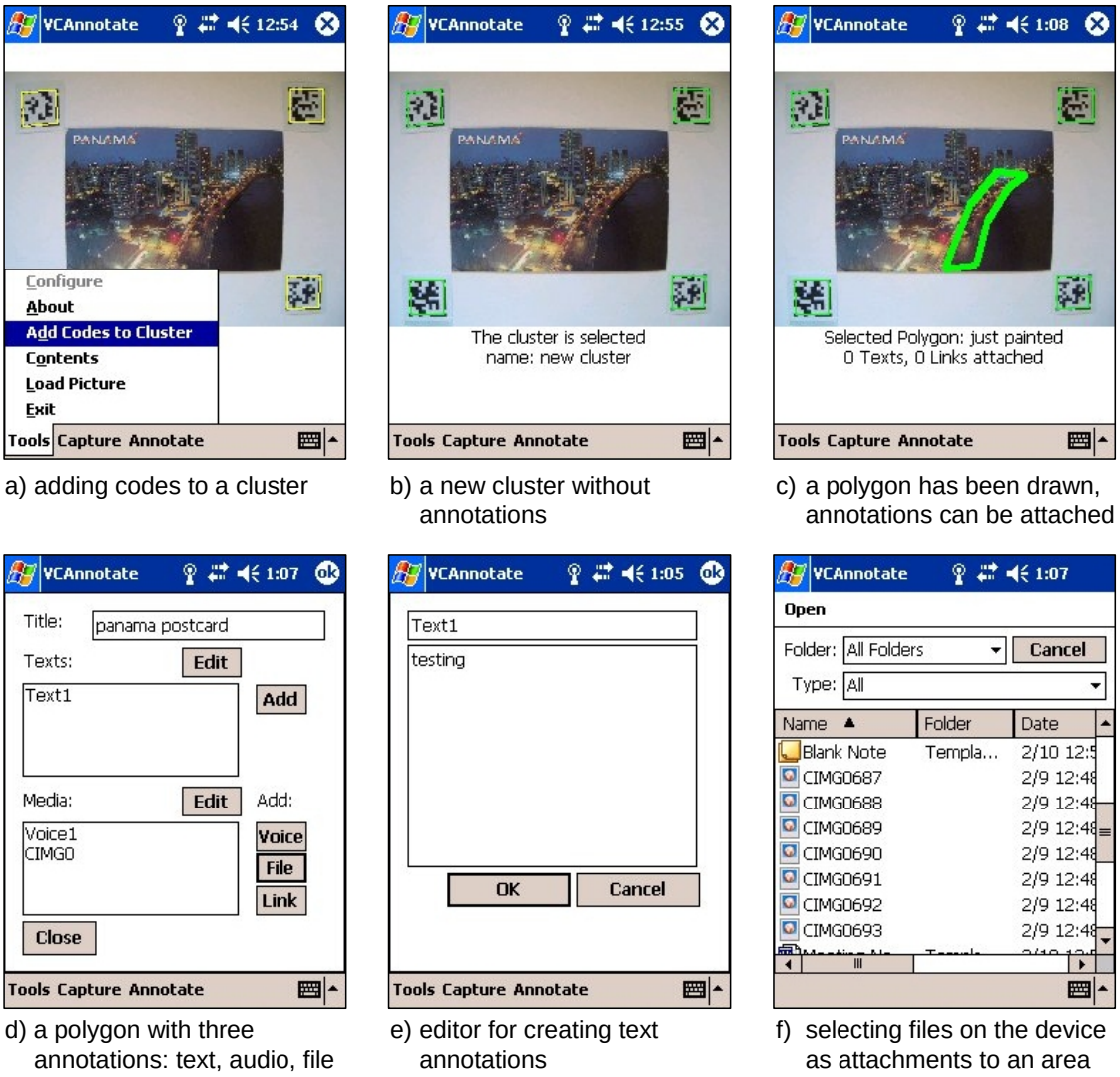
In order to annotate a photo on an album page, users take a picture of that page with their camera-equipped smartphone. Our annotation application, which is running on the phone, then extracts the marker from this snapshot and yields the unique identifier of the album and the page. At the same time, the snapshot is presented to the user on the phone's display. By drawing a polygon with the pen, users can then specify the part of the photo that they would like to annotate.

The phone then maps the polygon drawn on the display to a corresponding polygon in the physical photo's plane. In order to achieve this, we use the features available in the visual code system [169]. In particular, we use the code coordinate systems, which are independent of the camera's orientation. This allows us to transform the display coordinates of the user's drawing on the device's screen into a coordinate system in the physical marker's plane, which gives the position within the page. In this way, the whole album page can be addressed in terms of code coordinates. In principle, each individual point on the album page could carry another annotation.

In our prototype application (see Figure 7.2), users can attach plain text, hyperlinks, voice recordings, and files to a polygon by encircling an object of interest in a photograph. Each photo on an album page can have zero or more polygons. Each polygon can have zero or more annotation objects associated to it. Along with the code's value, which is identifying the album page, and the polygon's coordinates in the code coordinate system, this annotation is sent over GPRS, WLAN, or Bluetooth to a backend server, on which it is stored in a MySQL[2] database.

Obtaining the annotations for a given album page works analogously: When the user takes a snapshot of an album page, the page identifier, which is stored in the visual code, is decoded. The application then fetches the coordinates of all the polygons that are available for the given album page from the backend service. These code coordinates are mapped to the corresponding pixels in the snapshot,

[2] www.mysql.com



a) adding codes to a cluster

b) a new cluster without annotations

c) a polygon has been drawn, annotations can be attached

d) a polygon with three annotations: text, audio, file

e) editor for creating text annotations

f) selecting files on the device as attachments to an area

Figure 7.2: Screenshots of the client application.

which allows the application to highlight the polygons on the phone's display. Users can then read or play back the annotations for an object of their interest by tipping the highlighted polygon that surrounds the object.

The size of a printed code is currently 2×2 cm. The camera provides a resolution of 640×480 pixels. With higher resolution cameras smaller codes (for example, 1×1 cm) can be used, which are less obtrusive. We experienced some difficulties regarding the placing of visual codes on the album pages. Depending on the size of an album page, the smartphone has to be held relatively far away from it in order to take a snapshot of the whole page. As a result of this, the application occasionally could not recognize the visual code any more. We thus attached up to six visual codes to a single album page. This ensures that, when the camera is held closer to the album and covers only a part of it, there is still at least one code located in this part. This, however, incurs the problem of an album part that, depending on how the camera is held, can be seen with a certain code first and another code later. Since each visual code has its own coordinate system, we needed a way to transform the coordinate systems into each other. A conceivable way to achieve this would be to pre-print the visual codes at fixed positions on

album pages. In our prototype, we opted for another approach that still allows users to freely place stickers on a page. However, we introduced an additional step to initialize album pages before annotation. In this step, users have to take a few snapshots of a page that contain several codes at the same time. The application can thus learn about the arrangement of the markers and obtain the data needed to transform the coordinate systems of the different codes into each other. A complete arrangement of visual codes on an album page is called a *cluster* (see Figure 7.2 a and b). To build up a cluster, users take several snapshots with overlapping sets of visual codes.

Since our prototype builds upon the generic platform of smartphones and does not require the annotated object to be available in a digital form, it has a wide array of potential applications. It would be possible to annotate not just photo albums, stamp collections, or elements in a newspaper, but also all kinds of everyday objects ranging from product packages to posters and places in a city. An example for the latter is the *Yellow Arrow*[3] project, which allows people to distribute yellow stickers in urban space and attach text messages to them. Another field are applications where professionals such as the police or insurers need to annotate, for example, crime scenes, accidents or damages. Architects could attach visual code stickers onto construction plans in order to add digital annotations (for example free-form drawings with the stylus) while at the construction site. Yet another application area is medical diagnosis, in particular the annotation of printed X-ray images onto which visual code stickers could be pasted.

7.4 Sign Annotations with Image Matching

Annotating objects by attaching visual markers is sometimes not an option, since the objects may not be under the control of the annotator, physically not reachable, or visual markers might be too obtrusive. This could be the case at public places, for example. Yet many objects, like street signs, shop signs, restaurant signs, indication panels, and even facades of buildings are sufficiently regular and have clear-cut borders to the background in order to be used as annotation anchors. Our idea for using signs as annotation anchors is based on interactive support by users and simple image matching, which makes the approach suited for execution on camera phones with pen-based input. Additionally, context data that is automatically gathered from the mobile phone is taken into account.

In order to attach an annotation to a sign or to retrieve annotations, users take a photo of the sign including any background with their camera phone. The result might be a picture as shown in Figure 7.3a. Users now tap the four corners of the sign on the device screen with their stylus (Figure 7.3b). A frame around the sign appears, whose corners have handles to allow for readjustment. This interactively supported sign selection approach solves two problems. First, if multiple candidate objects are present in the image, the one of interest to the user is selected. Second, the image segmentation process becomes trivial.

To enable simple matching of the framed part of the image (the sign) against a set of templates (Figure 7.3c), the framed part is projected into the unit square (Figure 7.3d). Depending on the orientation of the user towards the sign when

[3] `yellowarrow.net`

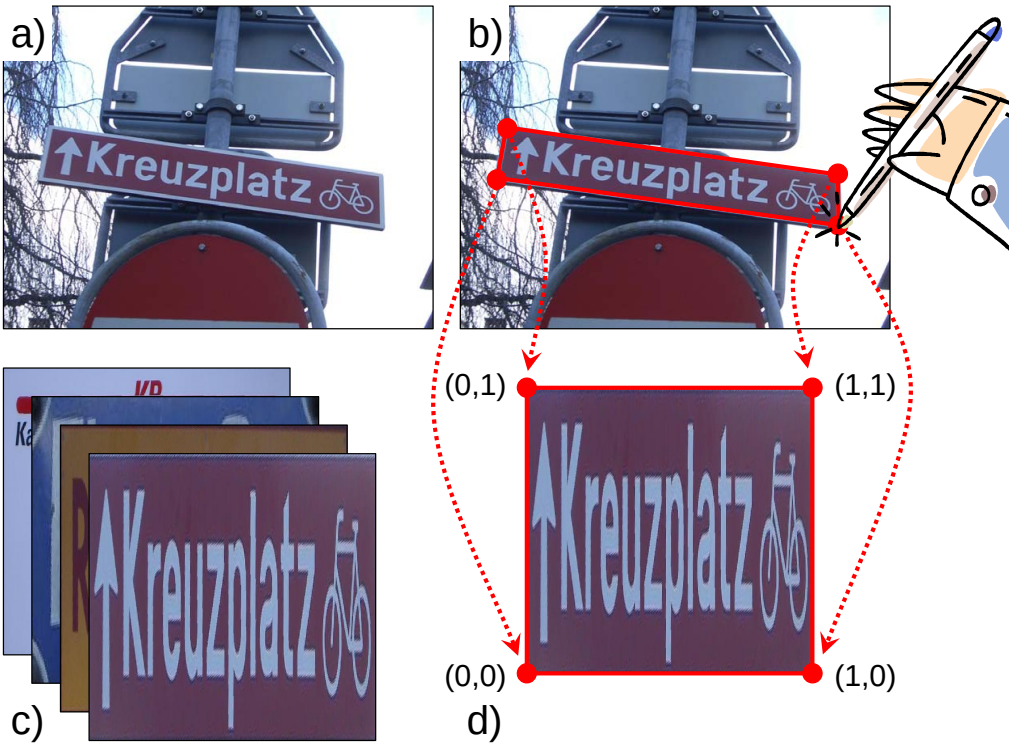


Figure 7.3: Annotating signs using camera phones with pen-based input: (a) captured photo, (b) framing a sign with the pen, (c) set of templates, (d) mapping framed area to unit square.

taking the photo, the sign may appear perspectively distorted in the photo. This perspective distortion can be removed and the framed part projected into the unit square as follows. The four corners of the frame are set as correspondences to the corners of the unit square (Figure 7.3d). Since the frame corners are coplanar, there exists a unique homography (projective transformation matrix) between the pixel coordinates of the framed area and the unit square [95]. By scaling the unit square, we can thus produce a square request image of a predefined size (in our current implementation 480x480 pixels), which is sent to a server for matching against a set of template images. If the mobile device stores the relevant set of templates (Figure 7.3c), then the matching algorithm can also be run on the mobile device.

To further facilitate image matching and to make it more robust, we take a number of context parameters into account that are automatically gathered from the phone at the time of capture and sent to the server together with the request image. The context parameters comprise the current GSM cell id(s) for spatially restricting the search and the time of day (morning, noon, afternoon) to restrict matching to images taken under similar light conditions. The server may optionally add the current weather conditions (sunny, cloudy, rainy) to further restrict the search.

In our current implementation, the actual matching algorithm on the selected subset of templates is executed on a background server, which stores the shared annotations and templates. It computes the sum of differences between the request image and each template image by adding up pixel-by-pixel differences of the hue value. If saturation is below a certain threshold for a pixel, it adds the (gray) value

difference. The server returns a list of matching annotations to the phone, ordered by increasing image difference values.

We expect that if we take contextual parameters, such as the current cell id, the time of day, and the current weather conditions into account, the remaining number of relevant templates will be a few dozen. Preliminary experiments on phonecam-generated images show promise that the matching algorithm can correctly distinguish that number of objects. A problem is of course that images of street signs are very similar – having a common text color and a common background color. Shop signs and restaurant signs typically show more variation in terms of color and visual appearance. Since street names are not unique, different physical signs with the same contents may appear multiple times along a street. In this case, annotations that refer to a particular location are not possible, but only annotations that refer to the street as a whole. This is true for all media that are reproduced multiple times – like flyers, product packages, images in newspapers, or logos.

The approach is beneficial for users, if it requires less effort to take a photo and tap the four corners than to enter some unique textual description of the annotated object (which of course needs to be identical across different persons if annotations are to be shared). Secondly, if the algorithm is not performed on the phone itself, the approach requires the upload of an image part and context data to the server via the phone network, which takes some time. We still need to investigate, how accurately users typically draw frames on a mobile phone with pen-based input and in what way imprecise frames degrade matching performance. In addition to simple pixel-to-pixel color comparisons, better image matching approaches need to be investigated [180]. For signs that have a clear visual border against the background it might suffice if users specify a single point on the sign. Image processing algorithms could then automatically extend the region based on color similarity and find the corners.

The presented approach is applicable if it is not desirable or impossible to attach visual markers to an object. Objects are recognized based on their unmodified visual appearance. Thus the facade of a building can be annotated even from a distance. A disadvantage is that a conscious effort is required for the user to retrieve annotations. Annotations are not discovered automatically, as is the goal in augmented reality systems [7, 66]. Still, there are a number of compelling applications, like pervasive gaming, in which the proposed sign annotation approach can be a component.

7.5 Summary

We have investigated interaction possibilities that camera phones with pen-based input provide for creating digital annotations of physical objects. Camera phones fulfill the technical requirements of object identification and orientation detection, online connectivity for sharing annotations, the ability to create and play back annotations, and ubiquitous availability. Pen-based input allows for more fine-grained annotations. We have presented two digital annotation approaches that are applicable under different circumstances. The first one relies on visual code stickers and enables the annotation of individual items on a printed page. The second approach is based on the interactive establishment of a 4-point correspondence,

which helps separating a selected area from the background and thus simplifies image matching.

The second approach has to be investigated further. The implementation is currently still at an early stage. Topics that need to be explored include user acceptance of tapping the corner points, the typical accuracy of the area framed by the user, better image matching techniques, a larger set of test images taken under different lighting conditions, as well as target applications that can be based on this approach. Applications that we intend to implement are restaurant recommenders as well as pervasive urban games that involve looking for sign annotations within a scavenger hunt. Another aspect is the creation of a coarse taxonomy of annotated objects that would allow for automatic processing of images and annotations. The background system could then automatically provide the user with other relevant shared annotations and related objects.

Chapter 8

Smart Product Packaging

8.1 Introduction

This chapter describes a novel view of product packaging as a medium that provides entry points to Web-based information services.[1] We show interaction concepts for product packages that are equipped with visual codes. These concepts use camera-equipped handheld devices as well as fixed cameras and large-scale displays, which may be placed in a store, for example. In the mobile case, the virtual counterpart of the physical product is selected and displayed with the handheld device. In the stationary setting, product packages themselves are used as interaction instruments, in the sense of tangible user interfaces [103, 113]. In both cases, the orientation parameters of the marker are detected to allow the user to select different information aspects, depending on the current posture – either of the handheld device or the product package.

Product packages are a prominent example of real-world objects for which it is beneficial to attach information technology to. According to [145] 740 billion secondary packages and 1.78 trillion primary packages are produced per year (900 billion board cartons, boxes, sleeves, and trays; 600 billion flexible bags, pouches, and wraps; 130 billion metal containers; 110 billion rigid plastic packages; 30 billion glass bottles and jars; and 10 billion wood boxes). Product packages are thus ubiquitously available in our everyday environment, but related information is only static and essentially limited to the printing area of the package. Online information and services are only very weakly or inconveniently linked, as illustrated in Figure 8.1.

A number of technical challenges have to be overcome in order to achieve a stronger coupling of online services to product packaging. The following components are necessary:

- A physical linking technology as part of the package. Possible options are 1-D and 2-D barcodes and RFID tags.
- A sensor to detect the package.
- An output device to show the associated information and services.

[1] This chapter is based on [142].

Weak entry points to online services on a product package:
"For nutritional information please visit our website: www.lilyobriens.ie sales@lilyobrien.iol.ie"

Figure 8.1: A small product package with inconvenient links to online resources.

- A means of interaction to select information aspects and to manipulate the state of the virtual counterpart. This might be a mobile device that acts as a mediator or the product package itself whose physical manipulation triggers operations.
- Online services and content as well as a supporting background infrastructure.
- A well-defined structure of the virtual counterpart of a product package. The virtual counterpart has to specify the product itself, but also access rights, for example.

Potential information services that may be linked to a physical product comprise information on the product itself, the manufacturer, marketing-related information, as well as anti-counterfaiting measures:

- **Additional information.** Allows the consumer to access additional information not available on the package, for example because of space limitations.
- **Similar or complementary products.** Provides a list of similar or complementary products to the one the user is looking at: "Customers who bought this product, also bought products x, y, and z."
- **Availability check.** Allows the user to immediately check the availability of the product in another size, color, style, etc.
- **Annotation and rating services.** Allows the user to rate a product, possibly after having bought and used it, as well as to look at ratings by other users.
- **Games and quizzes.** Electronic versions if well-known marketing games on packages. Instead of just writing a postcard or sending an SMS, advanced games such as handheld augmented reality games are possible, as described below.

- **Allergy warning services.** The user's mobile device carries a profile that contains information on drugs the user is allergic to. The allergy alert service would intervene in case the contents of product matches an element on the list. This service requires an open data model that allows for automatic matching of items on the allergy list to contents of a product.

The core of the background infrastructure is a model for virtual counterparts of tagged products, for example as a product ontology. This shall provide an open structure for information and services linked to a product that can be used by manufacturers, distributors, and third parties. The structure needs to be flexible enough to unify information by different providers, such as background information by manufacturers as well as reviews by other consumers. For modelling virtual product counterparts, modelling languages such as RDF/S (Resource Description Framework) and OWL (Web Ontology Language) might be used.

Product packaging is an essential component of the retail and consumer industry. A package not only protects valuable goods, but is also a form of communication with potential customers and contributes to forming the "image" of the product it contains. Looking at product packaging from a broader perspective reveals that the area comprises a large number of stakeholders with different – sometimes conflicting – objectives. Product packaging has to serve these different parties in different ways. Additionally, legal requirements make product packaging a highly regulated field.

Manufacturer. The design of a package determines the first impression of the manufacturer's product. It should be appealing to customers in order to positively influence the selling of the product. The manufacturers corporate identity should be clearly conveyed by the package. For certain products, such as chemicals or pharmaceuticals, forgery proofness is an important issue. In addition, a package should be cheap and easy to produce in large quantities.

Distributor. Main requirements are physical robustness, simple handling, good stackability, and low weight to minimize shipping costs.

Consumer. An essential question is what factors influence a consumer's buying decisions during shopping. The price range may be an important issue, but also quality expectations, previous experiences, and advertising. For similar products with no significant price difference, the appearance of the package is the only clue. Often, the look-and-feel of the package might decisive, but also consumers appreciate useful information. According to [62] 81% of surveyed consumers read the labels on food packages to learn about contents of foodstuff. To match consumer expectations, the package should represent the product well. According to the principle "what you see is what you get," it should be sufficient to look at the package instead of looking at its content or the enclosed manual.

Today, consumers are interested not only in the final product, but also in its origin and the production process. According to an Emnid survey [62] in Germany among 1003 men and women, 72% argued for a complete listing of all contents of foodstuff. 64% even want to know about regional provenance of the contents. The contents have to be listed in a way that is understandable and unambiguous. In the survey, only 5% were able to understand details, such as E numbers[2] for food additives, given on the package.

