Steerable Interfaces for Interactive Environments

Stanislaw Borkowski

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

N ° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Spécialité : Imagerie, Vision et Robotique
Ecole Doctorale : Mathématiques, Sciences et Technologie de l'Information, Informatique

présentée et soutenue publiquement
par

Stanisław Borkowski

le 26 juin 2006

Steerable Interfaces for Interactive Environments

Directeur de thèse : M. James L. Crowley

JURY

Mme Joëlle Coutaz,	Président
M. Andreas Butz,	Rapporteur
M. Alex Pentland,	Rapporteur
M. Pierre Wellner,	Examinateur
M. James L. Crowley,	Directeur de thèse

Thèse préparée dans le laboratoire GRAVIR – IMAG au sein du projet PRIMA
INRIA Rhône-Alpes, 655 av. de l'Europe, 38334 Saint Ismier, France.

INTERFACES PILOTABLES POUR L'INTERACTION HOMME MACHINE DANS DES ENVIRONNEMENTS INTERACTIFS

Stanisław Borkowski
Institut National Polytechnique de Grenoble, 26 June 2006

La mobilité est une caractéristique fondamentale pour l'informatique moderne. Cependant, la mobilité n'est toujours considéré que comme propriété propre aux dispositifs d'interaction portée. Cette thèse étudie une nouvelles classe d'interfaces mobiles, capables d'être déplacées dans l'environnement à l'aide de systémes informatiques : les interfaces dirigeables. Nous présentons un prototype d'interface dirigeable et investiguons les applications possibles de ce type d'interfaces, et évaluons également l'impact de la dirigeabilité d'une interface dans un contexte de travail collaboratif co-présent.

Un couple composé d'un vidéo projecteur et une camera motorisée nous permet d'afficher des images numériques sur presque toutes les surfaces de notre salle d'expérimentation. Pour compenser les distorsions projectives nous proposons deux méthodes de rectification dynamique d'images : *(a)* basée sur la détection des bords de la surface d'affichage, et *(b)* basée sur un modéle de projecteurs. Nous utilisons l'algorithme de détection de bords pour créer une surface portable d'affichage à partir d'une piéce de carton.

Pour rendre les images projetées interactive, nous utilisons la vision par ordinateur. Le champ de vue de la caméra couplé avec le vidéo projecteur pilotable contient l'image projetée. Néanmoins, projection et la mobilité de la projection rend difficile le traitement d'images pour la détection d'interactions. Nous présentons une implémentation de widgets de base comme des boutons ou défileurs adaptés aux l'interaction avec des interfaces projetées.

Le couple projecteur-caméra et la surface portable d'affichage nous permet d'expérimenter des interfaces mobiles projetées. Nous proposons un nombre de techniques d'interaction pour contrôler la position d'une interface. Les techniques sont analysées en utilisant le modéle de couplage de ressources d'interactions. Cette thèse se conclut par une étude d'utilisation de notre systéme, qui a pour but de comparer les interfaces fixe, portable et dirigeable dans le cadre d'un travaille collaboratif. Les résultats de notre étude démontre l'utilité d'interfaces pilotables et devrait motiver la recherche sur se type d'interaction.

STEERABLE INTERFACES FOR INTERACTIVE ENVIRONMENTS

Stanisław Borkowski, Ph.D. dissertation
Institut National Polytechnique de Grenoble, 26 June 2006

Mobility is a fundamental feature of modern information technology. However, currently mobility is usually related to either data or to portable computing devices. Alternatively, mobility may refer to the perceptible boundary of the computer system, i.e. to the interface.

In this thesis, we investigate the use of a new class of interfaces that have the ability to be moved within the environment: steerable interfaces. We address the issue of rendering interfaces steerable, explore their potential applications and evaluate the impact of interface steerability in the context of colocated collaborative work.

A Steerable Camera-Projector pair, built for this study, allows us to display digital images on almost any surface in our laboratory. However, projecting on surfaces at arbitrary angles may distort the image. We propose two methods for dynamic image rectification; *(a)* based on detection of screen boundaries, and *(b)* based on a pinhole projector model and the relative orientation of the projector and the display surface. We use the screen boundaries detection algorithm to create a lightweight Portable Display Surface out of a handheld piece of cardboard.

We use computer vision to render projected images interactive. Indeed, the view of the camera mounted on the computer controlled video projector overlaps with the projection cone. The camera can be used to detect user actions over the displayed images. However, light emitted by the video projector and the mobility of our interfaces result in challenging conditions for conventional computer vision algorithms. We propose a computer vision based implementation of basic interactive widgets that is robust to changes in lighting conditions.

To develop our projection-based interactive system we adopt a Service Oriented Architecture. We propose a set of software services that are necessary to develop applications with steerable interfaces. To ensure the quality of our system, we evaluate these services with respect to user-centered usability criteria.

The Steerable Camera-Projector together with the Portable Display Surface allows us to experiment with projection-based mobile interfaces, both portable and steerable. We propose a number of interaction techniques for controlling the location of a steerable interface, and analyze them using an adaptation of the "coupling" model. To evaluate interface steerability as a new form of interaction with computers, we perform a user study comparing immobile, portable and steerable interfaces in the context of colocated collaborative work. The results of the study show that users enjoy working with both portable and steerable interfaces. These results should motivate further research on steerable interfaces.

Acknowledgements

I would like to thank all people that have contributed to the completion of this thesis. Especially Prof. James L. Crowley who offered me the possibility to work in his group and gave me all the freedom and assistance I needed to develop my work. His original ideas and many suggestions were a great help to me. I am grateful to Prof. Joëlle Coutaz for introducing me to the human-centric part of computer science, and to Prof. Alex Pentland and Prof. Andreas Butz for their interest in my work and for being the reporters of this dissertation.

I would also like to thank Julien Letessier and Jérôme Maisonnasse for their criticism and their input to this thesis. I am grateful to Gyorgy Dorkó, Julien and Dominique Vaufreydaz for explaining me their point of view on computer programming.

I would like to thank Daniela Hall and Olivier Riff for helping me to integrate the PRIMA group, Doms for being a fun officemate, Krystian Mikołajczyk for inviting me to join the Polish community in Grenoble, Nicolas Barralon for inventing the "coupling model", Francois Bérard and Gaetan Rey for showing me how to play baseball, and other fellow researchers from INRIA and CLIPS for making Grenoble a fun and motivating place to work.

My special thanks goes to my friends Gyuri, Carla, Bram, An, Peter, Eric and Audrey with whom Marlen and myself spent many joyful moments in Grenoble, and to Christina, Jérôme and Thomas, who shared with us their mountaineering experience. I am also grateful to my Polish friends Bartek, Michal, Kasia, Głowki and Gosia who have not forgotten about me even though I am so far from home.

Last, but not least, I would like to thank my wife Marlena and my family for their love, support and encouragement.

This research was supported by the European project FAME (IST-2000-28323) and the RNTL/Proact ContAct project.

Table of Contents

First Chapter

Introduction

The way people interact with computers evolves with technical advances of information technology. Flat display screens, optical mice adapted to the shape of human hand, and better application interfaces improve the experience of working with desktop computers. Though these improvements contribute to the ergonomics of computers, they are not determinant for the future of Human-Computer Interaction (HCI). It is rather the proliferation and diversification of information technology that is the driving force for new forms of interaction.

Information technology becomes omnipresent. Desktop computers, personal digital assistants and even wrist-watches provide interfaces to the virtual world. As the various electronic devices start to communicate and become interoperable, new scenarios for using information technology arise. Ubiquitous computing is one example of the emerging concepts.

Ubiquitous Computing called also *Ubicomp*, refers to a trend in computer science following the idea of integrating information technology into the world to offer seamless support in everyday activities. By embedding computation into the environment, ubiquitous computing aims at enabling people to interact more naturally than they currently do. In such interactive environments, users will be free to interact with computers almost everywhere, without being forced to use a mouse or keyboard installed in front of a fixed display screen. People will also have the possibility to move their computer interface at will.

Mobility is one of the key features of Ubicomp, and of modern IT in general. The use of mobile computing alleviates spatial constraints inherent to conventional desktop computers. Palmtops, mobile phones and other portable devices allow users to interact with the virtual world almost anywhere. While mobile computing is usually related to portable computers, mobility can be equally realized through both steerability and portability.