[2]E numbers denote food additives and are typically printed on food labels. The 'E' prefix indicates the additive is approved for use in the European Union.

Legal regulations. What has to be declared on a package differs between different countries and regions. It also depends on the type of product (food, electronic devices, medication, etc.) Within the European Union, a unification of the necessary information has taken place [63, 65].

The supply chain is legally regulated as well. In January 2002, the European Union passed a regulation to improve food safety[3] [64], which became effective in January 2005. It forces all food supply chain members to enable forward and backward tracing of all food-related products, including "food, feed, food-producing animals, and any other substance intended to be, or expected to be, incorporated into a food or feed." The traceability comprises "all stages of production, processing and distribution." The regulation is implemented in the so-called "Rapid Alert System for Food and Feed (RASFF)."[4] It ensures a rapid notification of authorities in case a food risk has been discovered. The alert system does not include the end consumer directly.

Environment. Product packaging have a major impact on the environment, since one-time packaging results in large amounts of waste. Various measures have been taken to enable recycling of materials, to separate waste according to materials (for example "green point" labeling).

Packaging industry. The packaging industry is driven by the requirements of the other entities and can thus be seen as neutral. The packaging industry strives to keep up-to-date with current technology. This not only includes new packaging materials and processes, but also identification technologies, such as RFID tags. Further up the value chain, packaging companies try to get involved in add-on services related to product packages, such as online services for products or marketing campaigns.

In the following, we describe how consumers can get more comprehensive information on the products they buy than from simply reading what is printed on the package. We focus on the interaction of the consumer with the product package, using either mobile devices as mediators between the product and the consumer or physical manipulations with the package itself. Since the kind of interaction only defines the point of contact between the user and online services, we shortly discuss how the virtual counterpart of a smart product package might be structured, in order to satisfy the requirements of the various parties described above. The interaction is targeted at a broad audience, basically any person who is a potential customer for a product. The interaction techniques thus have to be quickly learnable and must allow for serendipitous interaction, i.e. they have to be designed for walk-up-and-use without a significant setup procedure. Product package interactions are often executed in public space, for example in a store, in which the consumer cannot fully concentrate on the interaction, but is distracted by other events in the environment.

The interaction techniques described in this chapter shall enable the provision of information at the point of sale, such as a store. This could make current stores more competitive in comparison to online stores such as Amazon,[5] which routinely provide reviews and product-related information. The techniques described here are targeted to improve the communication channels between the customer and

[3] Web site of the European Food Safety Authority: `efsa.eu.int`

[4] `europa.eu.int/comm/food/food/rapidalert`

[5] `www.amazon.com`

the manufacturer. Customers want to learn ever more about the products they buy, about the origin of the components, and about the production process. For the packaging industry and for the telecommunications industry the approaches described here could highlight new business opportunities. For the packaging industry the virtual enlargement of physical packages by attaching information services to them is attractive, since it overcomes the limitations of the printing space on a package, which is often already fully occupied by legally required product labeling, the manufacturer's logo, and other desirable or stipulated information. Limitations to virtual enlargement are only imposed by the performance of mobile devices and the wireless network, which both increase rapidly. Attaching online services to product packages in the way we describe in this chapter, has a big potential to make marketing campaigns on product packaging more interesting to consumers and improve the response ratio in marketing campaigns.

8.2 Background

8.2.1 Technologies for Identifying Products

Products are typically identified by printing 1-D barcodes or 2-D barcodes on the package. In the future, RFID tags will be attached to packages, which has significant advantages in the supply chain. No line of sight is necessary to perform the identification and multiple items can be identified at once. A detailed comparison of different tagging technologies is given in Chapter 2. Here, we use visual codes not only as an identification technology, but, with its orientation parameters, primarily as an interaction technology. However, visual codes can only be seen as a prototype technology, since many of the problems of optical markers are aggravated on product packages. The need for line of sight and sufficient lighting are minor problems. More severe issues come from the great variety in shapes and surface materials. Many packages, such as bags or bottles, are not planar, which is a problem for the current detection algorithm. Glossy surfaces cause illumination disturbances, like reflections. For the future, optical markers that are detectable on arbitrary surfaces, that operate under a wide range of lighting conditions, and that tolerate partial illumination disturbances, have to be developed. Nonetheless, visual codes work reasonably well on many kinds of product packages and are well suited to demonstrate the proposed interaction techniques. Compared to RFID tags, printed markers have advantages of negligible cost, consumer availability of mobile code detection devices, lesser privacy concerns and absence of environmental concerns. Cost is a crucial issue with product packages, since they are mass products that have to produced as cheaply as possible. Privacy issues have been discussed before in Chapter 2. The most critical aspect is that current RFID tags do not allow for embodied interaction, as visual codes do. In addition, RFID readers are not as wide-spread with consumers as camera phones. This might change in the near future with the incorporation of near-field communication technology (NFC) [151] into mobile phones.

8.2.2 Global Trade Item Number

The Global Trade Item Number (GTIN) as defined within the EAN.UCC System uniquely identifies manufacturers and trade items in the supply chain and in retail establishments. The GTIN is administered by the European Article Numbering (EAN) International[6] through its regional member organizations, such as the Uniform Code Council (UCC)[7] for North America and the Distribution Code Center (DCC) in Japan. The EAN International represents an alliance of numerous manufacturers and distributors world-wide.

The original motivation for establishing UCC and EAN was faster customer checkout, higher data quality (no typing errors), simplification of goods traffic and stock-keeping, and traceability for perishable goods. Historically, the Uniform Product Code (U.P.C.) was introduced in the USA in 1973. In 1977, the EAN was founded to develop a compatible system for the European market. EAN was later renamed to EAN International, since its standards are used world-wide. In an effort to unify both systems, U.P.C. was integrated into EAN. The new system is called EAN.UCC.[8] It includes standards for electronic data interchange across company borders, such as EDI.

The GTIN is a family numbering structures, including UCC-12 (U.P.C.), EAN / UCC-13, EAN / UCC-14, and EAN / UCC-8. These numbering structures are tightly coupled to their representation as 1-D barcodes. This limits their data capacity to the capabilities of 1-D barcodes, which have to be small enough to be printed on product packages. The primary numbering structure that is used in the USA and Canada is U.P.C. It encodes a 12-digit number. Outside North America, EAN numbering structures are used. They encode 8, 13, or 14 digits. GTINs are 14 digit numbers. To generate a valid GTIN, the shorter numbering structures are zero-padded. Figure 8.2 shows the structure of UCC-12 and EAN / UCC-13. The UCC-12 (U.P.C.) numbering structure consists of a 1-digit numbering system character, a 5-digit manufacturer identifier, a 5-digit item number, and a 1-digit modulo check character. The numbering system character allows to differentiate between regular U.P.C. codes, weighted items (such as grocery), and coupons. The manufacturer identifier is assigned by the UCC and has a capacity of 100000 companies. The item number has a capacity of 100000 items per manufacturer and is assigned by the manufacturer. EAN / UCC-13 consists of four variable-length partitions: a 2- or 3-digit country prefix which represents the UCC / EAN Numbering Organization assigning the manufacturer number, a 4- or 5-digit manufacturer number, a 5-digit item number, and a 1-digit check character. The 7-digit country prefix and manufacturer identifier can represent 10 million companies world-wide with 100000 products each.

This shows that the EAN.UCC System is designed to serve trading and is not targeted at the consumer. There are no services attached to GTINs that are accessible to the consumer. The system is also tied to its realization as printed barcodes, which limits automation in the supply chain. To overcome these disadvantages, the Electronic Product Code (EPC), which is based on RFID tagging, has been developed.

[6] www.ean-int.org
[7] www.uc-council.org
[8] www.ean-ucc.org

Figure 8.2: Global trade item numbering structures: (a) UCC-12 (U.P.C.) and (b) EAN/UCC-13.

8.2.3 Electronic Product Code

The Auto-ID Center[9] [78], funded in 1999 and closed down in 2003, was a research consortium of five universities and more than a hundred companies. Its objective was to develop an open standardized architecture for an "Internet of things," based on RFID, to automatically identify consumer products in the supply chain. The research results of the Auto-ID Center are now commercialized by EPCglobal,[10] which is led by global companies, such as Gillette, Procter & Gamble, and Wal-Mart, as well as standardization agencies, such as EAN International and the Uniform Code Council. The university labs of the former Auto-ID Center are now referred to as Auto-ID Labs.[11]

The "EPC Network" is a global infrastructure for the automatic identification across corporate and national boundaries. The goal is to improve business processes in the supply chain, during production, and in stock-keeping. The main elements of the EPC Network infrastructure are

- the specification of cheap RFID tags and readers,
- the Electronic Product Code (EPC),
- the Object Naming Service (ONS), and
- the Product Mark-up Language (PML).

The Electronic Product Code (EPC) [36] is a 96-bit identification number that is stored on RFID tags. A new feature is that the EPC identifies individual physical items using a serial number. The original format, the General-Identifier-Format (GID) divided the 96 bits into an 8-bit header, a 28-bit EPC Manager field, a 24-bit object class, and a 36-bit serial number (see Figure 8.3). To accommodate legacy identifiers, different domain-specific formats, such as SGTIN-96 to integrate GTIN numbers, have been developed.

Since the goal is to equip every trade item with an RFID tag, the cost of a tag is crucial. The hope is that EPC tags will be cheap, since they can be read-only and just need to store the 96-bit EPC. Current research tries to find new methods for

[9] www.autoidcenter.org
[10] www.epcglobalinc.org
[11] www.autoidlabs.org

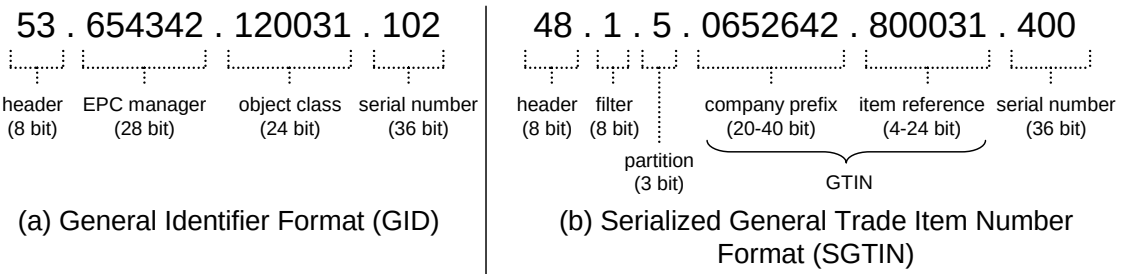


Figure 8.3: EPC numbering structures: (a) General Identifier Format (GID) and (b) Serialized General Trade Item Number Format (SGTIN).

placing the microchip onto the antenna, since this is a major cost factor. The Auto-ID Center has developed a reader reference design, which has since been replaced by a standardized reader-interface-protocol.

In addition to the EPC and the RFID tags, EPCglobal also defines higher level services. A lookup mechanism, called Object Naming Service (ONS), allows to find data related to the EPC number. ONS is based on DNS. It converts EPC numbers (without the serial number) to valid DNS names to get a list of URLs. The URLs refer to services and documents that contain information about the particular product.

The Physical Markup Language (PML) was developed to describe physical objects that are equipped with EPC tags [37]. The original goal was to describe general attributes of objects, processes, and environments in a standardized way, in order to facilitate inventory management, automatic transactions, supply chain tracking, machine control, and inter-object communication. Since this ambitious goal proved difficult, PML core [79] was specified as a first step to allow for a standardized exchange of data generated by RFID readers and sensors.

The EPC Information Service, which is not yet fully specified, is planned to provide the history of tag sightings to enable tracking and tracing. In addition, instance data, such as the production date and the best-before date, shall be provided.

To conclude, the EPC Network focuses on low level technologies and services and is focused on the supply chain rather than the end user. The goal of EPC is to streamline and automate operations in sophisticated supply chains and business-to-business transactions. It currently does not take into account added-value services for the consumer. The latter is the goal of our smart product packaging concept and of the described interactions.

8.2.4 Service-Oriented Infrastructures

Neither EAN.UCC nor EPCglobal are targeted to the end user. User-accessible electronic services attached to product packages are virtually non-existent, apart from a few research prototypes. There are a few online databases to access product information given a U.P.C. or EAN number,[12] but these are mostly accessible from Web pages only and not in mobile settings. URLs begin to appear on product packages, which seldom link to specific services but rather to the manufacturer's homepage. Sometimes marketing campaigns are realized by sending coupons via

[12] `www.upcdatabase.com`

mail or sending text messages with answers to quizzes. There is still room for a lot of improvement as we will describe in the following sections.

8.3 Smart Packaging Infrastructure Concept

The objective of the concept we are introducing here is not to facilitate smooth operation of the supply chain and management of business processes, but rather allowing industries involved in the product supply chain to equip products with services for the consumer. The physical interaction of the user with the product package thus becomes important.

8.3.1 Requirements

Higher-level information services targeted at the consumer need to fulfill several requirements. First, the stakeholders listed in the introduction have to be taken into account. Manufacturers, distributors, retail stores, legal authorities, and third parties each have different interests and a different relation to the consumer. A successful concept needs to take these different interests and relations into account and needs to allow for these different parties to provide their own services to the consumer.

Second, the authority to provide product-related services, as well as access to these services needs to be controlled. Since information services attached to a product virtually extend the product in some sense and have an impact of the consumer's experience of the product as a whole, the manufacturer should have the primary authority to control which services are attached. On the other hand, third party organizations, like consumer-driven rating services surely would want to link their services to a product. For the consumer it should be clear, which services are authorized by whom, in order to decide the degree of trust in a piece of information. This might be achieved by employing trusted signers with in a public key infrastructure. If a machine-detectable marker is attached to a package, it is impossible for the manufacturer to control to what service the identification number is resolved. However, services can be certified by a trusted signer and thus signal the origin of the information to the consumer.

Third, the assignment of services to products should be dynamic, for example depending on the environment (store or home), on current ownership (retailer or consumer), on the date (before or after expiration date), on the time (the service might only be active if you are watching a certain television program and scan the visual code on your coke bottle exactly when one appears on the television screen), or on an external event (virtual soccer-championship marketing campaign is removed once the Swiss team is out of the competition).

Fourth, the content provided might depend on the consumer or the socio-economic group the user belongs to. Content would be presented to a teenager in a different way than to a senior adult. Content might also depend on the history of previously scanned products or even on other products bought before. This requires a user profile and rises privacy concerns. It also requires new skills from advertisers and marketizers. In earlier times they tailored their content to a specific brand or product. Now, they are in a position to know exactly who is interested in a product and can tailor their marketing campaigns more precisely. This shows

that the resolution of the identifier of a product is a complex issue. It not only involves a lookup in a database, but also requires consulting a user profile and taking contextual parameters into account.

Last, but not least, the effort to develop a virtual extension to a product has to pay off in the end. This might be achieved through creating an innovative image of the product, by letting users subscribe to and pay for services, or by using information services as a marketing vehicle to sell a product or related products, product updates, or related services. This can only be achieved by making the services attractive to the consumer. To this end, the appearance of a service to the end user is crucial. Especially the interaction of the user with the virtual counterpart has to be considered in detail. In addition, novel tools for creating and managing the virtual counterparts of products as well as user profiles and scanning histories have to be developed.

8.3.2 Smart Packaging Infrastructure Concept

In this section, our approach for the design of a virtual product infrastructure is described. We concentrate less on the design of a single virtual counterpart of a product, but more on a distributed infrastructure that takes the requirements of the different stakeholders into account. As our main focus is on interaction we only outline the infrastructure instead of completely specifying and implementing it. The latter is out of the scope of this dissertation. The outline describes in what organizational framework the interaction with smart product packages takes place. The main components of the distributed infrastructure are shown in Figure 8.4.

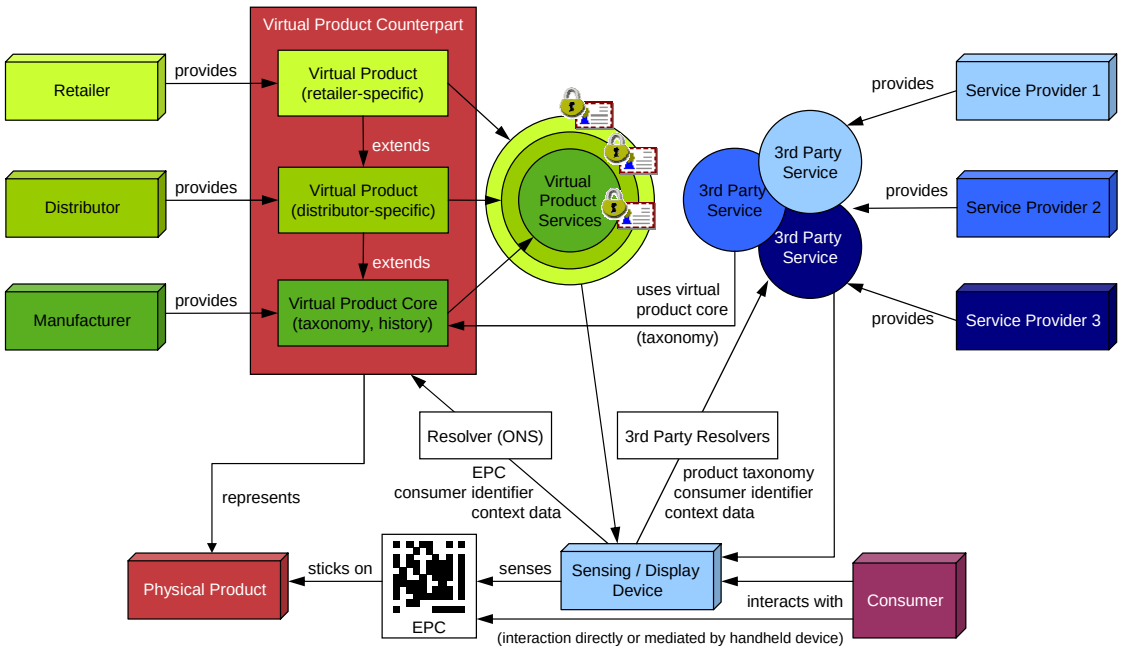


Figure 8.4: Infrastructure for virtual product services.

The starting point in the physical realm is a product package that is equipped with a machine readable identifier that encodes an Electronic Product Code (EPC). In our case this is a visual code, but an RFID tag would also be possible in this infrastructure. The identifier provides just a very limited amount of information.

It thus acts as a key to access the virtual counterpart of the product, i.e. its representative in the virtual world. We propose to use a resolver service like ONS to retrieve the virtual counterpart that is hosted with the manufacturer of the physical product. If the consumer wants to get information intrinsic to the product, the manufacturer is the natural primary point of contact. The manufacturer is responsible for providing information about the origin of the product components, product updates, replacements, spare parts, and the like. The manufacturer's goal should be to improve the communication channels to customers and provide corresponding services. Services that let the product act as the end point of a marketing channel should also be of interest to the manufacturer. Yet the manufacturer is not the only possible service provider. Entities along the supply chain or value chain should be in a position to also provide their specific services. The distributor might, for example, add free warranty returns to a product, locally adapted product descriptions, or local contact information for an imported product. The retailer might add services such as home delivery, discounts, or suggestions for complementary or related products.

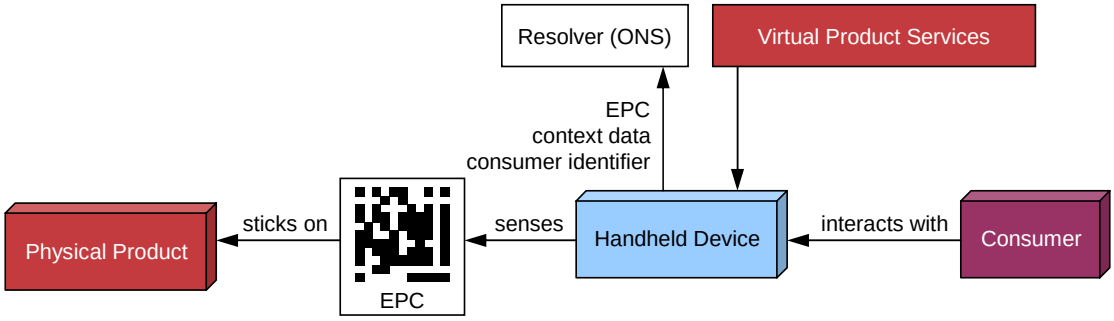
To integrate entities along the supply chain and value chain into service-providing entities for a product, we propose to create a multi-layered virtual counterpart. In this model, the manufacturer would provide a minimal core description of a product that includes its description within an open and standardized product taxonomy. This description would contain intrinsic static attributes of a product class, like its manufacturer and its retail price, as well as attributes of a particular product instance, like the expiration date or inspection date. It would also include the history of events related to the product. This history would record business transactions like transferring the product from the manufacturer's premises to the distributor, transferring ownership to the retailer, etc. The proposed concept relies on the capabilities of the supply chain to generate and record these data automatically, since scanning individual visual codes is not a feasible approach. Furthermore, appropriate access and security mechanisms have to be in place to control entries into the product's history. The virtual product counterpart depends on the particular instance of a physical product. Different instances that are produced by the same manufacturers will potentially have different distributors and different retailers and thus different services associated to them.