In this study, we define, develop and test a new class of user interfaces, distinguished from conventional computing by their ability to move in space. We anticipate the evolution of IT, and investigate interaction techniques for controlling the emerging

interactive spaces. Our objectives are to evaluate the impact of interface steerability on interaction with computers, and to propose a conceptual framework for reasoning about the design of interaction techniques allowing spatial control of steerable interfaces.

In the following sections, we discuss the problems addressed, our approach and the principal contributions of this thesis.

1.1 Approach and addressed problems

Our approach is to develop a system allowing people to interact with a graphical steerable interface. The system must be interactive, and must allow us to design and test interaction techniques for controlling interface location. We evaluate the proposed interaction techniques, and perform a user-study to compare steerable interfaces with portable and immobile interfaces.

Currently interaction resources are designed to be either fixed (e.g. display screens, keyboards etc.), or portable (e.g. PDAs, mobile phones etc.). Similarly, research on human-computer interaction is also focused on fixed and mobile interfaces. In order to study steerable interfaces, we had to investigate both enabling technology to create such interfaces, and frameworks for modeling the interaction.

Out of available technologies, we chose to use steerable projection combined with computer vision. Together, they allow us to create a steerable interactive display. While video projectors have a number of advantages, such as small form factor and the ability to enhance ordinary objects with digital imagery, they have also several drawbacks. In particular, geometric and radiometric image distortions are inherent to projection systems, when displaying on uneven surfaces. As a result, our projection-based steerable display system has to detect surfaces suitable for projection, and dynamically deform displayed images before projection to compensate distortions.

Computer vision offers the means to render the projected display interactive. Interaction events can be detected based on user gestures, without mechanical transducers such as mice or keyboards. However, steerable projected interfaces provide a very challenging context for vision-based user interface design. Objects observed by the camera are likely to be surrounded by shadows cast in the projection beam. What is more, the appearance of objects is modified by the projected light. Most of the vision-based algorithms suitable to detect user actions work only in very constrained conditions, and thus cannot by used in our system. Therefore, we investigate algorithms for user interaction detection that are robust to the difficult conditions specific to this problem.

People can move steerable interfaces in the environment by issuing command to the computer system. The system is bound to the physical space in which the interface is moved. The coupling of IT with its environment is unusual in conventional computing. As a result, there is a need for new interaction techniques that allow efficient control of interface location, and for new formalisms to comprehensibly describe and evaluate

such interactions. We address both these issues in this thesis.

With respect to the development of the components of our interactive system, we adopt a user-centric and developer-centric approach. The user-centric approach means that we consider usability properties, such as reliability or latency, as central to the design, and evaluate them for each component. Developer-centric approach implies that our software design should minimize difficulties in deploying our interactive system; i.e. it should hide domain specific issues and expose only high level abstracted structure of the interactive system.

1.2 Summary of contributions

The principal contribution of this study is the development of our interactive testbed enabled with steerable projection. Our prototype has been built from scratch, and involved both the choice of appropriate technology, and the integration of the technology with the computer system. Therefore the contributions of this study relate to both technical and conceptual issues.

The contributions related to technical issues are related to the development of the projection-based interactive display. We present a model of the projector-screen configuration that can be used for dynamic image rectification. We also present a simple model of the environment for representing surfaces that are suitable for interaction. The model is generated automatically, by taking advantage of several non-calibrated cameras present in our instrumented environment.

We introduce a new display technology combining steerable projection and computer vision, to create a projection based portable display made of a lightweight cardboard.

To render the projection interactive, we propose a computer vision based implementation of standard GUI widgets (buttons and sliders). Our widgets are developed within a developer-centric, service-oriented framework, designed to allow developers unfamiliar with vision to use computer vision as an interaction modality.

We implement three interaction techniques for controlling the location of a projected interface using simple actuators: fingers, a laser pointer, and a portable display surface. These techniques serve as a showcase for the analysis of steerable interfaces.

Our analysis draws on the IFIP usability properties, and the framework for coupling interaction resources introduced by Barralon et al. in [60]. The application of the coupling framework to the proposed interaction techniques allowed us to make the work of Barralon et al sounder.

Finally, we perform a user-study to compare the usability of fixed, portable and steerable interfaces in the context of collaborative work. The results of the study show that colocated collaborative work is a potential field where steerable interfaces can be applied to improve the interaction.

1.3 Structure of the Thesis

In the second chapter, we present in more details the context of this study. We refer our work to the concepts of Ubiquitous Computing, and Virtual and Augmented Reality. We identify steerability as a primary characteristic of a new class of interfaces, and propose a set of properties which form a classification space. We refer our definition of the steerable interface to the work of Pingali et al. [64], who proposed a more restrictive definition. A brief review of state-of-the-art systems in which space is rendered interactive through interface steerability, serves to show how the identified properties can be applied to analyze steerable interfaces. We refer to the presented systems in the following chapters to illustrate different aspects of the design and analysis of steerable interfaces.

In chapter 3 we present our approach to the design and development of interactive systems. For our software components, we adopt a Service Oriented Architecture (SOA). We propose a number of user-centred requirements that services must meet to be successful in interactive systems. To position our work with respect to established software evaluation standards, we refer to the ISO 25000 standard.

Chapter 4 describes tools and techniques forming the basis of our steerable interface implementation. We compare the two alternative approaches to create steerable projection; the mirror-based beam control, and the motorized assembly solution. We describe our Steerable Camera Projector (SCP) pair, and present both the advantages of projected graphics and issues related to computer vision based interaction with projected images. We also introduce a projection-based Portable Display Surface (PDS). We refer the characteristics of both the SCP and PDS to usability properties defined in chapter 3.

In chapter 5, we present the two facets of interactive systems: the input subsystem, i.e. components that allow users to express commands to the computer, and the output subsystem responsible for providing the feedback to the user. We identify input modalities available for projected interfaces and discus how they can be exploited for mobile interfaces. We describe in more detail our implementation of vision-based interactive widgets. This chapter also addresses issues related to projecting digital images on arbitrary planar surfaces. We present a model of the geometrical relation between the display surface and the video projector. The model allows us to dynamically rectify images even while they are moved across a planar surface. We describe our approach to environment modeling, and to model data acquisition. Finally, we present the minimal set of services necessary for building interactive applications deployed with the SCP as the main input/output device.

In chapter 6, we focus on interaction techniques allowing users to control the location of an interface, and on the evaluation of interface mobility as a means to enhance interaction with computers. We also identify the application fields in which steerable interface can enhance user experience with information technology. To illustrate the potential of mobile interfaces in the field of Computer Supported Collaborative Work

(CSCW), we present a prototype application for authoring presentations. Finally, we present the results of a user study evaluating different types of mobile interfaces in the context of collaborative work.

Second Chapter

Interface mobility in interactive spaces

Mobility is one of the key features of modern information technology. Laptop and palmtop computers, mobile phones and personal digital assistants allow users to interact with the virtual world almost anywhere. Although, mobile computing is usually related to portable interaction devices, it can also be realized through interface steerability.

Currently, in both home and office environments, interaction with information technology is mostly limited to conventional workstations designed for use in immobile setups. However, advances in information technology increasingly render computing ubiquitous and environment interactive. In interactive spaces both users and interaction resources can be mobile. In consequence, steerability of interfaces becomes a fundamental feature for interactive spaces.

In this chapter, we define the concept of a steerable interface, and propose a set of properties that form a classification space. We relate our definition of a steerable interface to the work of Pingali et al. [64], who proposed a more restrictive definition of steerable interfaces. A brief review of state-of-the-art systems, in which space is rendered interactive through interface steerability, serves as a showcase for the application of the identified properties to analyze steerable interfaces. We also refer to the systems presented in the following chapters in order to illustrate different aspects of the design and analysis of steerable interfaces.

2.1 Interactive spaces

Our environment has become populated with panoply of electronic devices. As these computing entities start to be interoperable and start to form a unified interactive system, the space becomes interactive. We define *an interactive space as a limited extent in one, two, or three dimensions where interaction is possible at any time.*

This definition can be applied to a variety of interactive systems depending on the interpretation of terms "any time", "dimensions" and "interaction is possible".

In this section, we express our understanding of interactive space. We refer the notion of interactive space with respect to different research currents, such as virtual and augmented reality and ubiquitous computing. Together they determine our investigation space and provide a framework for analysis of interactive spaces.

2.1.1 Virtual vs. Augmented Reality

Virtual reality (VR) is a trend in computer science, driven by the idea of providing the user with an illusion of being a part of an environment generated by the computer. The user is immersed in the virtual world.