Yet, even if multiple suppliers provide product-related information and services, the user should have a homogeneous view of product-related information, at least for the entities along the supply chain. The layered architecture of the virtual counterpart is used to assemble a coherent set of core services for a product. The consumer does not have to be aware that the services are actually provided by different entities. The individual layers of the virtual counterpart have private parts that are only accessible by the respective providers. The services are cryptographically signed in order to convey to the consumer a sense of trust in the provided information and services.

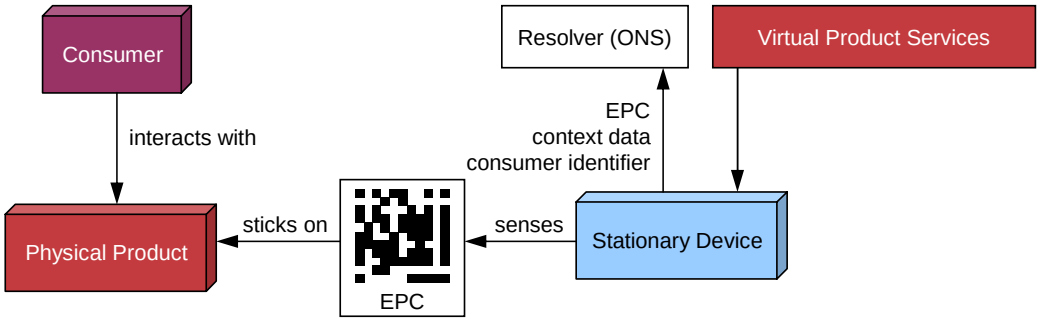
In addition to the primary resolver, which is based on ONS, other resolvers could link to third-party services. These do not necessarily use the raw EPC, but the *virtual product core* that is provided by the manufacturer. This is a publicly accessible piece of information that categorizes a product according to a shared taxonomy. Each category provides a number of predefined attributes that describe the product. This allows third-party service providers to link their services not to

an EPC number, but to a specific category of products. A rating service for digital cameras, for example, would thus register this product category with third-party resolvers. Service provision thus gets simpler for third-party providers. Consumer access to third-party services may be subject to subscriptions or billing services.

Primary virtual product services as well as third-party services may manage consumer profiles that store the users behavior, interest in previous products, type of access device, as well as demographic data about the user. This information can be employed to provide more specific services, but also requires appropriate privacy protection and explicit consumer consent.



(a) Consumer interaction with physical product mediated by handheld device.



(b) Direct consumer interaction with physical product as a tangible user interface.

Figure 8.5: Direct and indirect interaction with a physical product and its virtual counterpart.

In the following, we concentrate on user interaction with product services. We describe two interaction approaches, which are illustrated in Figure 8.5. The first one (see Figure 8.5a) mediates consumer interaction with physical products through handheld devices. It is based on the notion of embodied user interfaces and accesses services by physical manipulation of the handheld device. The second approach (see Figure 8.5b) uses product packages as tangible user interfaces. No mediating devices is necessary, but the consumer interacts with the physical product directly by physically manipulating it.

8.4 Stationary Interaction: Product Packages as Tangible User Interfaces

8.4.1 Motivation

The stationary approach allows for direct interaction with a product package by treating it as a tangible user interface (TUI). To this end, the position and orientation of a package in 3-D space is sensed. The stationary hardware setup consists of a fixed camera that acts as the input device to the system, a large-scale screen to visualize the product counterpart, a computer to which the camera and the screen are attached. The computer processes camera input and generates output on the large screen. It has a network connection to a backend system to access product-related services. The implementation is realized in Java and uses the Java Media Framework (JMF) to access the camera. Such a fixed reader station could be located at different places within a retail store.

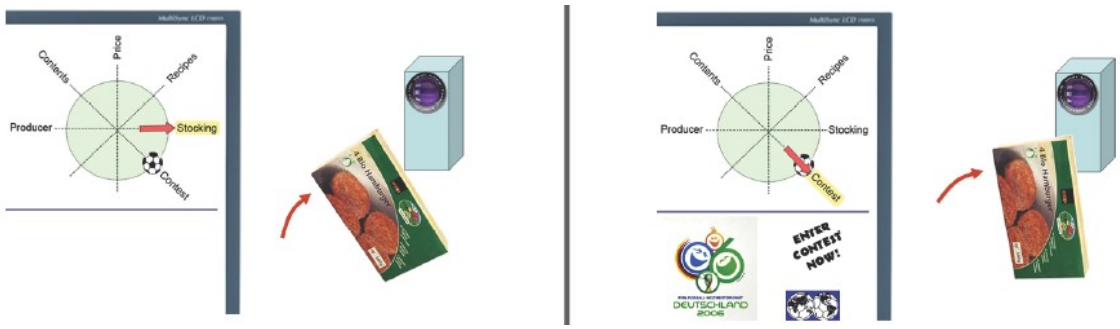


Figure 8.6: A product package as a tangible user interface: the amount of rotation controls which menu item is selected (from [142]).

The advantage of such a setup is that no dedicated input devices, such as a keyboard or a mouse, have to be provided at the station. Such devices could easily be damaged or quickly get worn-out in a retail environment. Moreover, traditional input devices, like mouse or keyboard are cumbersome to use in a retail setting. In our approach, the product package itself is used as the input device, simply by moving it in front of the camera. Initially, operating instructions are displayed on the station's screen. As soon as a package is present in the camera view, the output shows product specific information (see Figure 8.6).

8.4.2 Requirements

The set-up time should be minimal. The stationary approach should be usable without a procedure that requires users to take actions prior to starting the actual interaction with the virtual counterpart of the product, such as logging in. The stationary approach fulfills this requirement since users can just grab a package, walk up to the station and start the interaction.

The interaction techniques should be easy to learn. The retail store is likely to be the only place where users can perform this kind of interaction. This is a huge drawback when compared to the mobile approach, since users are typically well-trained at operating their mobile phones. The interaction techniques also have

to serve a wide range of consumers, potentially anybody who enters a retail store. They thus have to be very simple and be suitable even for first-time users.

The interactions are performed in public space. The station is a shared device. Other users might be waiting in line behind the current user. The screen is usually visible to others. People probably do not like to be observed, especially if they are first-time users who have to learn how to operate the service. People might also feel stressed to quickly finish their turn if other people are waiting. These are all drawbacks when compared to the mobile approach. Users are more likely to take their time to operate their personal device. It is not easily observable by others. It is not a shared resource. In addition, users do not have to walk through the store to find the next station.

Consumers want to save time. Shopping is usually a necessary task, but not high-quality leisure time. Thus, consumers are more likely to use the stationary approach if it gets them required information in a shorter amount of time than, for example, asking a human shop assistant.

8.4.3 TUI Widgets

We have developed a number of interaction components, which are based on the parameters provided by visual codes. Since they are designed for tangible interaction with product packages, we call them *tangible user interface (TUI) widgets*. The widgets described in this chapter have not been thoroughly evaluated in a formal usability study. The discussion of usability is based on personal experience and a few test users. Nonetheless, we gained valuable feedback to help judge the usability of the individual TUI widgets.

An overview of the developed TUI widgets as presented on the stationary display is shown in Figure 8.6. None of these widgets shows the camera image as is done in the mobile approach. The main reason is that in the mobile approach, as the device moves, the camera image remains its orientation relative to the user. It feels as if looking through a magnifying glass that adds virtual overlays. With the stationary approach, the whole background, which is covered by the product package changes when the package is moved. Rotating the package, for example, rotates the camera image as well. This turned out to be very distracting to users. The background image also cluttered the user interface with the camera image. If the camera is facing towards the users, it is very intrusive to them to see themselves and the real-world scene behind their back in the camera image. We therefore decided to provide abstract graphical representations for input and feedback in the stationary case. Moreover, we limited the input task to selection of discrete items. We did not consider the input of precise parameter values, which could be inferred from the object's position. The motivation is that item selection tasks with tangible input based on product packages will provide sufficient input capabilities for interacting with virtual counterparts of product packages.

Position Map

The *position map* (see Figure 8.7a) captures the horizontal and vertical position of the visual code in the camera image. The current position of the code is outlined in the display. The position is horizontally mirrored for intuitive interaction. Without mirroring, the outline of the code would move in the opposite direction of the user's

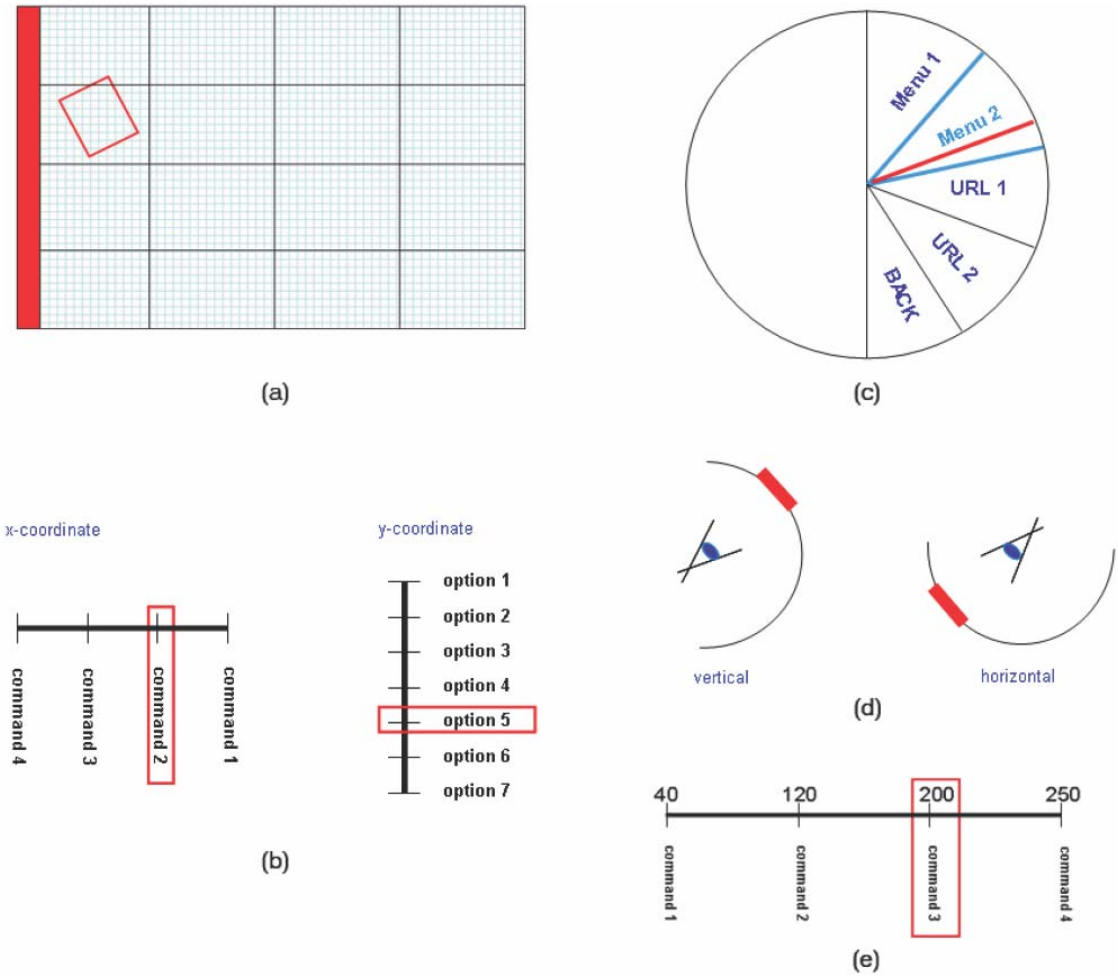


Figure 8.7: TUI widgets: (a) position map, (b) horizontal and vertical sliders, (c) rotation map with menu extension, (d) vertical and horizontal tilting control, and (e) distance map (from [142]).

hand movement, which turned out to be confusing. The original idea was to display a menu item in each cell of the grid and allow the user to select a menu item by moving the outline to the corresponding cell. A red bar (see Figure 8.7a, left edge) indicates to the user that the code is about to leave the camera range.

Usability. Moving a package in 3-D space in front of the camera in order to select a grid cell turned out to be difficult. The unconstrained operation in 3-D space does not appropriately limit the user's movements. The extent of the camera's field of view depends on the distance between the visual code and the camera. Therefore, it was difficult for users to estimate the required position in order to select the desired grid cell. "Blind" operation is not possible, instead the user constantly has to observe the position of the outline on the map and adaptively change the position until the desired menu item is reached. For a natural interaction, visual feedback was not quick enough. There was a perceptible delay between the movement of the physical object and the movement on the screen, which rendered the position map barely unusable as an input widget. Nonetheless, it was well-suited as a feedback provider for the other controls.

Rotation Map

The *rotation map* (see Figure 8.7c) senses the rotation of the visual code relative to the camera. To provide feedback it displays a pointer with the corresponding rotation.

Usability. The *rotation map* is more intuitive and predictable than the position map. Delays and jitters of visual feedback are less of an issue. The main difference compared to the position map is that the target rotation can be precisely estimated by the user without relying on visual feedback. Position and distance relative to the code is not an issue as with the rotation map. The necessary rotary movement is independent of these parameters. Due to physiological limitations of the hand, it is inconvenient to rotate an object by more than $\pm 90°$ from the upright position. The rotation map thus only senses rotation within that range.

Users usually hold a package such that they rotate it around its center of gravity. If the visual code is not placed at the center of gravity, the visual code might inadvertently move out of the cameras area of view. This might require the user to readjust the package in the hand, which is a disruptive task.

Tilting Control

The *tilting control* (see Figure 8.7d) shows the amount of horizontal and vertical tilting of the visual code relative to the camera. In Figure 8.7d, the red bar symbolizes a side-view of the visual code. It is vertically tilted towards the top and horizontally tilted towards the left.

Usability. Tilting was found impractical since its interpretation depends on the rotation of a package. A package that is rotated counterclockwise by 90° and vertically tilted up by the user is interpreted by the system as horizontally tilted to the left, because the left edge of the code is at a closer distance to the camera. Therefore, tilting is not used in the stationary approach. This issue does not arise if the camera is tilted, as in the mobile approach.

Horizontal and Vertical Sliders

A *slider* (see Figure 8.7b) selects one of the tick marks along one dimension. Horizontal sliders (see Figure 8.7b, left) use the mirrored x-coordinate to determine the currently selected item. Vertical sliders (see Figure 8.7b, right) use the y-coordinate to select an item.

Usability. Sliders worked well for most users after a short learning time. Preferences between horizontal and vertical sliders were equally split. The maximum resolution that is conveniently controllable was between 6 to 8 tick marks (choices) per slider. The increased usability compared to the position map is probably related to the reduced number of input degrees. With sliders, users have to control just a single parameter at a time. Position maps control two parameters at once.

Distance Map

A *distance map* (see Figure 8.7e) selects one of a number of items according to the distance between the code and the camera.

Usability. Distance maps are intuitive to use. A major problem is related to the pyramid shape of the camera's field of view. The closer the package is to the

camera, the smaller the field of view. If the user starts at a far-distance position off the camera's optical axis, while moving closer, the camera's field of view will be left at some point. This is counter-intuitive to users.

Discussion of the TUI Widgets

The issues we encountered during development and testing of the TUI widgets can be summarized as follows:

- Unconstrained movement in 3-D space is difficult.

  Moving a package in 3-D space in an unconstrained manner provides many degrees of freedom. This should theoretically be translatable into highly expressive tangible interactions with a large input capacity. Unfortunately, this is not true. A large number of parameters, which are controlled at once, is asking too much from users. This is especially relevant for first-time users and users who do not want to take a long time to learn new interaction techniques.

- TUI widgets with limited degrees of freedom provide better usability results.

  Intuitive usability and expressiveness is a tradeoff. TUI widgets with just on degree of freedom, such as one-dimensional sliders, are simpler to use than TUI widgets with two degrees of freedom, such as position maps.

- The constrained setup has a persistent physical state.

  With the constrained setup, the package remains at the position where the user has put it. If the user puts down the package, the physical state of rotation reflects the selection state of the virtual counterpart. Grasping the package again, the user can continue interaction at the current point. This is obviously not true for the unconstrained interaction in 3-D space.

- Avoid the need for visual feedback.

  It is advantageous to use movements that the user can estimate without visual feedback, such as the target rotation in the rotation map. In this case visual feedback, and the delay and jitter with which it occurs, matters less. Before starting the movement, the user's sensory-motor system can estimate the amplitude of the required rotary movement. For the position map this is impossible, since the interpretation of the position depends on the distance to the camera.

- The view of the camera is limited and its bounds are not obvious to the user.

  When using a single camera, the view of the camera is shaped like an inverse pyramid that is small close to the camera and extends with growing distance. The boundaries of the camera's field of view are not obvious to the user. It thus happens that the user inadvertently moves the object out of the camera view, which interrupts the interaction. The solution to this problem might be to use a wide-angle lens of even an array of cameras that provide a larger area with uniform parameter characteristics. In such an array of cameras, the distance would not have an effect on the position map.

- Heavy products are not suited for lifting.

  Operating in 3-D spaces requires the user to lift the product package in the air. This is obviously not suitable for heavy-weight packages, but only for small retail items.

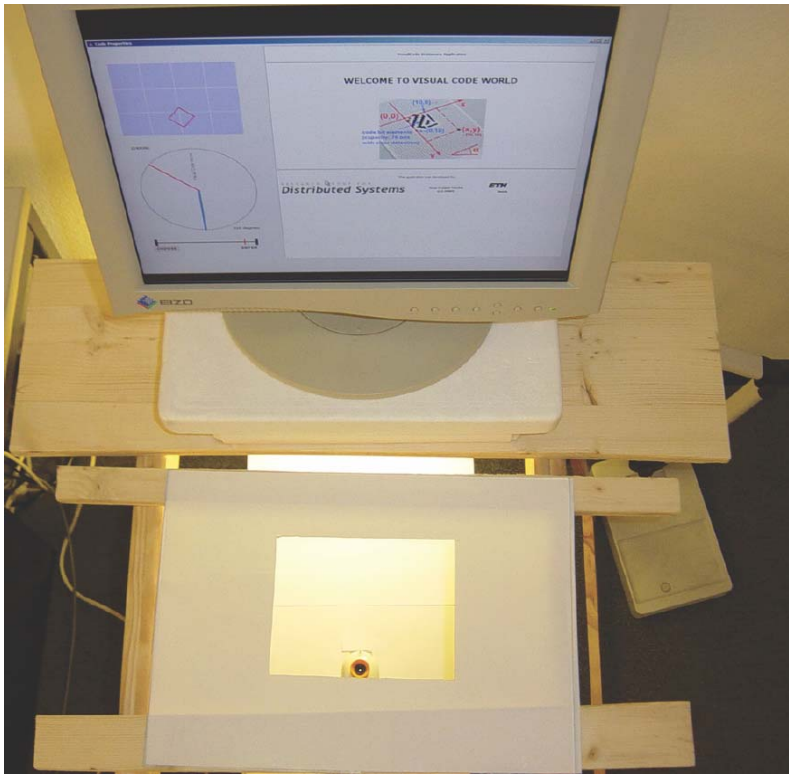


Figure 8.8: Stationary device prototype with coated glass panel (from [142]).

To overcome these issues we decided to restrict the number of degrees of freedom that need to be controlled by the user. This required a modification of the hardware setup (see Figures 8.8 and 8.9). Instead of facing the user, the camera is facing upwards. A coated glass panel is placed in horizontal orientation above the camera. Coated glass was necessary in order to avoid reflections. Indirect light from below and white paper around the camera ensured uniform and constant lighting conditions. The underside of the glass panel was masked by paper in order to indicate the camera's area of view to the user. This setup reduced the degrees of freedom and the necessity to hold the product package in the air. The non-masked area of the glass panel clearly indicates the camera's view area to the user. Short-term lifting of the package is still possible for activating the distance map.