In [30] Fischer et al. presented a seminal VR system. It rendered the virtual world in a head-mounted display worn by the user. The display was equipped with a 3D position and orientation sensor, allowing capturing user's movements. Additionally, the user could also wear sensors on hands or other body parts. Each sensed limb was then integrated into the simulated reality. This setup was followed by others, and the combination of a data-glove for sensing hand gestures and a head-mounted display became an emblem for VR systems.

The user immersion is very demanding to both data acquisition subsystem and to the scene generation and rendering subsystem; each movement of the user results in a change of the presentation of the generated world. Therefore, despite two decades of research in the field, VR applications remain limited mostly to experimental prototypes.

Augmented reality (AR) aims to combine the real and the virtual in order to assist the user. The approach of AR is opposite to that of Virtual reality; the technology is integrated in the real world to add new functionalities to mundane objects. The user perceives the "real" world, in which some elements are modified to allow interaction with the computer system.

The DigitalDesk, presented by Wellner in [102], illustrates the concept of AR. Wellner envisioned the use of a camera and a video projector to enhance the functionality of a standard desk. A camera observing the surface of the desk is used to track user's fingers. The projector provides visual feedback to the user. As a result the desk is rendered interactive. Wellner illustrates his idea with an application of a calculator projected on the desktop. The user can enter numbers either by pressing buttons on the projected calculator or by showing to the computer numbers printed on paper.

Examples presented above represent approaches of VR and AR in their "pure" form. However, there is a continuous space of different mixed reality systems in between. For instance, in the Cave system [24] users enter a special visualization chamber, where images are displayed all-around. The user is physically immersed in the virtual world and becomes a "real" intrusion to the pure VR. Archeoguide project [95], offers users reconstructed "views" of ruins in their original location. Images of the ancient

buildings are displayed in see-through glasses worn by the user. The rendered views are generated based on user's location and orientation with respect to the historical site. This system augments the perception of the reality rather than the physical world. The user experiences a mixed reality; the environment is visible, but is augmented with digital images.

In [27] Dubois and Nigay propose a classification of AR systems based on *task focus* and *nature of augmentation*. Task focus refers to the nature of the object of interaction. If the goal of the performed task is to manipulate or modify a virtual object, the task focus is "virtual". If the object belongs to physical world, the task focus is "real". The augmentation of historical sites in Archeoguide yields task focus to be real. On the other hand, DigitalDesk calculator serves to compute numbers that exist only in the memory of the computer, thus the task focus is virtual.

Nature of augmentation describes whether the augmentation results in an increase of the number and/or quality of tasks that the user can perform, or the augmentation results in an increase of information or information quality provided to the user. In the former case, it is the task *execution* phase that is augmented. In the latter, the augmentation is applied to the *evaluation* phase (perception). The DigitalDesk calculator augments the execution phase of performing calculation by creating the calculator's graphical interface on the tabletop. The Archeoguide, on the other hand, augments only user's perception of historical sites.

We propose to extend this classification by a characteristic that we call *augmentation focus*. Similarly to task focus, its extreme values are "real" and "virtual". If an AR system augments a physical object, the augmentation focus is "real". As a counterpart, an AR system that augments only user's perception of the world, the augmentation focus is "virtual". The DigitalDesk calculator is projected on a tabletop augmenting its functionality, thus augmentation focus is "real". On the other hand, if the DigitalDesk calculator would be visualized in see-through glasses worn by the user, the table would not really be augmented. The augmentation would concern only user's perception of the table, therefore the augmentation focus would be "virtual".

These three characteristics of AR systems are orthogonal and offer a Cartesian space for classification of AR systems (see figure 2.1). The line defined by virtual task focus and virtual augmentation focus, defines VR systems.

In the context of this study, the notion of an *interactive space* is grounded in the physical world and refers to the properties of physical objects. This implies that the augmentation focus must be real. As a result the interactivity of a space is exhibited by all persons operating and observing it. On the other hand, our definition of interactive space does not apply to VR systems.

An interactive space can offer the ability to modify or operate both physical and virtual objects. Similarly, nature of augmentation can be of any kind. This definition maps interactive spaces on the "task focus"-"nature of augmentation" plane within the presented classification space. This plane describes systems that are common for augmented reality and ubiquitous computing, that we present next.