8.4.4 Menu Selection with TUI Widgets

Menu-based user interfaces provide sufficient input capabilities for interaction with virtual product counterparts. Most graphical user interfaces are based on two independent interaction steps: *selection* and *execution*. Selection can be implemented with a single TUI widget as described above. In this case, selecting a menu item immediately shows the associated content on the screen. Interfaces that provide

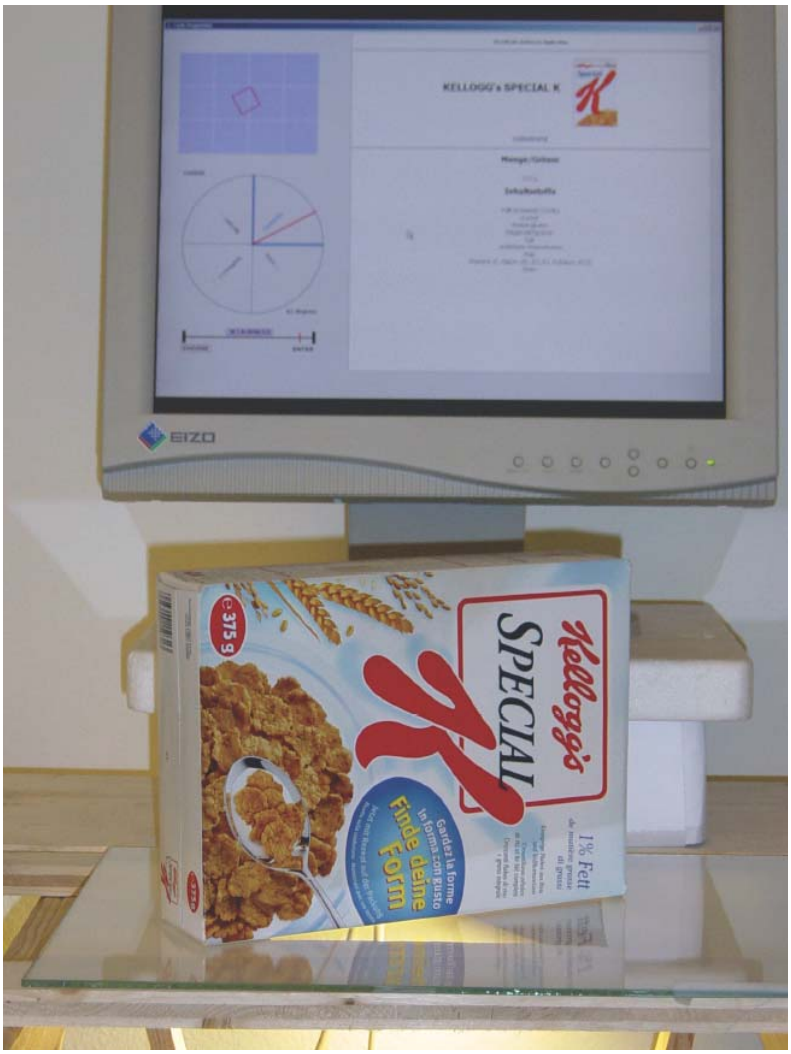


Figure 8.9: Front view of the stationary device prototype. The screen displays the currently selected aspect of the virtual product (from [142]). For a detail view of the screen contents, see Figure 8.10.

only selection are limited. To allow for hierarchical menus and the explicit activation of applications, a second execution gesture in addition to menu item selection is required. This means that two of the basic TUI widgets have to be combined – the first one would select a menu item and the second one would explicitly activate a command or descend into the menu hierarchy. The following discussion is refers to the reduced degree of input version that operates on the glass panel.

Rotation Map and Distance Map

This variant combines the rotation map for menu item selection and the distance map for command execution. The user first rotates package to position the pointer in the middle of the pie menu wedge (see Figure 8.7c) and thereafter slightly lifts the package off the glass panel to execute the command. The distance map is limited to just two states: near for choosing and far for executing.

Usability. The rotation map on its own was intuitively usable. Combining it with the distance map works well, since the rotation state can be easily maintained

while lifting the package. The "lifting up" gesture should be used to trigger the execution, since in that direction unintentionally leaving the camera's field of view is less of an issue. In the distance map, more than two states could easily be encoded.

Position for Selection and Execution

In this variant, translation in x- or y-direction are used for menu item selection as well as execution. A position map or a horizontal and a vertical slider may be used. One dimension would be mapped to menu selection, the other dimension to command execution.

Usability. Even if translation-based widgets are easier to use in the constrained case, they are more difficult to operate than the rotation map. Moreover, the dimensions are not sufficiently independent. It is difficult to maintain a selection, made, for example, based on the x-coordinate, while traversing from choosing to execution by moving in y-direction.

Position and Distance Map

The products position is used for menu item selection. Execution is triggered with a distance map.

Usability. Again, it is difficult to maintain the selection based on the object's position while changing the distance. This variant is thus not a practical approach.

8.4.5 Prototype Application

The prototype application is based on the glass panel hardware setup. It realizes menu selection by TUI widgets and shows content related to the current selection. For its intuitive usability, we chose to use the rotation map as the primary means of selection within an associated pie menu. The distance map was chosen as the execution gesture. The user interface of the prototype application (see Figure 8.10) contains a rotation map for selection, a distance map for execution, a position map to provide additional visual feedback about the position of the code, and the browser panel to display the content requested by the user. The rotation map is associated with a pie menu. Menu items can refer to submenus, content, or the "back" button to go back to the parent menu.

8.5 Mobile Interaction: Augmented Reality on Product Packages

8.5.1 Motivation

This section discusses the application of visual code image maps as described in Chapter 4 to product packages. The limited printing space on product packages is typically completely covered with different kinds of information related to the product or marketing. The example in Figure 8.11 shows a food package that contains preparation instructions, nutrition information, producer contact information, and

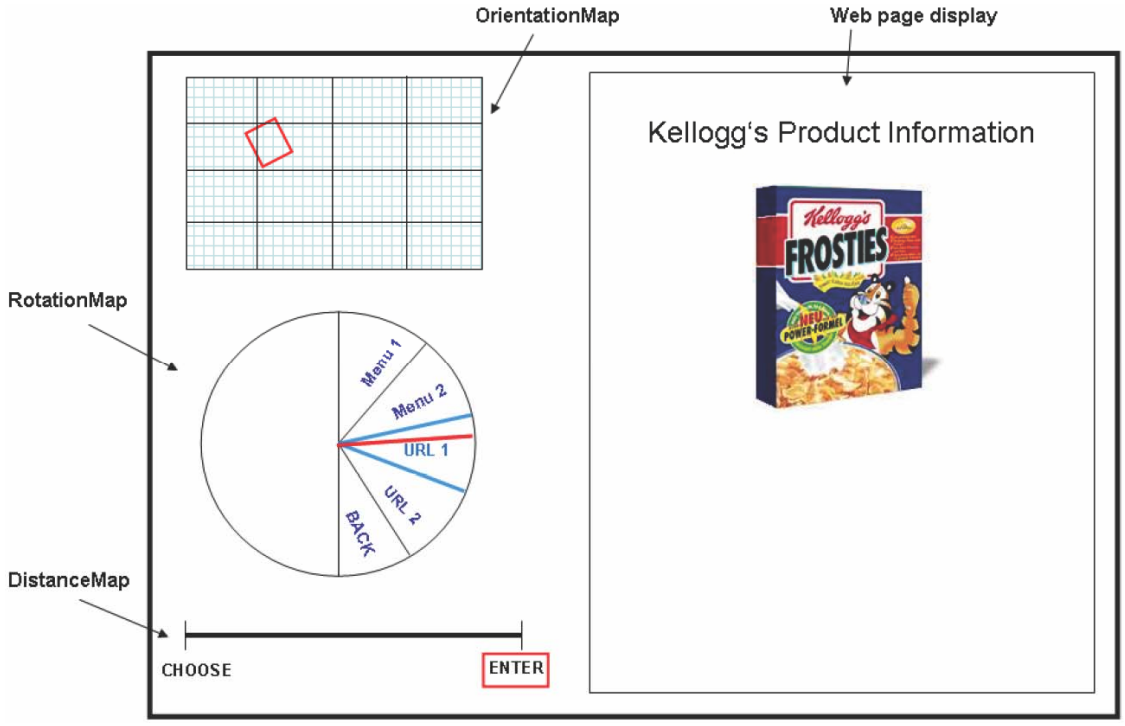


Figure 8.10: User interface of the stationary prototype application (from [142]).

a coupon. In the mobile interaction approach, these already available areas are virtually extended using visual code image maps. Visual code image maps also allow for a second kind of virtual extension that is not reflected in the printed areas, but relies on other visual code parameters, such as distance from the visual code, for example.

The mobile interaction approach provides shoppers with an enhanced shopping experience. They can use their own personal devices as a symbolic magnifying glass that allows them to access product details, reviews, forums, test results, and price comparisons. Through their personal device that has unconstrained online connectivity, users are in a much stronger position to access personalized and unbiased information than in the stationary interaction approach. Users may subscribe to third-party service providers, such as reputed product test organizations or trusted online communities. Unrestricted instant information access might sound like a threat to physical stores, but the effects of immediate availability and possibility to buy without the delay incurred by using an online store as well as the touch-and-feel experience of products in a store, should not be underestimated. Combining the touch-and-feel experience of the real products with the wealth of information available at online stores might actually be a competitive advantage of physical stores when compared to online stores.

8.5.2 Visual Code Image Maps on Product Packages

Visual code image maps solve the linking problem of physical products to their virtual counterparts and also provide a conceptual framework for embodied interaction with these counterparts. The input postures distribute the output across multiple screens and are thus suited to adapt the output to the capabilities of the limited

Figure 8.11: A visual code image map on a product package. The dotted lines show the individual areas of the image map.

screen space. This of course means that adapted content has to be developed for this new medium. Simply copying long pages of text from Web pages designed for static use will not work. It might even be worthwhile to develop standards that are valid across multiple products to achieve some degree of consistency in a user interface based on visual code image maps. A certain posture could, for example, be the standardized posture for price comparisons, another one for product tests, etc.

8.5.3 Communication between Consumer and Manufacturer

The mobile device approach is particularly suited to improve communication channels between consumer and manufacturer. The mobile device can provide an identification of the user that can help to maintain a user model or profile on the side of the manufacturer. The set of products the user accesses can be used to categorize the user into a socio-economic group. Even if a user model is not maintained, for example for privacy reasons, the feedback gained about a product is very valuable for manufacturers.

An interesting example for a realization of the communication channel is sending predefined text messages to the manufacturer based on interaction postures. Each interaction posture can invoke a different text message. The text message could, for example, represent answers to a survey question. Users do not actually need to write the text message, but just focus the visual code that is associated with the survey and choose their answer with the indicated interaction primitives. An example survey question (from [142]) is: “When do you eat Kellogg’s Corn Flakes?” Focusing the code displays rotation interaction primitives. The user can now rotate the phone to select “for breakfast,” “for lunch,” or “for dinner.” Pressing the joystick

button automatically sends the text message. This simple interaction scheme is likely to increase participation rates in such surveys – especially if they are coupled with online lotteries.

The phone action is stored in an image map that is part of the virtual product counterpart. It uses the `SendStaticSMS` action described in Chapter 4 to send a predefined text message to a predefined number. The action of sending the text message is executed upon pressing the joystick button. The text lines are displayed as soon as the rule is activated. An example rule is shown below (from [142]).

```
<Rule name="SendStaticSMS">
  <Rotation category="absolute" start="10" end="180"/>
  <Action functionName="sendStaticSMS" phoneNumber="0781234567"
          body="product=cornflakes survey=1234 answer=breakfast">
    <IconicCue name="Keystroke"/>
    <Line value="I usually eat Kellogg's Corn"/>
    <Line value="Flakes for breakfast."/>
    <Line value="Yes, I would like to win a"/>
    <Line value="free skipping rope."/>
  </Action>
</Rule>
```

Supporting word of mouth between consumers is also possible using visual code image maps. The `SendStaticTextSMS` action can be used to send a predefined text to a phone number entered (or selected from the phone book) by the user. Upon triggering such a rule, the text message editor is automatically invoked, populated with the predefined text, and the cursor is positioned in the receiver phone number text field. Again, the text lines are shown as soon as the rule's constraints are satisfied. See the following example rule (from [142]).

```
<Rule name="SendStaticTextSMS">
  <Distance start="0" end="60"/>
  <Action functionName="sendStaticTextSMS"
          body="Have you heard of the new Chevy Corvette?
                It's a fantastic car! Check it out
                as soon as you get a chance!">
    <IconicCue name="Keystroke"/>
    <Line value="Tell your friend"/>
    <Line value="about the new Corvette!"/>
  </Action>
</Rule>
```

8.5.4 Augmented Reality Games and Animations

We not only see potential in providing background information or third-party information about a product itself, but also in making marketing campaigns on product packages – that are often realized as games or quizzes – more attractive. Marketing campaigns on product packages are not new. Cereal packages have included game boards on the back side for years. Even though these games have been very simple and they appeal only to a very limited proportion of the population, such as infants and children, such games have remained and did not disappear. The idea presented in this section is to link handheld augmented reality games and animations to packages. In this approach, packages provide the visual background for the augmented reality games in the form of a playing field and a visual code.

The playing field containing the visual code is captured by the camera. The games are controlled by applying embodied interaction gestures to the handheld device. Such games would be interesting to a larger proportion of consumers and increase participation rates in marketing campaigns they are part of. The approach opens up new design possibilities for marketing campaigns on product packages.

Game Components

Figure 8.12 shows the hardware setup of the envisioned games. They contain the elements listed below. In the following, we generalize the discussion to include not only product packages, but also other kinds of background media, such as magazines, flyers, tickets, situated displays, etc. This highlights the general applicability of the proposed concept to a large range of media.

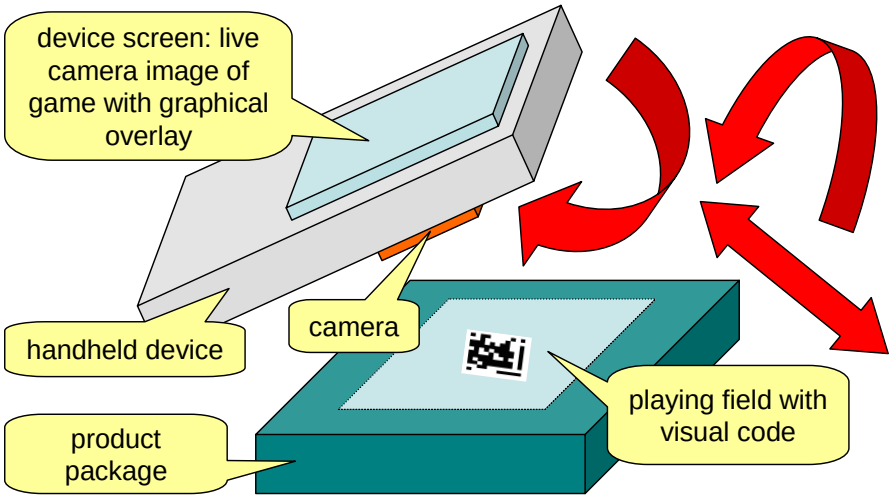


Figure 8.12: Hardware components of the handheld augmented reality game. The game is controlled by applying embodied interaction gestures to the handheld device.

The proposed mobile approach consists of the following elements:

- a visual background medium showing an image or drawing that serves as the playing or application surface and provides visual context to the user;
- a machine-readable marker or other machine-detectable features on the background medium that can be used to automatically identify the game and to compute the spatial orientation of the camera relative to the background medium;
- a camera-equipped mobile device with an electronic display for showing dynamically changing game-related visible (graphical or textual) overlays over the live camera image, the device being also capable of generating other kinds of game-related output, such as audio and tactile output;
- a sensor operable to detect the spatial orientation of the mobile device relative to the background medium (the sensor may be the integrated camera using a machine-readable marker or other machine-detectable features on the background medium);

- a software component (a self-contained application or a generic execution engine that interprets a description language) on the mobile device capable of interpreting user input – primarily based on position and orientation changes or static postures of the device (but also on keyboard input, voice input, touch-screen input, etc.) – in order to directly manipulate the state of the game or animation, and capable of rendering visual output on the screen as well as generating other kinds of output, such as audio output, tactile output, etc.; and

- optionally, a description language for interactive augmented reality games and animations (the game or animation may be completely implemented in the software component, in which case no external description is necessary).

Detailed Description

The visual background medium provides the visual context for the graphical overlays over the live camera image. The background medium and the graphical overlays are together presented on the device screen. A game-enabled product package, such as a box of cereals, has an area on its back side containing a drawing or photography and one or more visual markers. It may also contain other machine-detectable visual features. The area serves as the playing surface and background for the augmented reality game or application. If we look beyond product packages as background media, additional implementation possibilities come to mind. The background can be printed, projected, or electronically displayed. It may be static or dynamically changing, depending on time or on the current state of the game. Examples of printed media are product packages, newspapers, magazines, tickets, posters, flyers, comic books, cartoons, CD covers, and lottery coupons. Printed media comprise paper, plastic, or any other substrate on which can be painted on. Examples of projected background media are beamer-generated images on large screens or floor projections in public space. Electronic displays can also be used as background media. An example of a static background medium is a comic strip printed in a newspaper. An example of a dynamically changing background medium is an electronic large-scale public display, a part of which serves as a game board.

The game or animation is activated by pointing with the camera-equipped mobile device to the playing surface. The visual marker appears in the camera image, it is detected and its value is decoded. The game or animation is loaded (for example by loading an associated description), and presented by the software component. The game is then controlled by changing the orientation of the mobile device relative to the playing surface, by pressing keys on the device, by using the stylus, by issuing voice commands, or by some other means. The live camera image of the focused part of the background medium and its graphical overlays are updated in real time as the device moves.

The game may either be single-user (human player against computer) or multi-user, where multiple users play against each other, possibly remotely on different background media. In this case the mobile devices would communicate over the network to exchange the game state. The devices may also communicate in ad-hoc mode in a peer-to-peer fashion. An example is a game in which two concurrent players each play on their own product package, but take different roles in the game

– like the goal keeper and the shooter in a penalty shoot-out game. Even more than two players may participate, possibly playing asynchronously separated over time.

The orientation sensing mechanism may be separated from the device camera and implemented without a visual marker and without other visual machine-detectable features in a separate hardware unit, such as a 6-D tracker, a distance sensor, an accelerometer, a gyroscope, a compass, a separate camera, ball switches, or with other technologies. In that case, other techniques need to be found to achieve precise alignment ("registration") of the camera image with items in the background medium.

The spatial orientation of the mobile device relative to the background medium may be given either as the full 6-D orientation in space (3-D position plus pitch, roll, and yaw) or as a number of individual parameters such as target point, distance, rotation, and tilting of the camera relative to the marker. The latter approach is realized in the visual code system.

The dynamically changing game-related graphical overlays over the live camera image can be images, icons, text, 2-D drawings, virtual 3-D objects, and GUI widgets. Output can also comprise audio via loudspeakers, headphones, or earpieces as well as tactile output via vibration controls.

The execution engine may be pre installed on the device, stored on a separate storage medium, or dynamically downloaded via the network. It may be a self-contained application or a generic interpreter of a description language for interactive augmented reality games and animations. The descriptions may be stored on the device, on a separate storage medium, or dynamically downloaded via the network.

The game may be implemented as a self-contained software component. This is the approach we took in the prototype game described below. It might also be realized as a generic game execution engine that reads game descriptions, similar to visual code image maps, upon encountering a visual code on a product package. This supports game and content creation by providing a few high level abstractions. The game description, if used, is based on a formal machine-readable language that allows for defining certain aspects of the device's behavior, such as graphical output, audio output, tactile output, controlling applications on the device, and sending data over the network. The formal language allows for describing the device's reaction to user input, particularly based on the device's orientation relative to the background medium, but also on keyboard, touch-screen, and voice input. Moreover, the description language allows for specifying the timed execution of operations, such as animations on the screen. In the variant that uses the description language, the mobile device contains a generic interpreter application that interprets the description and executes operations on the device accordingly. The interpreter holds the current state of execution, including the state of the game. The description can be retrieved via the network, previously stored on the device, or loaded from a separate data storage medium.

The description language is designed to easily describe interactive augmented reality games and animations on a high level and in a compact way. It has elements to specify

- graphical overlays, such as text, images, 2-D, or 3-D objects;

- audio output;
- tactile output;
- the flow of animations in time;
- the use of device capabilities, such as sending data over the network or starting applications;
- rules for reacting to user input, such as orientation changes of the device;
- rules for reacting to sensor input, such as the camera image;
- rules for encoding the game and animation logic.

The device's display shows the camera image of the focused part of the background medium. It also contains virtual graphical or textual overlays over the camera image. The overlays are registered with visual elements of the background medium. Registration is achieved using visual code parameters. The virtual graphical output is not only static, but can also be dynamically animated. The animations can be controlled by events, such as user input.

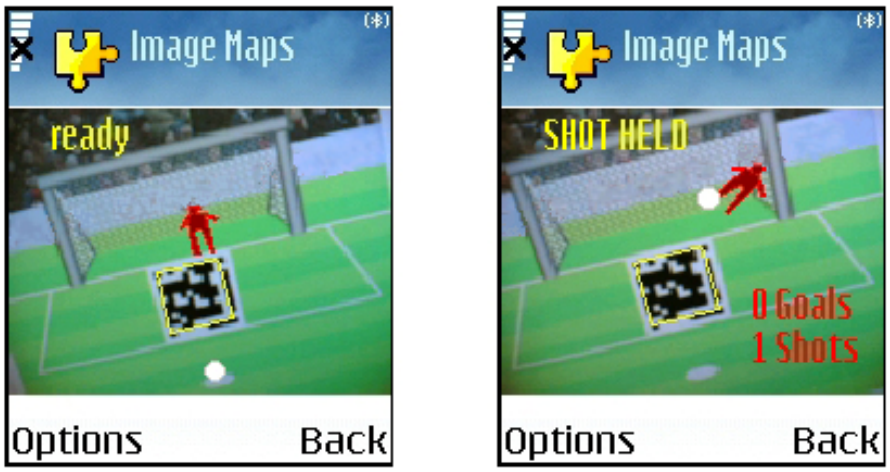


Figure 8.13: Screenshots of the penalty shootout game (a) before and (b) after kicking the ball (from [142]).

Prototype Game: Penalty Shootout

The prototype game presented here illustrates our approach. It is a simple penalty shootout game that consists of a printed playing surface on a box of cereals and virtual overlays generated by a camera phone. The playing surface shows a visual code in the center of the scene, a soccer field, a goal, the penalty spot, and spectators in the background. The display of the camera phone shows the virtual goal keeper, initially standing on the goal-line, a virtual ball, initially positioned on the penalty spot, and textual overlays (see Figures 8.13 and 8.14).

The code coordinate system of the central visual code is used to register the generated graphical overlays (the goal keeper and the ball) with elements in the camera image (the goal-line and the penalty spot). For large game boards, multiple visual codes would be required to ensure that a visual code is visible in the camera image at all times.

Figure 8.14: The penalty shootout game in action: The back side of a cereal box serves as a game board; the player performs rotation and tilting actions on the device to control the direction of the ball (from [142]).

The value of the visual code is a key to the game description. The game description is loaded via Bluetooth and then activated on the phone. In contrast to other handheld augmented reality games, such as the "Invisible Train,"[13] the user is not confined to control the game by extensive use of the keypad or touch-screen. Instead, the game was designed for embodied interaction. It applies visual code interaction primitives, such as rotation and tilting, that the player performs on the device (see Figure 8.14). The actions required to interact with the game are simple and intuitive spatial mappings. Aiming at the goal requires rotation and tilting. The amount of device rotation relative to the goal controls the horizontal shot direction. The amount of tilting of the device controls the vertical shot direction (high or flat shot). The user kicks the ball by pressing the joystick key. Upon pressing the joystick key, the ball quickly moves towards the goal (or misses it if the user did not aim right) and the goal keeper jumps to one of four possible positions to either catch or miss the ball. In the prototype implementation the goal keeper correctly guesses the shot direction in 15% of the cases and randomly chooses one of the four positions in 85% of the shots. The textual overlay shows the number of shots and goals.

[13] `www.studierstube.org/invisible_train`

The game instructions require just a few words, they can even be explored by users in a few trials without any instructions. Extremely short learning time and intuitive spatial mappings are crucial to reach a large target audience. Few people would be willing to read long instructions before playing a game like this. In a real deployment, the game would be associated with a contest. Players who perform well in the game might, for example by reaching a certain score, immediately enter a contest or win another box of cereal for free. The proposed concept allows for the design of exciting single player games on product packages. The computer assumes the role of the second player. Most conventional games on product packages are only playable and enjoyable with two players.

8.6 Related Work

Gershman et al. [81] define the situation encountered by a shopper, in which digital information about a product is inaccessible, as an *information discontinuity*. The information about the product would be useful in the physical context of the shop, but it is not available. There is a gap between the intention of the shopper – to get to know details about the product – and the possible actions in the given situation. In order to empower the user to do the relevant and effective action, Brody and Gottsman prototyped a portable Web-based comparison shopping assistant, called "Pocket BargainFinder" [39], which is a mobile phone equipped with a barcode scanner. Upon scanning a barcode on a product package, the device sends the identifier via the phone network. A server checks the price in several online stores and returns the results to the phone. The device translates the intention to buy into the action of buying, thereby creating a so-called "moment of value." The device can be seen as a situated cash register for the online store.

Newcomb et al. [150] investigate, how grocery shopping can be enhanced with a PDA that takes into account the physical space of the store and the shopping activity. They examine customer's grocery shopping habits and evaluate the usability of their prototypes. Moreover, they present a framework for designing and evaluating mobile applications that are situated in the retail arena.

Augmented reality research [7, 8] mainly focuses on visual augmentation of physical spaces through head-worn or hand-held displays. [207] is an example of recent work in *handheld* AR. The "Invisible Train"[14] is a game in which a virtual train is overlaid onto physical toy rail-tracks. The course of the train can be controlled by altering the track switches. The game is controlled using touchscreen input. Our work is more resembling embodied user interfaces and has a stronger focus on physical actions as input.

TUIs typically have fixed one-to-one mappings [187] between physical handles and virtual objects. This is also true for product packages in our concept, which have a fixed mapping to their virtual product counterpart. In contrast, traditional mouse-based interaction employs time-multiplexed one-to-many mappings, since the input vocabulary of a mouse (for example clicking and dragging) is extremely limited.

Beaudouin-Lafon [19] provides an interaction model called *instrumental interaction* that also captures TUIs. In this model, interaction between users and ob-

[14] `www.studierstube.org/invisible_train`

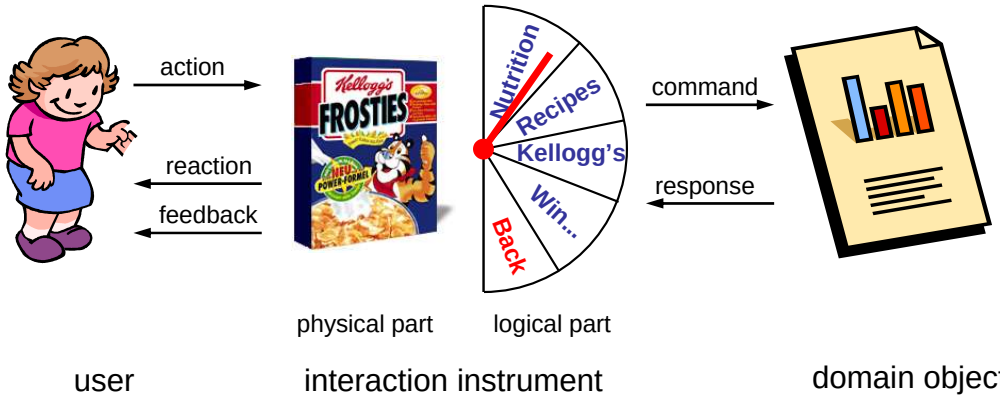


Figure 8.15: A product package as an interaction instrument mediates the interaction between the user and the virtual product counterpart (adapted from [19]).

jects of the application domain are mediated by *interaction instruments*, similar to conventional tools for interacting with the physical world. In the TUI approach to package interaction, the package and pie menu act as an interaction instrument that mediates between the user and the virtual product counterpart (see Figure 8.15). The instrument translates the physical actions of the user into commands on the virtual counterpart. Instruments are generally composed of a physical part, here the package, and a logical part, here the pie menu. The reaction of the instrument is the rotation of the pointer as the user rotates the package. The response of the domain object is the immediate display of the selected aspect of the virtual counterpart. In WIMP[15] interfaces, interaction instruments have to be explicitly activated. The mouse can be associated with different logical parts, like scrollbars and menus. The package instrument is continuously activated, since a fixed one-to-one mapping between the package and the pie menu exists.

Beaudouin-Lafon defines three properties to evaluate interaction instruments: *degree of indirection*, *degree of integration*, and *degree of compatibility*. The degree of indirection is a measure of the spatial and temporal offsets of an instrument. For the package instrument, the spatial offset is low to medium, since the logical part (the pie menu) is located next to the domain object (the product information on the screen). The temporal offset is low, because the domain object responds immediately to rotation changes by changing the displayed content. Hence our stationary approach has a low degree of indirection. The degree of compatibility is defined as the ratio between the number of DOFs of the logical part of the instrument and the number of DOFs of the input device. The pie menu has 1 DOF (rotation) and captures an explicit execution action for starting applications associated with a menu item. For the initial unconstrained setup, the product package has 4 DOFs (x, y, rotation, distance, we did not use tilting). This results in a degree of integration of 1/4. For the constrained setup, the product package has 1 DOF (rotation). Slightly lifting the package is used as an explicit execution gesture. We do not consider this binary selection as an additional DOF. The degree of integration for the constrained setup is thus 1/1, which is ideal. The degree of compatibility is a measure of the similarity between the user's physical actions and the response of the domain object. The rotation of the pointer of the pie menu

[15] windows, icons, menus, and pointing

follows the rotation of the package, but the mapping of rotation ranges to content is arbitrary. Therefore, the stationary setup has a low degree of compatibility.

In Fishkin's taxonomy [70],[16] our stationary approach to product package interaction would exhibit the noun metaphor, since "a virtual product counterpart in our system is like a physical product package in the real world." It also possesses the verb metaphor, since "rotating and lifting the product package in the real world is like rotating the pointer of the pie menu and selecting different aspects of the virtual product counterpart in our system." However, the analogy is very weak and rather an arbitrary mapping rather than being metaphoric. Consequently, the verb-and-noun metaphor is only very weak. With respect to embodiment, the stationary approach can either be seen as nearby or distant. A TUI has nearby embodiment, if the output is tightly coupled with the *focus* of the input. If the user operates (rotates and lifts) the product package only using tactile and kinesthetic feedback, while focusing visual attention at the display showing the virtual counterpart, this would be nearby embodiment. If users have to split attention between the input object they are operating on and the output on a nearby display to observe the results of their actions, this would be best described as distant embodiment. The shift from unconstrained 3-D input to operation on the glass panel resulted in a change from distant embodiment to nearby embodiment. With the 3-D input, users frequently split attention between the package they were holding to ensure it is within the camera range. With the 2-D input task, they kept visual attention on the display and did not look at the package anymore. However, first time users split their visual attention even for the 2-D task. This means that the level of embodiment is not a fixed property of the system, but a combination of the user and the system. Depending on their experience, users perceive the interface either as a distant embodiment or as a nearby embodiment.

ToolStone [167] is a cordless multi DOF (MDOF) input device for bimanual interaction. It operates on a tablet surface and senses rotation, flipping, and tilting. Depending on which face of the ToolStone touches the tablet, different functions are selected. In addition, 8 directions of rotation are sensed, similar to our rotation-based package interaction. In [167], Rekimoto and Sciammarella discuss two design principles that are relevant for *rich-action input*, i.e. interacting by physically changing the way an input device is held. The first design principle states that the input device's state should be perceived through touch. This relieves the user from spending visual attention to the input device. This is true for ToolStone and also for our constrained setup that operates on a glass panel. The second design principle says that the interaction space should be easily understandable. The pie menu effectively lays out the interaction space on the screen and presents all menu options of a layer at once. If the menu hierarchy is not too large, the interaction space is easily understandable. Fishkin [70] classifies ToolStone as metaphor verb, but not noun, since, for example, "moving the stone is like moving the camera," but the shape of the stone is not analogous to any real-world physical object. The embodiment is classified as nearby, even though the results of the interaction are visible on the screen, and not immediately near the stone. This can be justified with the single point of visual focus that ToolStone enables by providing haptic feedback.

[16] cf. Section 2.3.3

8.7 Summary

In this chapter we have shown two approaches to interact with virtual counterparts of product packages. The stationary interaction regards product packages as tangible user interfaces. We have developed two variants. The first is based on unconstrained 3-D input, the second restricts operation to a horizontal glass panel to simplify interaction demands. The constrained approach showed a more intuitive usability and is expressive enough to realize interaction with the virtual counterpart. The mobile interaction approach uses visual code image maps to access virtual product counterparts with camera phones. We have shown a handheld augmented reality game for product packages that uses embodied interaction with the device to control the game.

We have shown the state of the art in automatic product identification in related infrastructures. Current approaches focus on the supply chain and fail to take the needs of the consumer into account. We have outlined the components of a user-centered infrastructure that includes entities along the supply chain as well as third party service providers. Implementing such an infrastructure is an enormous effort, but might be worthwhile as it enables new ways to deliver information services to consumers by using product packages as entry points.

Chapter 9

Conclusions

In this chapter, we draw conclusions from the work described in the previous chapters. We summarize the main points of this dissertation and discuss its contributions. We conclude by outlining open issues and suggesting areas for future work.

The overall goal of this work was to explore and develop new ways of interaction between users and objects in their environment. Linking the physical and the virtual world is a central research issue in pervasive and ubiquitous computing. Allowing users to effectively interact with combined physical and virtual worlds is an essential prerequisite for the realization of the ubiquitous computing vision. To this end, we have proposed the use of camera-equipped mobile devices as mediators between the physical and the virtual realm. Mobile handheld devices have many advantages in terms of the pragmatics of their use. They are continuously available in many everyday situations, since they are carried or worn by their users throughout the day. Mobile phones in particular have become the most ubiquitous communication devices of our days. From a technical perspective, these devices offer ever increasing processing power, constant wireless connectivity over both long-distance and short-distance communication links, and they offer multimodal input and restricted output capabilities. Last but not least, camera-equipped devices allow the usage of the camera as an additional input channel. To that end, it is possible to process camera input on the device itself using a wide range of available algorithms.

The main research question thus was how to turn camera-equipped mobile devices into effective and useful interaction devices for physical entities. The convenient form factor of handheld devices in general, and small camera phones in particular, allows to physically handle these devices in great variety of ways. They can be targeted at objects, swept across a surface, or waved through the air. The distance, rotation, and tilting relative to targeted objects can be precisely controlled by human users. We thus proposed to use camera phones and similar devices as *symbolic magnifying glasses* that provide an augmented view of the scene that is focused. In this paradigm, the output of the device not only depends on the targeted object, but also on the orientation of the device relative to the object. Moving the computing device has analogies to moving an ordinary optical magnifying glass, in order to change the focused area and the viewing perspective. This leads to the concept of *embodied user interfaces*. The device embodies a well-known object. Ideally, users can draw analogies from their real-world experience to explore the virtual functionality of the device.

9.1 Summary

We began this dissertation by reviewing the vision and technological enablers of ubiquitous and pervasive computing (Chapter 2). In this field of research, human interaction ranges from calm computing and unobtrusive background assistance to explicit foreground interaction. We have organized the discussion of linking the physical and the virtual world to "linking in the large," i.e. the design of situated information spaces and infrastructures, and "linking in the small," i.e. the fine-grained and natural interaction with a single artifact. As an instance of "linking in the large," we described the ETHOC infrastructure for a smart campus environment and associated concepts. In the section on "linking in the small" we introduced the notions of tangible and embodied user interfaces, augmented reality, and marker-based interaction. *Marker-based interaction* is concerned with the space of interaction in the vicinity of a visual marker, i.e. the range within which it can be recognized from a camera-equipped device.

As the technical basis for the interaction techniques introduced later we have then presented a visual marker system for camera phones, called *visual codes* (Chapter 3). Visual codes and their recognition algorithm are designed so as to be efficiently recognizable on resource constrained devices with low-resolution integrated cameras. Visual codes are recognizable from almost any arbitrary orientation to account for the inherent mobility of handheld devices. Each code defines its own local coordinate system that is independent of perspective distortions. The visual code system provides the rotation, tilting, distance, and target point as parameters that can be used for interaction. Additionally, the movement of the device relative to the background is detected by the camera. This enables further interaction possibilities. We have shown how the parameters can be used in a number of example applications. We then have introduced the notion of visual code sequences. These are periodically repeating sequences of visual codes that can be used in conjunction with active displays. They provide an anonymous undirectional communication channel and increase the amount of data that can be transferred via visual codes.

In Chapter 4 we have presented a conceptual framework for marker-based interaction. It allows to define physical gestures, called *interaction primitives*, that can be flexibly combined. Iconic and auditory cues indicate interaction possibilities and guide users in their interactions. We introduced the notion of *visual code image maps*. These are printed or displayed surfaces that contain one or more visual codes and sensitive areas next to these codes. The areas are associated with online information that is retrieved by the mobile device. The orientation of the device is used as an additional parameter, which allows to retrieve multiple information aspects per area. A language to specify visual code image maps and corresponding tools and interpreters have been described. The language maps discrete device postures within a 3-D state space to information items and operations on the device. A usability study has validated the viability of the approach. As a further abstraction step, visual code widgets have been proposed in Chapter 5. They are printable user interface elements that provide clear affordances and act as building blocks for marker-based interactions. We have implemented a visual code widget renderer and generator as a proof of concept.

We analyzed interaction with large-scale public displays in Chapter 6 and proposed the use of personal devices and two complementary interaction techniques for this domain. The *sweep* interaction technique uses relative movement detection for relative positioning of a cursor on a large display. The *point & shoot* used a visual code grid for absolute positioning.

User-defined digital annotations to objects are an alternative to predefined contents. In Chapter 7 we described two techniques for using pen-based camera phones and PDAs as annotation devices. The first technique uses the code coordinate systems of visual codes for generating annotations to different parts of printed photographs. The second one operates without visual codes and matches interactively segmented images to templates on a server.

Chapter 8 was devoted to product packaging, as one exemplary class of physical objects that may be associated with online content using the developed techniques. We presented a stationary interaction approach, in which product packages are used as tangible user interfaces, and a mobile interaction approach, in which product packages are used as the backgrounds of handheld augmented reality games and animations.

9.2 Contributions and Results

The main contribution of this dissertation is a spectrum of concepts and techniques for marker-based interaction. Our system enables embodied interaction with visual markers and thus turns camera phones and similar devices into versatile mediators between the real and the virtual world. It detects the orientation of the mobile device relative to the marker and provides the basis for augmenting the camera image with precisely aligned graphical overlays. The framework of interaction primitives, the visual code widgets, the interaction techniques for large public displays, and the annotation techniques have proven the viability of the concept of marker-based interaction and shown its versatility. A number of example applications have served as a proof of concept and as illustrations of its general applicability.

Specifically, the individual contributions can be summarized as follows:

- The entry points concept for signaling the availability of physical hyperlinks in situated information spaces.
- A visual code system for linking computation and services to physical objects and passive media as well as projected and electronic displays.
- A conceptual framework of physical interaction primitives, their combination, associated interaction cues, and a corresponding specification language.
- A set of visual code widgets that serve as marker-based interaction elements with clear affordances.
- Two interaction techniques for large-scale displays; *sweep* for relative positioning and *point & shoot* for absolute positioning of a pointer on a display.
- Two annotation techniques for camera-equipped devices with pen-based input. One based on visual codes and their coordinate systems and one using

4-point correspondences between objects in the camera view and template images.

- Stationary and mobile approaches to product packaging that use product packages as tangible user interfaces and backgrounds for handheld augmented reality games, respectively.

Major parts of the research results of this thesis have been presented at workshops and conferences and have been published in journals and proceedings: [10, 11, 12, 30, 169, 170, 171, 172, 173, 175, 176, 177, 178].

9.3 Open Issues and Future Work

There are a number of open issues – and opportunities – that have been touched upon in this thesis and that deserve further investigation.

The visual code system works reliably under different lighting conditions. However, the algorithm is currently limited to black-and-white codes with a white quiet zone surrounding them. For steerable projected displays as presented in [157] this is less suitable, since black means no projected light and white means presence of projected light. It would thus be simpler to use black for the surrounding quiet zone in the case of projected codes.

Moreover, visual codes are only detectable on relatively flat surfaces. If codes are strongly bended, they can no longer be reliably detected. When working with flexible product packages, like bags, or with curved surfaces, like bottles, this becomes an issue. In such situations it would be desirable to have codes that can be detected on any arbitrarily shaped surface.

For applications like games, in which multiple visual codes are detected at once, and in which the relative placement of these codes is important, suitable abstractions would be desirable. It would be nice to be able to reason with geometric relationships such as “close to,” “in front of,” “in neighborhood to,” etc.

The visual code widgets have been developed, but not yet used within a real usability study that would evaluate broader usage issues in the context of a complete application.

The *sweep* technique for interacting with large public displays is not fully satisfying on current camera phone hardware. It would be interesting to try a hardware setup that emulates the hardware capabilities that can be expected in a few years from now.

Handheld augmented reality games could be extended beyond product packages to include other background media as well. Examples are concert tickets, flyers, newspapers, posters, and large public displays. The prototype game was developed on a very low level of abstraction using the visual code parameters directly. A game and animation description language that operated on a higher level of abstraction and that allowed to rapidly specify games would be desirable.

From a more fundamental perspective, the dexterity required for marker-based interaction in terms of hand-eye coordination and motor skills has to be investigated in more detail and for a more diverse user group.