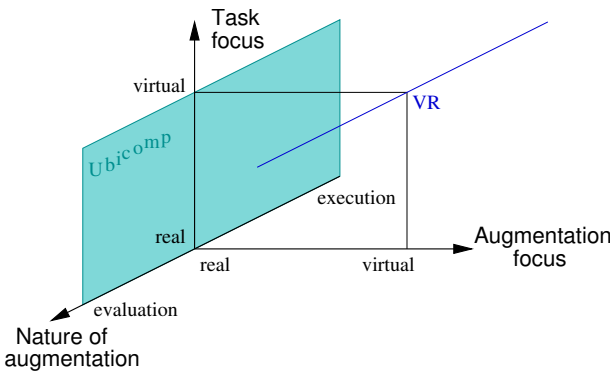

Figure 2.1: Classification space for Augmented Reality (AR) system

2.1.2 Ubiquitous Computing and interactive spaces

In his seminal paper on Ubiquitous Computing [101], Weiser foresees that "specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence". Though this concept is closely related to augmented reality, augmenting physical objects is only one of possible approaches to achieve ubiquity. Alternatively, ubiquitous computing can be implemented by integrating conventional computing resources into a unified system bound to the environment. In fact, the integration of computing devices to offer new functions is a common characteristic for both approaches to Ubiquitous Computing.

MediaCups [5] and SmartIts [38] projects represent the object-augmentation oriented stream of Ubicomp. They focus on creating new functions by adding computing and communication abilities to artifacts. In the case of MediaCups the artifacts are; coffee cups called Mediacups, door plates and arm watches. Each augmented artifact has very limited abilities, but it can communicate with other computing devices. For instance, the Mediacup can sense temperature and accelerations. The number of Mediacups filled with a hot drink can serve to infer the number of persons present in a room. The number of room occupants is then displayed by the augmented door plate.

The EasyLiving [13] project is an example of ubiquitous computing created using mainly traditional interaction devices. There are no electronically augmented artifacts, but users can interact with the system using mice, keyboards and several displays available in multiple locations. Additionally, stereoscopic cameras track users as they move within the EasyLiving environment. The aim of this project is twofold; to develop an "architecture and technologies for intelligent environments", and to explore new interaction scenarios in a home-like environment enhanced with computing.

The EasyLiving environment also offers new functionalities to its users. For instance, when a user logs-in to a computer next to the entrance door and then moves to the couch in the living room, her/his opened session follows and appears on the closest display device.

The EasyLiving and MediaCups projects propose two opposite approaches for ubiquitous computing environments. In the MediaCups project computers are considered as secondary artifacts, and the stress is put on unobtrusive augmentation of artifacts. On the contrary, in EasyLiving project, computers in their classic form (mouse, keyboard and display) remain the main interaction resource. The approach to Ubicomp in iLand [83] project can be classified between the two above. In iLand computers are embedded in furniture. Chairs equipped with tablet computers, a table with wide touch-sensitive screen and a wall size touch-sensitive display are used to support "cooperative work of dynamic teams with changing needs".

Above examples show that ubiquitous computing environments are often populated with a large number of interaction devices. In iLand or EasyLiving projects users are able to reach an interaction device any time. In this context, one can claim that these environments are interactive spaces as a whole. On the other hand, MediaCups project is oriented toward perception of human actions rather than interaction. Users are not aware of their interactions with MediaCups. Here, the space is perceptible rather than interactive.

In the context of this study, the term *interactive space* describes an environment where interfaces allowing explicit interaction are within users' reach anywhere. The adjective *interactive* is used only to describe interaction resources that can handle both input and output. Since the term "anywhere" is somewhat vague, our definition of interactive spaces reflects how we imagine the future of everyday human computer interactions, rather than provides a new method for classifying computer-enhanced environments. Interactive spaces form an approach to Ubicomp that aims at minimizing the gap between the virtual and physical worlds, through the integration of the computer system with the environment. The integration can be done either through proliferation of interaction devices and modification of the environment like in EasyLiving, or through seamless augmentation of artifact like in DigitalDesk. In any case, any person present in the space can observe and manipulate interaction resources. The interfaces are embodied, and augmentation focus is "real" (see section 2.1.1).