Appendix A

XML Schema for Visual Code Image Maps

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:element name="ImageMap">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Area" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="imageName" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Area">
    <xs:complexType>
      <xs:sequence>
        <xs:choice>
          <xs:element ref="Rectangle" maxOccurs="unbounded"/>
          <xs:element ref="Ellipse" maxOccurs="unbounded"/>
          <xs:element ref="Polygon" maxOccurs="unbounded"/>
        </xs:choice>
        <xs:element ref="Rule" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Rectangle">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Point" minOccurs="2" maxOccurs="2"/>
      </xs:sequence>
      <xs:attribute name="coordinateSystem"
                    type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Point">
    <xs:complexType>
      <xs:attribute name="x" type="xs:decimal" use="required"/>
```

```
      <xs:attribute name="y" type="xs:decimal" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Ellipse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Point" minOccurs="2" maxOccurs="2"/>
      </xs:sequence>
      <xs:attribute name="coordinateSystem"
                    type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Polygon">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Point" minOccurs="3" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="coordinateSystem"
                    type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Rule">
    <xs:complexType>
      <xs:sequence>
        <xs:group ref="Constraints"/>
        <xs:choice>
          <xs:element ref="Information"/>
          <xs:element ref="Action"/>
        </xs:choice>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:group name="Constraints">
    <xs:all minOccurs="0">
      <xs:element ref="Rotation" minOccurs="0"/>
      <xs:element ref="Distance" minOccurs="0"/>
      <xs:element ref="Tilts" minOccurs="0"/>
      <xs:element ref="Keystrokes" minOccurs="0"/>
      <xs:element ref="Stay" minOccurs="0"/>
    </xs:all>
  </xs:group>

  <xs:element name="Rotation">
    <xs:complexType>
      <xs:attribute name="category" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="absolute"/>
            <xs:enumeration value="relative"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="start" use="required">
```

```
      <xs:simpleType>
        <xs:restriction base="xs:integer">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="360"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="end" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:integer">
          <xs:minInclusive value="0"/>
          <xs:maxInclusive value="360"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="Distance">
  <xs:complexType>
    <xs:attribute name="start"
                  type="xs:nonNegativeInteger" use="required"/>
    <xs:attribute name="end"
                  type="xs:nonNegativeInteger" use="required"/>
  </xs:complexType>
</xs:element>

<xs:element name="Tilts">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Tilt" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Tilt">
  <xs:complexType>
    <xs:attribute name="value" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="n"/>
          <xs:enumeration value="ne"/>
          <xs:enumeration value="e"/>
          <xs:enumeration value="se"/>
          <xs:enumeration value="s"/>
          <xs:enumeration value="sw"/>
          <xs:enumeration value="w"/>
          <xs:enumeration value="nw"/>
          <xs:enumeration value="c"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="Keystrokes">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element ref="Keystroke" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Keystroke">
  <xs:complexType>
    <xs:attribute name="value" use="required" type="xs:string"/>
  </xs:complexType>
</xs:element>

<xs:element name="Stay">
  <xs:complexType>
    <xs:attribute name="start" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="0.0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="end" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:decimal">
          <xs:minInclusive value="0.0"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="Information">
  <xs:complexType>
    <xs:group ref="InformationItems"/>
    <xs:attribute name="dynamic" type="xs:boolean"
                  default="false" use="optional"/>
  </xs:complexType>
</xs:element>

<xs:group name="InformationItems">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="IconicCue"/>
    <xs:element ref="AuditoryCue"/>
    <xs:element ref="TactileCue"/>
    <xs:element ref="Line"/>
    <xs:element ref="DrawPolygon"/>
  </xs:choice>
</xs:group>

<xs:element name="IconicCue">
  <xs:complexType>
    <xs:attribute name="name" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="DistanceBoth"/>
          <xs:enumeration value="DistanceCloser"/>
          <xs:enumeration value="DistanceFarther"/>
          <xs:enumeration value="Keystroke"/>
          <xs:enumeration value="RotationBothDir"/>
```

```
            <xs:enumeration value="RotationCW"/>
            <xs:enumeration value="RotationCCW"/>
            <xs:enumeration value="Stay"/>
            <xs:enumeration value="TiltC"/>
            <xs:enumeration value="TiltE"/>
            <xs:enumeration value="TiltN"/>
            <xs:enumeration value="TiltNS"/>
            <xs:enumeration value="TiltS"/>
            <xs:enumeration value="TiltW"/>
            <xs:enumeration value="TiltWE"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="AuditoryCue">
    <xs:complexType>
      <xs:attribute name="frequency" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="20.0"/>
            <xs:maxInclusive value="20000.0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="duration" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0.0"/>
            <xs:maxInclusive value="60.0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="TactileCue">
    <xs:complexType>
      <xs:attribute name="duration" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0.0"/>
            <xs:maxInclusive value="60.0"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="intensity" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:integer">
            <xs:minInclusive value="-100"/>
            <xs:maxInclusive value="100"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
```

```
  <xs:element name="Line">
    <xs:complexType>
      <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="DrawPolygon">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Point" minOccurs="3" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="coordinateSystem" type="xs:string"
                    use="required"/>
      <xs:attribute name="penColor" type="xs:string" use="optional"/>
      <xs:attribute name="brushColor" type="xs:string" use="optional"/>
      <xs:attribute name="penSize" type="xs:nonNegativeInteger"
              use="optional"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="Action">
    <xs:complexType>
      <xs:group ref="InformationItems"/>
      <xs:attribute name="functionName" type="xs:string" use="required"/>
      <xs:attribute name="arguments" type="xs:string" use="optional"/>
      <xs:attribute name="body" type="xs:string" use="optional"/>
      <xs:attribute name="phoneNumber" type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

Appendix B

Usability Evaluation of the Interaction Primitives

This appendix is taken from [228]. It shows the material presented to users during the usability tests.

B.1 Questionnaire

Please answer the following questions:

1. What is your gender?
   - ☐ Male
   - ☐ Female
2. How old are you?
   - ☐ Between 8 and 16
   - ☐ Between 17 and 25
   - ☐ Between 26 and 35
   - ☐ Between 36 and 45
   - ☐ Older than 45
3. Do you use a mobile phone?
   - ☐ Yes
   - ☐ No
4. What features of your mobile phone do you use?
   - ☐ Make phone calls.
   - ☐ Write SMS.
   - ☐ Take pictures with the mobile phone camera.
   - ☐ Access information over WAP.
   - ☐ Play games on the mobile phone.

- ☐ Use the built-in address book and calendar.

5. How often do you write short messages (SMS) with your mobile phone?
   - ☐ More than three times a day
   - ☐ Daily
   - ☐ Weekly
   - ☐ Monthly
   - ☐ Never
6. How often do you use a personal computer?
   - ☐ Daily
   - ☐ Weekly
   - ☐ Monthly
   - ☐ Never
7. How often do you play computer games?
   - ☐ Daily
   - ☐ Weekly
   - ☐ Monthly
   - ☐ Never

B.2 Tasks

B.2.1 Part 1: User Interaction Evaluation

You have seen and used all user interactions. In the next tasks, you have to apply these user interaction technique to find a *secret number* and a *secret letter* for each tasks. If a tasks seems too complicated, then skip it and try the next one. Remember that the user interactions and the visual code system is tested and **not** you.

Task 1

Find the secret number and the secret letter:

Secret number: ☐

Secret letter: ☐

Task 2

Find the secret number and the secret letter:

Secret number: ☐

Secret letter: ☐

Task 3

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 4

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 5

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 6

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 7

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 8

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 9

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 10

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 11

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 12

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 13

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 14

Find the secret number and the secret letter:

Secret number:

Secret letter:

Task 15

Find the secret number and the secret letter:

Secret number:

Secret letter:

B.2.2 Part 2: Campus Map Application

You have seen two application scenarios and have used all user interactions. Now, we will use a visual code application. Remember that this visual code application is evaluated and **not** you. Have fun!

Task 16

You see a visual code image map application and have your mobile phone. Find out what this visual code application is about. What does it do? How does it help you?

Task 17

Find the building LEH with help of the mobile phone.

Task 18

Find the building CLT with help of the mobile phone.

Task 19

Find the building LFO with help of the mobile phone.

Task 20

Find the building CNB with help of the mobile phone.

B.3 Final Interview

1. We have seen two application scenarios at the beginning and have used another one at the end. What is your first impression of using the visual code system to access information? Would you use it, if it were free of any charge? Would you use it, if it cost a small fee (for example like the cost of sending a text message)?

2. Which user interaction do you like best (the interviewer explains each user interaction again)?

   - ☐ Rotation
   - ☐ Tilt
   - ☐ Keystroke
   - ☐ Stay at the same area

3. Why do you like this user interaction best?

4. What user interaction is the most difficult user interaction for you?

   - ☐ Rotation
   - ☐ Tilt
   - ☐ Keystroke
   - ☐ Stay at the same area

5. Why is this the most difficult user interaction for you?

6. What user interaction is the best one to show information in a table?

   - ☐ Rotation
   - ☐ Tilt

- ☐ Keystroke
- ☐ Stay at the same area

7. Why is this the best user interaction to show information in a table?

8. What user interaction is the most difficult user interaction to show information in a table?

   - ☐ Rotation
   - ☐ Tilt
   - ☐ Keystroke
   - ☐ Stay at the same area

9. Why is this the most difficult user interaction to show information in a table?

10. Does the display size suffice to recognize table cells and the written text?

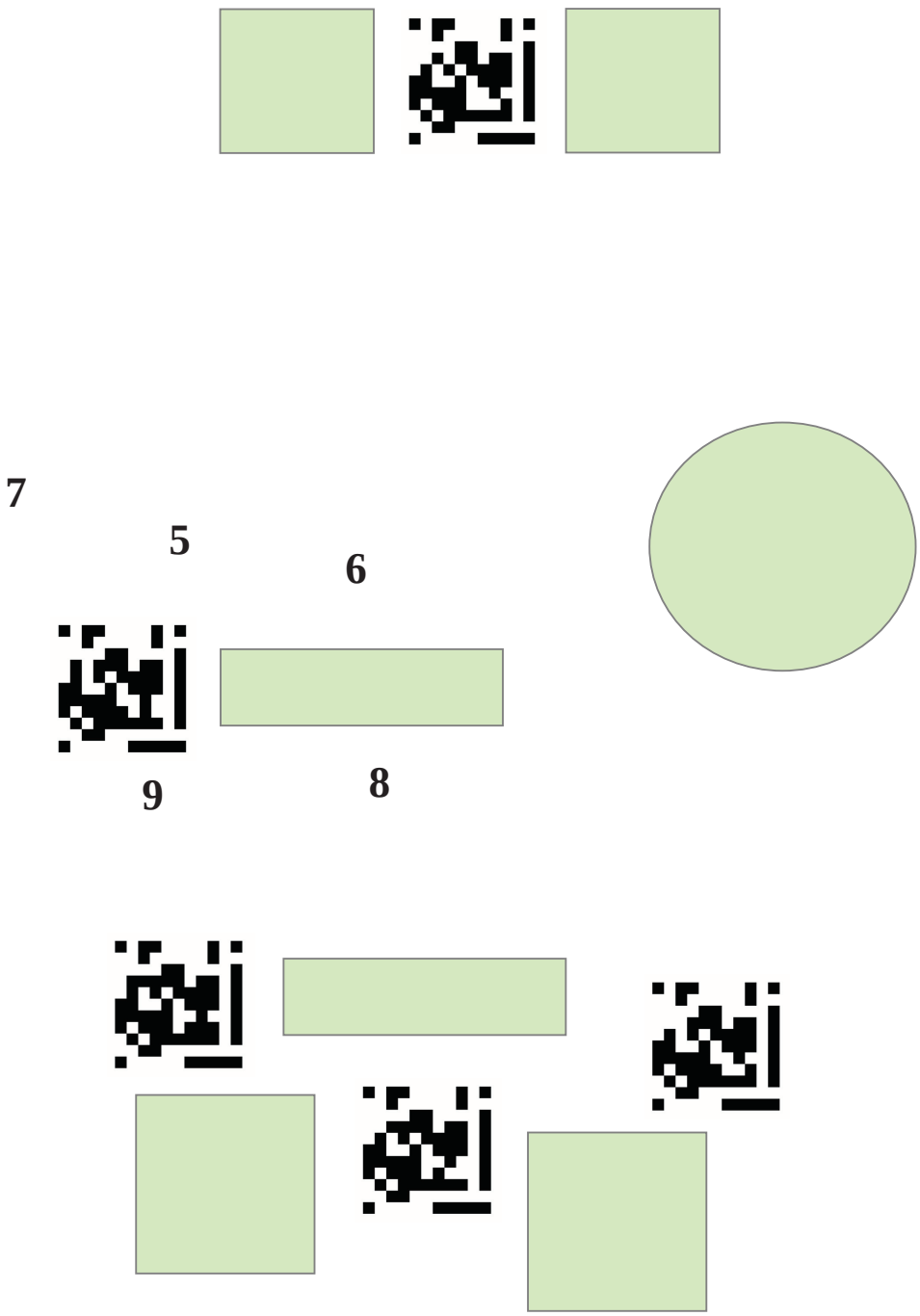
11. What do you think of the visual code information map?

12. What did you dislike about this usability study?

13. What did you like about this usability study?

B.4 Evaluation Tasks for Interaction Primitives

(These image maps were actually presented to users in the depicted layout on DIN A4 paper.)



Task 1:

secret number
& secret letter

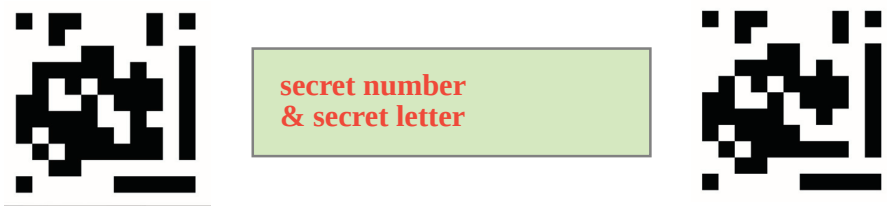
Task 2:

secret number
& secret letter

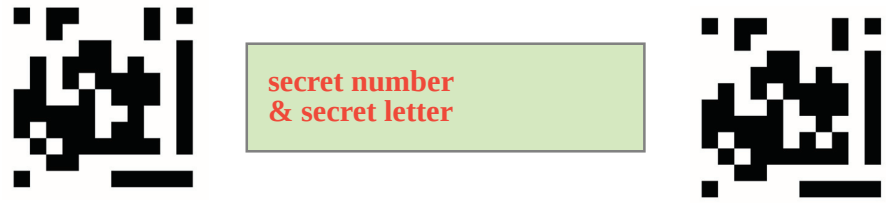
Task 3:

secret number
& secret letter

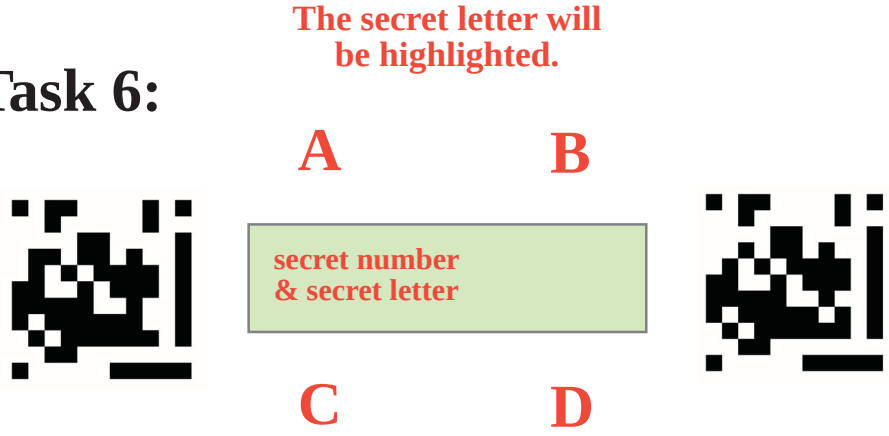
Task 4:



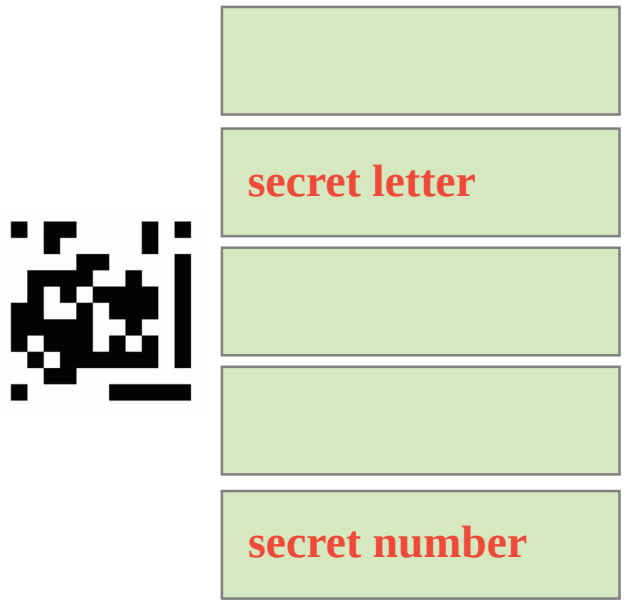
Task 5:



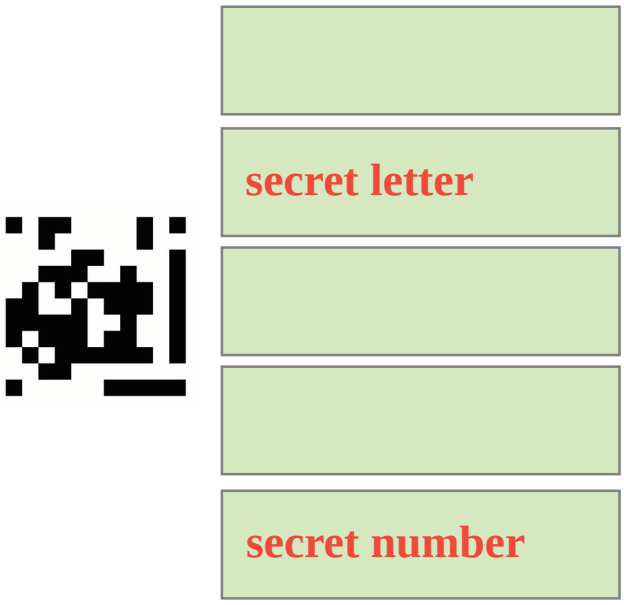
Task 6:



Task 7:



Task 8:



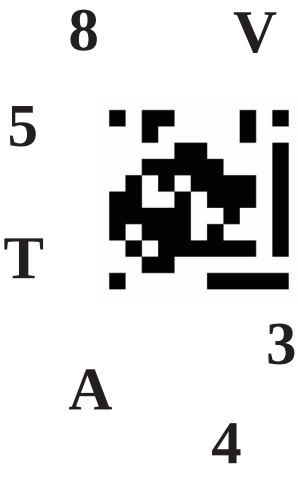
Task 9:

secret letter

secret number

Task 10:

The secret letter and the secret number will be highlighted.



secret letter

secret number

Task 11:

secret letter

secret number

Task 12:

secret letter

secret number

Task 13:

secret number
& secret letter

Task 14:

secret number
& secret letter

Task 15:

secret number
& secret letter

B.5 Campus Map Application

(The campus map was presented to users stuck to a wall and printed in DIN A1 format.)

Appendix C

Visual Code Widget Creation Tool

This appendix describes the usage of two Java command line tools to generate PNG images of visual code widgets.

C.1 Visual Code Menu Creation

Vertical menus

- java MenuEditor <id> <item height> <item count left> <item count right>
- Example: java MenuEditor 0x123 1 3 4

Square pie menus

- java MenuEditor <id> <bit map of available menu items>
- Example: java MenuEditor 0xabc 01010101

Circular pie menus

- java MenuEditor <id> <outer radius> <item count>
- Example: java MenuEditor 0x1234567890abcdef 3 6

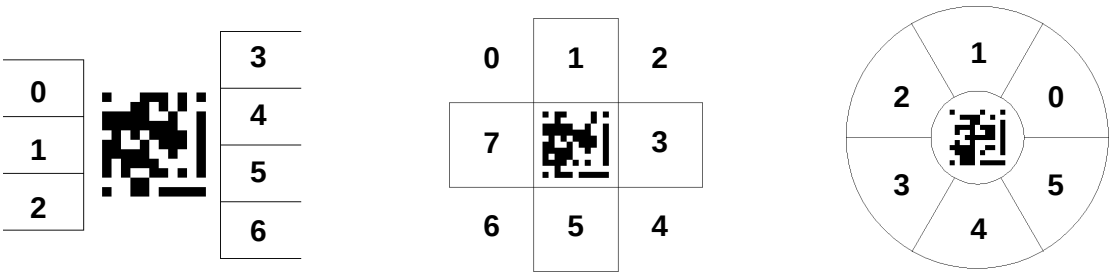


Figure C.1: The results of the above example tool invocations.