In interactive spaces, the user is free to move and to choose where to interact with the system. To support user mobility the interactive space must offer mobile access to its interface. We discuss the issues related to mobile computing in the following section.

2.2 Steerable interfaces

Steerable interfaces offer a particular form of mobile computing. However, interface steerability is rarely taken in account as a major property of interactive systems. In this section, we analyze how mobility is exploited in interactive systems, and define the two alternative facets of interface mobility: steerability and portability. We draw our definition of steerable interfaces on the definition provided by Pingali et

al. [64]. We illustrate the concept of steerable interfaces with a brief description of three state-of-the-art systems, in which interface steerability occurs. Finally, we identify three basic properties of steerable interfaces (controllability, observability and non-preemptiveness), and refer them to the presented state-of-the-art examples.

2.2.1 Mobile computing and mobile interfaces

Mobility characterizes anything that can move or can be moved, i.e. anything that can change position or orientation with respect to its environment. For humans mobility is natural and ordinary. In general, we do not have to think about how to move or how to interpret movements of others. Mobility is an integral part of our life. On the contrary, information technology only recently started to discover this propriety as a means to enable new forms of interaction.

In [75] Renevier identifies three main research streams related to mobility: nomadic computing, ubiquitous computing, and context aware systems (with an emphasis on localisation awareness). While Renevier illustrates nomadic computing with laptop computers and handheld game platforms regardless their communication abilities, nomadic computing is usually referred only to computers allowing mobile access to information space. Nomad users of information technology are free to move and travel with their computers and use them anywhere they go. Similarly, ubiquitous computing (see section 2.1.2) alleviates the spatial constraints inherent to desktop computers. But instead of relying on personal portable devices, ubicomp is focused on providing interfaces spread in the environment. The third category, context aware systems, is less discriminative, as context of use can be exploited by both nomadic and ubiquitous systems. Typically, context aware systems can adapt services they offer based on the localization of the user.

The Archeoguide [95], Mediacups [5] and EasyLiving [13] all focus on the mobility and localization of users. In Archeoguide, representing nomadic computing, images of reconstructed historical sites are rendered based on user's location. Mediacups and EasyLiving projects offer new functionalities enabled thanks to localization of users (see section 2.1.1 and 2.1.2).

Mobility is a general concept and can be exhibited differently depending on the object to which it refers. The above examples show that mobility in computing often refers to humans. Alternatively, it can characterize interaction resources, including both interfaces and data.

The ability to conveniently move data between workstations is a fundamental requirement for modern computer systems. The mediaBlocks [92] are portable, electronically tagged wooden blocks, that can be used to "move" digital data between computing devices. In reality, mediaBlocks do not store data to be moved, but only reference it and the actual transfer are done via network connections. The "Pick-and-Drop" interaction technique introduced in [74], is somewhat similar in spirit to

mediaBlocks. Here, data can be transferred between computers with help of an electronically enhanced pen.

Many researchers exploit portability of handheld computers to devise new interaction techniques. In [85, 84] personal digital assistants (PDAs) are used in conjunction with stationary displays. Their relative proximity influences how information is visualized. In Caretta [85] the PDA can be used as a virtual pencil to draw strokes on the stationary display.

While mobility of artifacts enabled with computing draws on humans' natural skills, mobility of virtual objects is more difficult to apprehend. Users have to learn how to manipulate virtual objects using interaction techniques based on metaphors of some sort of real-world activity. The "drag-and-drop" metaphor is used in PointRight system [42] to move digital documents between spatially distributed computers. The virtual objects can be dragged using a mouse between several displays as if they would form a continuous workspace. In iLand [83] users can "throw" documents within a wall-size display to each other. The direction and acceleration of the throw are derived from the dynamics of the user's initial drag movement. Similarly to the "Pick-and-Drop", in iLand [33] users can also pick a digital item and put it somewhere else on the DynaWall display. The DynaWall is a touch sensitive device, which allows direct interactions with hands and fingers.