C.2 Selection and Data Entry Widget Creation

Check boxes

- java WidgetEditor <id> c <item height> <item count>
- Example: java WidgetEditor 0x123 c 1 5

Radio buttons

- java WidgetEditor <id> r <item height> <item count>
- Example: java WidgetEditor 0x123 r 1 5

Sliders

- java WidgetEditor <id> s <h,v> <c,d> <o,p,q,n> <tick distance> <tick count> [<bound> <step>]
- Example 1: java WidgetEditor 0xabc s v c p 2 5
- Example 2: java WidgetEditor 0xabc s h d o 2 5
- Example 3: java WidgetEditor 0xabc s v c n 5 3 -1000 1000

Dials

- java WidgetEditor <id> d <c,d> <o,p,q,n> <tick distance> <tick count> [<bound> <step>]
- Example 1: java WidgetEditor 0xabc d c p 5
- Example 2: java WidgetEditor 0xabc d c n 5 -10 3

Free-form input

- java WidgetEditor <id> f <l,t> <frame width> <frame height>
- Example: java WidgetEditor 0xabc f l 8 5

Relative movement detection

- java WidgetEditor <id> m
- Example: java WidgetEditor 0xabc m

Text input

- java WidgetEditor <id> t
- Example: java WidgetEditor 0xabc t

Bibliography

[1] Gregory D. Abowd, Liviu Iftode, and Helena Mitchell. Guest Editors' Introduction: The Smart Phone – A First Platform for Pervasive Computing. *IEEE Pervasive Computing*, 4(2):18–19, April 2005.

[2] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact.*, 7(1):29–58, 2000.

[3] Johnny Accot and Shumin Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 73–80. ACM Press, 2002.

[4] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a Sentient Computing System. *IEEE Computer*, 34(8):50–56, August 2001.

[5] Heikki Ailisto, Johan Plomp, Lauri Pohjanheimo, and Esko Strömmer. A Physical Selection Paradigm for Ubiquitous Computing. In Emile Aarts, editor, *Proceedings of EUSAI 2003, LNCS 2875*, pages 372–383. Springer-Verlag, October 2003.

[6] Kiyoharu Aizawa, Kentaro Kakami, and Koji Nakahira. Ubiquitous Displays for Cellular Phone Based Personal Environments. In Y-C Chen, L-W Chang, and C-T Hsu, editors, *IEEE Pacific Rim Conference on Multimedia (PCM2002)*, pages 25–32. LNCS 2532, Springer, 2002.

[7] Ronald Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.

[8] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent Advanced in Augmented Reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, November 2001.

[9] Ravin Balakrishnan and I. Scott MacKenzie. Performance differences in the fingers, wrist, and forearm in computer input control. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 303–310, New York, NY, USA, 1997. ACM Press.

[10] Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Sweep and Point & 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, April 2005. ACM Press.

[11] Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Mobile Phones as Pointing Devices. In *Pervasive 2005 Workshop on Pervasive Mobile Interaction Devices (PERMID)*, pages 27–30, May 2005.

[12] Rafael Ballagas, Michael Rohs, Jennifer G. Sheridan, and Jan Borchers. BYOD: Bring Your Own Device. In *UbiComp 2004 Workshop on Ubiquitous Display Environments*, Nottingham, UK, September 2004.

[13] Kaspar Baltzer. Portierung eines Visual Code Systems für Mobiltelefone auf Pocket PC Phone Edition. Semester project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2005.

[14] Guruduth Banavar and Abraham Bernstein. Software infrastructure and design challenges for ubiquitous computing applications. *Commun. ACM*, 45(12):92–96, 2002.

[15] Rob Barrett and Paul P. Maglio. Informative Things: How to Attach Information to the Real World. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 81–88, 1998.

[16] Joel F. Bartlett. Rock 'n' Scroll Is Here to Stay. *IEEE Comput. Graph. Appl.*, 20(3):40–45, 2000.

[17] John Barton, Patrick Goddi, and Mirjana Spasojevic. Creating and Experiencing Ubimedia. HP Labs Technical Report HPL-2003-38, 2003.

[18] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus plus context screens: Combining display technology with visualization techniques. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 31–40. ACM Press, 2001.

[19] Michel Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 446–453, New York, NY, USA, 2000. ACM Press.

[20] Marcel Beer. PhotoWall: Interaktion mit Fotos auf Wanddisplays. Semester project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2005.

[21] Victoria Bellotti, Maribeth Back, W. Keith Edwards, Rebecca E. Grinter, Austin Henderson, and Cristina Lopes. Making sense of sensing systems: five questions for designers and researchers. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–422, New York, NY, USA, 2002. ACM Press.

[22] Olav W. Bertelsen and Christina Nielsen. Augmented reality as a design tool for mobile interfaces. In *DIS '00: Proceedings of the conference on Designing interactive systems*, pages 185–192, New York, NY, USA, 2000. ACM Press.

[23] Eric A. Bier and Steven Freeman. MMM: A user interface architecture for shared editors on a single screen. In *UIST '91: Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 79–86, New York, NY, USA, 1991. ACM Press.

[24] Eric A. Bier, Maureen C. Stone, Ken Fishkin, William Buxton, and Thomas Baudel. A taxonomy of see-through tools. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 358–364. ACM Press, 1994.

[25] Biörn Biörnstad. Jini Dienstvermittlung und Bluetooth Service Discovery Protokoll. Semester project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2001.

[26] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Allgegenwart und Verschwinden des Computers – Leben in einer Welt smarter Alltagsdinge. In Ralf Grötker, editor, *Privat! Kontrollierte Freiheit in einer vernetzten Welt*, pages 195–245. Heise-Verlag, March 2003.

[27] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Disappearing Computers Everywhere – Living in a World of Smart Everyday Objects. In *Proc. of New Media, Technology and Everyday Life in Europe Conference*, London, UK, April 2003.

[28] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Living in a World of Smart Everyday Objects – Social, Economic, and Ethical Implications. *Journal of Human and Ecological Risk Assessment*, 10(5):763–786, October 2004.

[29] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, and Michael Rohs. Social, Economic, and Ethical Implications of Ambient Intelligence and Ubiquitous Computing. In W. Weber, J. Rabaey, and E. Aarts, editors, *Ambient Intelligence*, pages 5–29. Springer-Verlag, 2005.

[30] Jürgen Bohn and Michael Rohs. Klicken in der realen Welt. Konferenz Mensch und Computer 2001, Workshop Mensch-Computer-Interaktion in allgegenwärtigen Informationssystemen, March 2001.

[31] Nicolas Born. Marker-basierte Interaktion mit Augmented Board Games. Semester project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2005.

[32] Stephen Brewster and Lorna M. Brown. Tactons: Structured tactile messages for non-visual information display. In *CRPIT '04: Proceedings of the fifth conference on Australasian user interface*, pages 15–23, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[33] Stephen A. Brewster. Using Nonspeech Sounds to Provide Navigation Cues. *ACM Transactions on Computer-Human Interaction*, 5(3):224–259, 1998.

[34] Harry Brignull, Shahram Izadi, Geraldine Fitzpatrick, Yvonne Rogers, and Tom Rodden. The introduction of a shared interactive surface into a communal space. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 49–58, New York, NY, USA, 2004. ACM Press.

[35] Harry Brignull and Yvonne Rogers. Enticing People to Interact with Large Public Displays in Public Spaces. In *Proceedings of Human-Computer Interaction – INTERACT'03*, pages 17–24. IOS Press, September 2003.

[36] David Brock. The Electronic Product Code – A Naming Scheme for Physical Objects. MIT Auto-ID Center, archive.epcglobalinc.org/publishedresearch/MIT-AUTOID-WH-002.pdf, January 2001.

[37] David Brock. The Physical Markup Language. MIT Auto-ID Center, archive.epcglobalinc.org/publishedresearch/MIT-AUTOID-WH-003.pdf, February 2001.

[38] Christina Brodersen and Jannie Friis Kristensen. Interaction through negotiation. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 259–268, New York, NY, USA, 2004. ACM Press.

[39] Adam B. Brody and Edward J. Gottsman. Pocket Bargain Finder: A Handheld Device for Augmented Commerce. In Hans W. Gellersen, editor, *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 44–51, London, UK, September 1999. LNCS 1707, Springer.

[40] William Buxton. A three-state model of graphical input. In *Proceedings of the IFIP TC13 Third Interantional Conference on Human-Computer Interaction*, pages 449–456. North-Holland, 1990.

[41] Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 95–100. ACM Press, 1988.

[42] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A morphological analysis of the design space of input devices. *ACM Trans. Inf. Syst.*, 9(2):99–122, 1991.

[43] Scott Carter, Elizabeth Churchill, Laurent Denoue, Jonathan Helfman, and Les Nelson. Digital graffiti: Public annotation of multimedia content. In *Extended abstracts of the 2004 conference on Human factors and computing systems*, pages 1207–1210. ACM Press, 2004.

[44] Deborah Caswell and Philippe Debaty. Creating Web Representations for Places. In P. Thomas and Hans W. Gellersen, editors, *HUC '00: Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, pages 114–126. Springer-Verlag, September 2000.

[45] Harry Chen, Tim Finin, and Anupam Joshi. An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3):197–207, May 2004.

[46] Elizabeth M. Churchill, Les Nelson, Laurent Denoue, and Andreas Girgensohn. The Plasma Poster Network: Posting Multimedia Content in Public Places. In *Proceedings of Human-Computer Interaction – INTERACT'03*, pages 599–606. IOS Press, September 2003.

[47] Alan Cooper. *About Face: The Essentials of User Interface Design.* IDG Books Worldwide, Inc., August 1995.

[48] Nigel Davies and Hans-Werner Gellersen. Beyond Prototypes: Challenges in Deploying Ubiquitous Systems. *IEEE Pervasive Computing*, 1(1):26–35, January 2002.

[49] Marc Davis. Mobile media metadata: Metadata creation system for mobile images. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 936–937. ACM Press, 2004.

[50] Marc Davis, Simon King, Nathan Good, and Risto Sarvas. From context to content: Leveraging context to infer media metadata. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 188–195. ACM Press, 2004.

[51] F. Dawson and T. Howes. vCard MIME Directory Profile. Internet RFC 2426, September 1998.

[52] F. Dawson and D. Stenerson. Internet Calendaring and Scheduling Core Object Specification (iCalendar). Internet RFC 2445, November 1998.

[53] Diego López de Ipiña, Paulo R. S. Mendonça, and Andy Hopper. TRIP: A Low-Cost Vision-Based Location System for Ubiquitous Computing. *Personal Ubiquitous Comput.*, 6(3):206–219, 2002.

[54] Philippe Debaty and Deborah Caswell. Uniform Web Presence Architecture for People, Places, and Things. *IEEE Personal Communications*, 8(4):46–51, August 2001.

[55] Corsin Decurtins, Moira C. Norrie, and Beat Signer. Digital annotation of printed documents. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 552–555, New York, NY, USA, 2003. ACM Press.

[56] Anind K. Dey. *Providing Architectural Support for Building Context-Aware Applications.* PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

[57] Anind K. Dey. Understanding and Using Context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001.

[58] Paul Dietz and Darren Leigh. DiamondTouch: A multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226. ACM Press, 2001.

[59] Sabine Do-Thuong. Magnifying Glass Metaphor for the Interaction with Virtual Counterparts. Master's thesis, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2002.

[60] ECMA International. Near Field Communication. White paper. Ecma/TC32-TG19/2004/1. Available at: www.ecma-international.org/activities/Communications/2004tg19-001.pdf, 2004.

[61] Ernest Edmonds, Greg Turner, and Linda Candy. Approaches to interactive art systems. In *Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Austalasia and Southe East Asia*, pages 113–117. ACM Press, 2004.

[62] Emnid-Studie. Verbraucherwünsche gehen über neue EU-Richtlinie hinaus – Verbraucher fordern Klartext bei der Kennzeichnung von Lebensmitteln. Survey for FRoSTA AG, available at: www.presseportal.de/print.htx?nr=458811, June 2003.

[63] European Commission, Directorate-General for Press and Communication. From farm to fork – Safe food for Europe's consumers. Europe on the move series, Office for Official Publications of the European Communities, Luxembourg, europa.eu.int/comm/publications, July 2004.

[64] European Parliament and Council of the European Union. Regulation (EC) No 178/2002 of the European Parliament and of the council of 28 January 2002 laying down the general principles and requirements of food law, establishing the European Food Safety Authority and laying down procedures in matters of food safety. Official Journal of the European Communities, February 2002.

[65] European Parliament and Council of the European Union. Directive 2003/89/EC of the European Parliament and of the Council of 10 November 2003 amending Directive 2000/13/EC as regards indication of the ingredients present in foodstuffs. Official Journal of the European Union, November 2003.

[66] Steven K. Feiner. Augmented Reality: A New Way of Seeing. *Scientific American*, 286(4):48–55, April 2002.

[67] Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[68] Alois Ferscha and Simon Vogl. Pervasive Web Access via Public Communication Walls. In *Proceedings of the First International Conference on Pervasive Computing*, volume 2414 of *Lecture Notes in Computer Science*, pages 84–97, Zurich, August 2002. Springer-Verlag.

[69] Klaus Finkenzeller. *RFID-Handbook*. John Wiley & Sons, 2nd edition, 2003.

[70] Kenneth P. Fishkin. A taxonomy for and analysis of tangible interfaces. *Personal Ubiquitous Comput.*, 8(5):347–358, 2004.

[71] Kenneth P. Fishkin, Anuj Gujar, Beverly L. Harrison, Thomas P. Moran, and Roy Want. Embodied user interfaces for really direct manipulation. *Commun. ACM*, 43(9):74–80, 2000.

[72] Kenneth P. Fishkin, Thomas P. Moran, and Beverly L. Harrison. Embodied User Interfaces: Towards invisible user interfaces. In *Proceedings of the Seventh Working Conference on Engineering for Human-Computer Interaction*, pages 1–18. Kluwer, B.V., 1999.

[73] George Fitzmaurice and William Buxton. The Chameleon: Spatially aware palmtop computers. In *CHI '94: Conference companion on Human factors in computing systems*, pages 451–452. ACM Press, 1994.

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

[75] George W. Fitzmaurice. Graspable User Interfaces. Ph.D. dissertation, University of Toronto, 1996.

[76] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[77] George W. Fitzmaurice, Shumin Zhai, and Mark H. Chignell. Virtual reality for palmtop computers. *ACM Trans. Inf. Syst.*, 11(3):197–218, 1993.

[78] Christian Floerkemeier. EPC-Technologie – vom Auto ID Center zu EPCglobal. In Elgar Fleisch and Friedemann Mattern, editors, *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*. Springer-Verlag, 2005.

[79] Christian Floerkemeier, Dipan Anarkat, Ted Osinski, and Mark Harrison. PML Core Specification Version 1.0. Auto-ID Center Recommendation (www.autoidcenter.org), September 2003.

[80] Hans-W. Gellersen, Michael Beigl, and Albrecht Schmidt. Environment-mediated mobile computing. In *SAC '99: Proceedings of the 1999 ACM symposium on Applied computing*, pages 416–418, New York, NY, USA, 1999. ACM Press.

[81] Anatole V. Gershman, Joseph F. McCarthy, and Andrew E. Fano. Situated Computing: Bridging the Gap between Intention and Action. In *ISWC '99: Proceedings of the 3rd IEEE International Symposium on Wearable Computers*, page 3, Washington, DC, USA, 1999. IEEE Computer Society.

[82] Ivan A. Getting. The Global Positioning System. *IEEE Spectrum*, 30(12):36–38,43–47, December 1993.

[83] Beat Gfeller. Recognition of 2-Dimensional Visual Codes with the Nokia 7650. Summer internship project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2003.

[84] Kathleen Gomoll and Anne Nicol. User Observation: Guidelines for Apple Developers, January 1990.

[85] Manuel Graber. Portierung eines Visual Code Systems für Mobiltelefone von Symbian nach Java. Semester project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2004.

[86] Saul Greenberg and Michael Boyle. Moving Between Personal Devices and Public Displays. In *Workshop on Handheld CSCW, held at ACM Conference on Computer Supported Cooperative Work*, November 1998.

[87] Saul Greenberg and Michael Rounding. The notification collage: posting information to public and personal displays. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 514–521, New York, NY, USA, 2001. ACM Press.

[88] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. An Ontology-based Context Model in Intelligent Environments. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation*, pages 270–275, January 2004.

[89] Martin Hachet, Joachim Pouderoux, and Pascal Guitton. A camera-based interface for interaction with mobile handheld computers. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 65–72, New York, NY, USA, 2005. ACM Press.

[90] 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 Press.

[91] Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24. ACM Press/Addison-Wesley Publishing Co., 1998.

[92] Beverly L. Harrison, Gordon Kurtenbach, and Kim J. Vicente. An experimental evaluation of transparent user interface tools and information content. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 81–90. ACM Press, 1995.

[93] Beverly L. Harrison and Kim J. Vicente. An experimental evaluation of transparent menu usage. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 391–398. ACM Press, 1996.

[94] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 59–68, New York, NY, USA, 1999. ACM Press.

[95] Paul S. Heckbert. Fundamentals of Texture Mapping and Image Warping. Master's thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1989.

[96] Jeffrey Heer and Peter Khooshabeh. Seeing the Invisible. In *Workshop on Invisible & Transparent Interfaces in conjunction with Advanced Visual Interfaces (AVI)*, May 2004.

[97] Jeffrey Hightower and Gaetano Boriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[98] Ken Hinckley, Jeff Pierce, Eric Horvitz, and Mike Sinclair. Foreground and background interaction with sensor-enhanced mobile devices. *ACM Trans. Comput.-Hum. Interact.*, 12(1):31–52, 2005.

[99] Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 91–100. ACM Press, 2000.

[100] Ken Hinckley, Mike Sinclair, Erik Hanson, Richard Szeliski, and Matt Conway. The VideoMouse: A camera-based multi-degree-of-freedom input device. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 103–112. ACM Press, 1999.

[101] Masahito Hirakawa and Priyantha Hewagamage. Situated Computing: A Paradigm for the Mobile User-Interaction with Multimedia Sources. *Annals of Software Engineering*, 12:213–239, 2001.

[102] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Proc. Ubicomp 2001*, number 2201 in LNCS, pages 116–122, Springer-Verlag, 2001.

[103] Lars Erik Holmquist, Albrecht Schmidt, and Brygg Ullmer. Tangible interfaces in perspective: Guest editors' introduction. *Personal Ubiquitous Comput.*, 8(5):291–293, 2004.

[104] Lars Erik Holmquist and Tobias Skog. Informative art: information visualization in everyday environments. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 229–235, New York, NY, USA, 2003. ACM Press.

[105] Don Hopkins. The Design and Implementation of PieMenus. *Dr. Dobb's Journal*, 16(12):16–26, 1991.

[106] 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 Press.

[107] International Organization for Standardization. Ergonomic requirements for office work with visual display terminals (VDTs) – Requirements for non-keyboard input devices. ISO/IEC 9241-9, 2000.

[108] International Organization for Standardization. Information technology – Automatic identification and data capture techniques – Bar code symbology – QR Code. ISO/IEC 18004, 2000.

[109] International Organization for Standardization. Information technology – Internationa symbology specification – MaxiCode. ISO/IEC 16023, 2000.

[110] International Organization for Standardization. Information technology – International symbology specification – Data Matrix. ISO/IEC 16022, 2000.

[111] International Organization for Standardization. Information Technology – Automatic Identification and Data Capture Techniques – Bar Code Symbology Specifications – PDF417. ISO/IEC 15438, 2001.

[112] International Organization for Standardization. Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1). ISO/IEC 18092, 2004.

[113] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM Press.

[114] Shahram Izadi, Harry Brignull, Tom Rodden, Yvonne Rogers, and Mia Underwood. Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 159–168, New York, NY, USA, 2003. ACM Press.

[115] Patrick Jayet. Assoziierung mobiler Geräte mit Visual Codes. Semester project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2005.

[116] Brad Johanson, Armando Fox, and Terry Winograd. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. *IEEE Pervasive Computing*, 1(2):67–74, April 2002.

[117] Nikolaos Kaintantzis. ETHOC – Every Thing Has Online Content. Diploma thesis, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2002.

[118] Hirokazu Kato, Mark Billinghurst, Suzanne Weghorst, and Tom Furness. A Mixed Reality 3D Conferencing Application. Technical Report R-99-1 Seattle: Human Interface Technology Laboratory, University of Washington, 1999.

[119] Tim Kindberg. Implementing physical hyperlinks using ubiquitous identifier resolution. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 191–199, New York, NY, USA, 2002. ACM Press.

[120] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic. People, places, things: Web presence for the real world. In *Third IEEE Workshop on Publication Mobile Computing Systems and Applications*, pages 19–28, December 2000.

[121] Tim Kindberg, Ella Tallyn, Rakhi Rajani, and Mirjana Spasojevic. Active photos. In *DIS '04: Proceedings of the 2004 conference on Designing interactive systems*, pages 337–340. ACM Press, 2004.

[122] Naohiko Kohtake, Jun Rekimoto, and Yuichiro Anzai. InfoStick: An Interaction Device for Inter-Appliance Computing. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 246–258, London, UK, 1999. Springer-Verlag.

[123] 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.

[124] Christian Kray, Gerd Kortuem, and Antonio Krüger. Adaptive navigation support with public displays. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 326–328, New York, NY, USA, 2005. ACM Press.

[125] Michael Kruppa. The better remote control – Multiuser interaction with public displays. In *Workshop on Multi-User and Ubiquitous User Interfaces (MU3I) at IUI'04*, January 2004.

[126] Michael Kruppa and Antonio Krüger. Concepts for a combined use of Personal Digital Assistants and large remote displays. In *Proceedings of SimVis 2003*. SCS Verlag, 2003.

[127] Matthias Lampe, Christian Floerkemeier, and Stephan Haller. Einführung in die RFID-Technologie. In Elgar Fleisch and Friedemann Mattern, editors, *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*. Springer-Verlag, 2005.

[128] Marc Langheinrich. *Personal Privacy in Ubiquitous Computing – Tools and System Support*. PhD thesis, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, May 2005.

[129] Marc Langheinrich, Vlad Coroama, Jürgen Bohn, and Michael Rohs. As we may live – Real-world implications of ubiquitous computing. Technical Report, ETH Zurich, April 2002.

[130] Erich Laube. Design and Implementation of a Computer-Augmented Memory Game. Semester project, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2004.

[131] Peter Ljungstrand and Lars Erik Holmquist. WebStickers: Using physical objects as WWW bookmarks. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 332–333. ACM Press, 1999.

[132] Peter Ljungstrand, Johan Redström, and Lars Erik Holmquist. WebStickers: using physical tokens to access, manage and share bookmarks to the Web. In *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*, pages 23–31, New York, NY, USA, 2000. ACM Press.

[133] Anil Madhavapeddy, David Scott, Richard Sharp, and Eben Upton. Using Camera-Phones to Enhance Human-Computer Interaction. In *Sixth International Conference on Ubiquitous Computing (Adjunct Proceedings: Demos)*, September 2004.

[134] Carsten Magerkurth and Peter Tandler. Interactive Walls and Handheld Devices – Applications for a Smart Environment. In *Workshop on Collaboration with Interactive Walls and Tables, UBICOMP*, September 2002. Available at: http://www.ipsi.fraunhofer.de/ambiente/collabtablewallws/.

[135] Jennifer Mankoff and Anind K. Dey. From Conception to Design: A practical guide to designing ambient displays. In Kenton O'Hara, Mark Perry, Elizabeth Churchill, and D. Russell, editors, *Public and Situated Displays – Social and Interactional Aspects of Shared Display Technologies*, The Kluwer International Series on Computer Supported Cooperative Work, Vol. 2. Kluwer Academic Publishers, 2003.

[136] Friedemann Mattern. The Vision and Technical Foundations of Ubiquitous Computing. *Upgrade*, 2(5):2–6, October 2001.

[137] Friedemann Mattern. Vom Verschwinden des Computers – Die Vision des Ubiquitous Computing. In Friedemann Mattern, editor, *Total vernetzt*, pages 1–41. Springer-Verlag, April 2003.

[138] Friedemann Mattern. Die technische Basis für das Internet der Dinge. In Elgar Fleisch and Friedemann Mattern, editors, *Das Internet der Dinge – Ubiquitous Computing und RFID in der Praxis*, pages 39–66. Springer-Verlag, 2005.

[139] Friedemann Mattern. Ubiquitous Computing: Eine Einführung mit Anmerkungen zu den sozialen und rechtlichen Folgen. In Jürgen Taeger and Andreas Wiebe, editors, *Mobilität, Telematik, Recht (DGRI-Jahrestagung 2004)*. Verlag Dr. Otto Schmidt, 2005.

[140] Friedemann Mattern. Allgegenwärtige und verschwindende Computer. *Praxis der Informationsverarbeitung und Kommunikation (PIK)*, 28(1), February 2005.

[141] Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In *IEEE Symposium on Security and Privacy*, pages 110–124, Washington, DC, USA, May 2005. IEEE Computer Society.

[142] Jean-Daniel Merkli. Smart Product Packaging. Diploma thesis, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2005.

[143] Keith Mitchell, Nicholas Race, and Michael Clarke. CANVIS: Context-aware network visualisation using smartphones. In *Proceedings of 7th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*. ACM Press, September 2005.

[144] 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 Press.

[145] Ben Miyares. RFID: The Packaging Imperatives – What needs to happen to make RFID integral to packaging. Packaging Machinery Manufacturers Institute, RFID Journal Live, www.packexpo.com, www.mmi.org, April 19 2005.

[146] Mathias Moehring, Christian Lessig, and Oliver Bimber. Optical Tracking and Video See-Through AR on Consumer Cell Phones. In *Workshop on Virtual and Augmented Reality of the GI-Fachgruppe AR/VR*, pages 193–204, 2004.

[147] Mathias Moehring, Christian Lessig, and Oliver Bimber. Video See-Through AR on Consumer Cell Phones. In *IEEE/ACM International Symposium on Augmented and Mixed Reality (ISMAR'04)*, pages 252–253, 2004.

[148] Gordon E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114–117, April 1965. See also: www.intel.com/research/silicon/mooreslaw.htm.

[149] Brad A. Myers, Herb Stiel, and Robert Gargiulo. Collaboration using multiple PDAs connected to a PC. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 285–294. ACM Press, 1998.

[150] Erica Newcomb, Toni Pashley, and John Stasko. Mobile computing in the retail arena. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 337–344, New York, NY, USA, 2003. ACM Press.

[151] Nokia Corporation. Nokia Mobile RFID Kit and Nokia NFC (Near Field Communication). www.nokia.com/rfid and www.nokia.com/nfc.

[152] Donald A. Norman. *The Design of Everyday Things*. Basic Books, New York, 1988.

[153] Kenton O'Hara, Mark Perry, Elizabeth Churchill, and D. Russell, editors. *Public and Situated Displays – Social and Interactional Aspects of Shared Display Technologies*. The Kluwer International Series on Computer Supported Cooperative Work, Vol. 2. Kluwer Academic Publishers, 2003.

[154] Tapan S. Parikh. Using Mobile Phones for Secure, Distributed Document Processing in the Developing World. *IEEE Pervasive Computing*, 4(2):74–81, January 2005.

[155] Jason Pascoe. The stick-e note architecture: extending the interface beyond the user. In *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 261–264, New York, NY, USA, 1997. ACM Press.

[156] Thomas Pederson. Human hands as a link between physical and virtual. In *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*, pages 153–154, New York, NY, USA, 2000. ACM Press.

[157] Claudio Pinhanez. The Everywhere Displays Projector: A Device to Create Ubiquitous Graphical Interfaces. In G.D. Abowd, B. Brumitt, and S. Shafer, editors, *Ubicomp 2001: Ubiquitous Computing*, pages 315–331. LNCS 2201, Springer, 2001.

[158] Moe Rahnema. Overview of the GSM System and Protocol Architecture. *IEEE Communications Magazine*, 31(4):92–100, April 1993.

[159] Jef Raskin. *The humane interface: New directions for designing interactive systems*. ACM Press/Addison-Wesley Publishing Co., March 2000.

[160] Irving S. Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 1960.

[161] J. Rekimoto, Y. Ayatsuka, and K. Hayashi. Augment-able Reality: Situated Communication through Physical and Digital Spaces. In *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, pages 68–75. IEEE Computer Society, 1998.

[162] Jun Rekimoto. The Magnifying Glass Approach to Augmented Reality Systems. In *International Conference on Artificial Reality and Tele-Existence '95 / Conference on Virtual Reality Software and Technology (ICAT/VRST '95)*, 1995.

[163] Jun Rekimoto. Tilting operations for small screen interfaces. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 167–168. ACM Press, 1996.

[164] Jun Rekimoto. Navicam: A magnifying glass approach to augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):399–412, August 1997.

[165] Jun Rekimoto and Yuji Ayatsuka. CyberCode: Designing Augmented Reality Environments with Visual Tags. In *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*, pages 1–10. ACM Press, 2000.

[166] 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 and software technology*, pages 29–36, New York, NY, USA, 1995. ACM Press.

[167] Jun Rekimoto and Eduardo Sciammarella. ToolStone: Effective use of the physical manipulation vocabularies of input devices. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 109–117. ACM Press, 2000.

[168] Meredith Ringel, Henry Berg, Yuhui Jin, and Terry Winograd. BareHands: Implement-free interaction with a wall-mounted display. In *CHI '01 extended abstracts on Human factors in computing systems*, pages 367–368. ACM Press, 2001.

[169] Michael Rohs. Real-World Interaction with Camera-Phones. In *2nd International Symposium on Ubiquitous Computing Systems (UCS)*, pages 39–48, Tokyo, Japan, November 2004.

[170] Michael Rohs. Real-World Interaction with Camera Phones. In Hitomi Murakami, Hideyuki Nakashima, Hideyuki Tokuda, and Michiaki Yasumura, editors, *Second International Symposium on Ubiquitous Computing Systems (UCS 2004), Revised Selected Papers*, pages 74–89, Tokyo, Japan, July 2005. LNCS 3598, Springer.

[171] Michael Rohs. Visual Code Widgets for Marker-Based Interaction. In *IWSAWC'05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems – Workshops (ICDCS 2005 Workshops)*, pages 506–513, Columbus, Ohio, USA, June 2005.

[172] Michael Rohs. Marker-Based Interaction Techniques for Camera-Phones. In *IUI 2005 Workshop on Multi-User and Ubiquitous User Interfaces (MU3I)*, January 2005.

[173] Michael Rohs and Jürgen Bohn. Entry Points into a Smart Campus Environment – Overview of the ETHOC System. In *IWSAWC '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 260–266. IEEE Computer Society, 2003.

[174] Michael Rohs and Beat Gfeller. Visual Code Recognition for Camera-Equipped Mobile Phones. Pervasive 2004 Demonstrations D06. Demonstration description available at: www.pervasive2004.org/program_demonstrations.php, 2004.

[175] Michael Rohs and Beat Gfeller. Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects. In Alois Ferscha, Horst Hoertner, and Gabriele Kotsis, editors, *Advances in Pervasive Computing*, pages 265–271, Vienna, Austria, April 2004. Austrian Computer Society (OCG).

[176] Michael Rohs and Christof Roduner. Camera Phones with Pen Input as Annotation Devices. In *Pervasive 2005 Workshop on Pervasive Mobile Interaction Devices (PERMID)*, Munich, Germany, May 2005.

[177] Michael Rohs, Jennifer G. Sheridan, and Rafael Ballagas. Direct Manipulation Techniques for Large Displays using Camera Phones. Demonstration at 2nd International Symposium on Ubiquitous Computing Systems (UCS), November 2004.

[178] Michael Rohs and Philipp Zweifel. A Conceptual Framework for Camera Phone-Based Interaction Techniques. In Hans W. Gellersen, Roy Want, and Albrecht Schmidt, editors, *Pervasive Computing: Third International Conference, PERVASIVE 2005*, pages 171–189, Munich, Germany, May 2005. LNCS 3468, Springer.

[179] D. A. Rosenbaum, J. D. Slotta, J. Vaughan, and R. Plamondon. Optimal movement selection. *Psychological Science*, 2:86–91, 1991.

[180] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image Retrieval: Current Techniques, Promising Directions, and Open Issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, March 1999.

[181] Enrico Rukzio, Albrecht Schmidt, and Heinrich Hussmann. An Analysis of the Usage of Mobile Phones for Personalized Interactions with Ubiquitous Public Displays. In *Workshop for Ubiquitous Displays, UBICOMP 2004*, September 2004.

[182] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the development of context-enabled applications. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 434–441, New York, NY, USA, 1999. ACM Press.

[183] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4):10–17, August 2001.

[184] Albrecht Schmidt. Implicit Human Computer Interaction Through Context. *Personal Technologies*, 4(2):191–199, June 2000.

[185] Albrecht Schmidt. *Ubiquitous Computing – Computing in Context*. PhD thesis, Computing Department, Lancaster University, UK, November 2002.

[186] David Scott, Richard Sharp, Anil Madhavapeddy, and Eben Upton. Using visual tags to bypass Bluetooth device discovery. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):41–53, 2005.

[187] Ehud Sharlin, Benjamin Watson, Yoshifumi Kitamura, Fumio Kishino, and Yuichi Itoh. On tangible user interfaces, humans and spatiality. *Personal Ubiquitous Comput.*, 8(5):338–346, 2004.

[188] Jane M. Shaw and Paul F. Seidler. Organic electronics: Introduction. *IBM Journal of Research and Development*, 45(1):3–10, January 2001.

[189] Ben Shneiderman. Direct manipulation: A step beyond programming languages. In *Proceedings of the joint conference on Easier and more productive use of computer systems. (Part - II)*, page 143, New York, NY, USA, 1981. ACM Press.

[190] Frank Siegemund. *Cooperating Smart Everyday Objects – Exploiting Heterogeneity and Pervasiveness in Smart Environments*. PhD thesis, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, December 2004.

[191] Frank Siegemund and Michael Rohs. Rendezvous Layer Protocols for Bluetooth-Enabled Smart Devices. *Journal for Personal and Ubiquitous Computing (PUC)*, 7(2):91–101, October 2003.

[192] Itiro Siio, Toshiyuki Masui, and Kentaro Fukuchi. Real-world Interaction using the FieldMouse. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 113–119. ACM Press, 1999.

[193] Marc Smith, Duncan Davenport, Howard Hwa, and Lik Mui. The Annotated Planet: A mobile platform for object and location annotation. In *1st Int. Workshop on Ubiquitous Systems for Supporting Social Interaction and Face-to-Face Communication in Public Spaces at UbiComp 2003*, October 2003.

[194] Marc Smith Smith, Duncan Davenport, Howard Hwa, and Tammara Combs Turner. Object auras: A mobile retail and product annotation system. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 240–241. ACM Press, 2004.

[195] Norbert Streitz and Paddy Nixon. Introduction. *Commun. ACM*, 48(3):32–35, 2005.

[196] 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 Press.

[197] Sun Microsystems. Java 3D API Specification, version 1.3. Available at: java.sun.com/products/java-media/3D, 2002.

[198] I. E. Sutherland. A head-mounted three-dimensional display. In *Fall Joint Computer Conf., Am. Federation of Information Processing Soc. (AFIPS) Conf. Proc. 33*, pages 757–764. Thompson Books, Washington, D.C., 1968.

[199] Joo Geok Tan, Daqing Zhang, Xiaohang Wang, and Heng Seng Cheng. Enhancing Semantic Spaces with Event-Driven Context Interpretation. In Hans W. Gellersen, Roy Want, and Albrecht Schmidt, editors, *Pervasive Computing: Third International Conference, PERVASIVE 2005*, pages 80–97, Munich, Germany, May 2005. LNCS 3468, Springer.

[200] Mark A. Tapia and Gordon Kurtenbach. Some design refinements and principles on the appearance and behavior of marking menus. In *Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 189–195. ACM Press, 1995.

[201] Johan Thoresson. PhotoPhone entertainment. In *CHI '03 extended abstracts on Human factors in computing systems*, pages 896–897. ACM Press, 2003.

[202] Peter Tolmie, James Pycock, Tim Diggins, Allan MacLean, and Alain Karsenty. Unremarkable computing. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 399–406, New York, NY, USA, 2002. ACM Press.

[203] Brygg Ullmer and Hiroshi Ishii. Emerging Frameworks for Tangible User Interfaces. In John M. Carroll, editor, *In Human-Computer Interaction in the New Millennium*, pages 579–601. Addison-Wesley, August 2001.

[204] Pasi Välkkynen, Ilkka Korhonen, Johan Plomp, Timo Tuomisto, Luc Cluitmans, Heikki Ailisto, and Heikki Seppä. A user interaction paradigm for physical browsing and near-object control based on tags. In *Physical Interaction (PI03) – Workshop on Real World User Interfaces in conjunction with Mobile HCI 2003*, September 2003.

[205] Remco C. Veltkamp and Michiel Hagedoorn. State of the Art in Shape Matching. In *Principles of Visual Information Retrieval*, pages 87–119. Springer-Verlag, London, UK, 2001.

[206] 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 Press.

[207] Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg. Towards Massively Multi-user Augmented Reality on Handheld Devices. In Hans W. Gellersen, Roy Want, and Albrecht Schmidt, editors, *Pervasive Computing: Third International Conference, PERVASIVE 2005*, pages 208–219, Munich, Germany, May 2005. LNCS 3468, Springer.

[208] Daniel Wagner and Dieter Schmalstieg. First Steps Towards Handheld Augmented Reality. In *Seventh IEEE International Symposium on Wearable Computers*, pages 127–135, Washington, DC, USA, October 2003. IEEE Computer Society.

[209] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology Based Context Modeling and Reasoning using OWL. In *Workshop on Context Modeling and Reasoning(CoMoRea), in conjunction with the Second IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 18–22, March 2004.

[210] Roy Want. RFID: A Key to Automating Everything. *Scientific American*, 290(1):46–55, January 2004.

[211] Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging Physical and Virtual Worlds with Electronic Tags. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 370–377. ACM Press, 1999.

[212] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

[213] Roy Want and Trevor Pering. System challenges for ubiquitous & pervasive computing. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 9–14, New York, NY, USA, 2005. ACM Press.

[214] Andy Ward, Alan Jones, and Andy Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[215] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, September 1991. Reprinted in IEEE Pervasive Computing, Jan.-Mar. 2002, pp. 19–25.

[216] Mark Weiser. Hot Topics – Ubiquitous Computing. *IEEE Computer*, 26(10):71–72, October 1993.

[217] Mark Weiser. The future of ubiquitous computing on campus. *Commun. ACM*, 41(1):41–42, 1998.

[218] Mark Weiser and John Seely Brown. The Coming Age of Calm Technology. Xerox PARC. Available as: www.ubiq.com/hypertext/weiser/acmfuture2endnote.htm, October 1996.

[219] Mark Weiser, Rich Gold, and John Seely Brown. The Origins of Ubiquitous Computing Research at PARC in the late 1980s. *IBM Systems Journal*, 38(4):693–696, 1999.

[220] Pierre Wellner. Interacting with paper on the DigitalDesk. *Commun. ACM*, 36(7):87–96, 1993.

[221] Pierre D. Wellner. Adaptive thresholding for the DigitalDesk. Technical Report EPC-93-110, Rank Xerox Research Centre, Cambridge, UK, 1993.

[222] Daniel Wigdor and Ravin Balakrishnan. TiltText: Using tilt for text input to mobile phones. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 81–90. ACM Press, 2003.

[223] 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 Press.

[224] Tom Yeh, Kristen Grauman, Konrad Tollmar, and Trevor Darrell. A picture is worth a thousand keywords: image-based object search on a mobile platform. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 2025–2028, New York, NY, USA, 2005. ACM Press.

[225] Tom Yeh, Konrad Tollmar, and Trevor Darrell. IDeixis: image-based Deixis for finding location-based information. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 781–782, New York, NY, USA, 2004. ACM Press.

[226] Shumin Zhai. User performance in relation to 3D input device design. *SIGGRAPH Comput. Graph.*, 32(4):50–54, 1998.

[227] Shumin Zhai, Paul Milgram, and William Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 308–315, New York, NY, USA, 1996. ACM Press.

[228] Philipp Zweifel. Object Selection and Form Data Collection with Camera-Equipped Mobile Phones. Diploma thesis, Institute for Pervasive Computing, Department of Computer Science, Swiss Federal Institute of Technology (ETH Zurich), Switzerland, 2004.

Appendix D

Curriculum Vitae

	Personal Data
	Michael Rohs
	Diplom-Informatiker M.S. Computer Science (USA)
	born January 13, 1974 citizen of Germany
	School
08/1980 – 07/1984	Gemeinschaftsgrundschule in Schönenberg, Germany
08/1984 – 05/1993	Hollenberggymnasium in Waldbröl, Germany Abitur diploma
08/1993 – 10/1994	Zivildienst in Much, Germany
	University
10/1994 – 07/1997	Darmstadt University of Technology, Germany undergraduate studies in computer science completed (Vordiplom) and first year of graduate studies
08/1997 – 05/1998	University of Colorado at Boulder, USA Master of Science, Computer Science grade point average (GPA): 4.0
10/1998 – 04/2000	Darmstadt University of Technology, Germany Diplom-Informatiker minor subjects: psychology, software ergonomics grade: “Mit Auszeichnung bestanden”
05/2000 – present	Swiss Federal Institute of Technology, Switzerland Ph.D. candidate in computer science

Work Experience

07/1993 – 08/1993	Grundpraktikum Deutsche Telekom in Bonn, Germany
10/1996 – 07/1997	Darmstadt University of Technology, Germany tutorship for undergraduate courses
07/1998 – 05/1999	Fraunhofer Institute for Applied Information Technology (at the time called GMD-FIT) in Sankt Augustin, Germany development work of a computer-supported cooperative work system, part-time job
05/2000 – present	Swiss Federal Institute of Technology, Switzerland Research Assistant at the Institute for Pervasive Computing

Additional Information

Languages	German (native language), English, French, Latin
Professional memberships	GI, ACM, IEEE