Mobility is exploited in HCI in a variety of ways, and we could multiply examples. It is inherent to many development approaches. Virtual reality, augmented reality and ubiquitous computing, all share the need for mobility; whether it concerns the user or virtual objects. Surprisingly, mobility is rarely identified as the key feature enabling interaction; rather it is taken for granted. We notice the lack of analyzing the influence of mobility on HCI. This is particularly flagrant when considering interactive spaces, where mobility of both users and interaction resources can introduce complexity. For instance, how can users of iLand's DynaWall know that during the "Pick-and-Drop" interaction, the DynaWall is blocked for other interactions? The DynaWall is composed of three touch-sensitive rear-projection whiteboards each supporting input from only one user at a time. Therefore there is a risk that dragging an object across the DynaWall would break when crossing the border to another DynaWall segment used by another person.

We propose to analyze interactive spaces in terms of mobility of the available interfaces. In particular, we focus on an emerging class of mobile interfaces that we call steerable interfaces. We present it in the following sections. We also discuss the relation of steerable interfaces to their portable counterparts.

2.2.2 Steerable interfaces – definition

Pingali et al. [64] define a steerable interface as "an interactive interface to computing that can move around a physical environment and appear on ordinary objects or surfaces or even in empty space". Authors enumerate six key characteristics "an

interface should ideally exhibit [...] to qualify as a steerable interface":

1. *Movable output interface* is the primary property of steerable interfaces. Pingali et al. note that this characteristic applies to interfaces based on different modalities. They list the most common two; visual and audible output.

2. *Movable input interface* is the second property that a steerable interface must have. Authors propose that "a steerable interface should serve as both input and output interface to computing".

3. *Adaptation to user context.* Suggests that steerable interfaces should be able to adapt to user's characteristics like location, gaze direction and to the context of use. "For example, an interactive display should be oriented towards the user and be close enough to the user to facilitate interaction." Pingali et al. suggest that it is the steerable interface responsibility to take into account the topology of the interactive space and the spatial relations of users with respect to the space. This requirement is stated more explicitly by the fourth characteristic.

4. *Adaptation to environment context.* "[...] a steerable interface typically needs to be aware of, and reason about, the geometry and properties of the environment, and relate the user as well as input and output devices to a model of the environment." Additionally authors stress the need for adaptation of the interface to characteristics of the environment. For instance, a projected graphical interface should compensate color changes resulting from projection on textured surfaces. A sound-based rendering should be adapted in function of noise present in the environment.

5. *Device-free interaction* refers in the first place to direct interaction, i.e. without any transducers like mice, laser pointers or pens. This requirement is related to the proposed implementation of steerable interfaces; "a steerable interface should be able to move on to different surfaces and objects and areas in an environment without the need to place special devices in these places."

6. *Natural interaction* requirement reflects authors' orientation towards ubiquitous computing, where interfaces should be intuitive and easy to use. This is rather en expected quality, resulting from "ubiquitous access and adaptation to user and environmental context", rather than a property of steerable interfaces.

Pingali et al. identifies a new class of interfaces together with its main characteristics. However, the presented list of "key characteristics" reflects the authors' preference for applications such as a steerable interface that automatically follows the user in an environment [65, 86], where the steerable interface is controlled by the computer, rather than by the user.

A user-controlled steerable interface does not necessarily need to adapt to "user context". In fact, in some situations, such adaptation would be impossible or even undesirable. For example, if a user decides to move the interface to a distant location to work with it later, the system is unable to infer the future context of use at the new location.

Although adaptation to user and environment context, as well as device-free and natural interaction requirements, are desirable characteristics for interactive systems in general, they are not discriminative for steerable interface. From the six characteristics listed by Pingali et al. only the "movable output interface" and "movable input interface" are inherent to steerable interfaces. In fact, these two characteristics are common to both steerable and portable interfaces, and describe mobile interfaces in general. We discuss the proprieties of mobile interfaces is section 2.2.3.

Rather than binding steerable interfaces to a particular technology or class of applications, we propose to define steerable interfaces based on a user-centric analysis of the nature of interface steering. Below, we provide definitions of the computer interface, the steerable interface, and the portable interface.

Who is steering the interface? Steering is an activity involving two entities: the governing one, and the controlled one. The governing entity acts (possibly with an actuator or actuators) on the controlled entity to modify some parameter of its state, e.g. its location or orientation. In order to define steerable and portable interfaces, it is necessary to specify the involved entities. The characteristics listed by Pingali et al. [64] indicate that the provided definition concerns a particular case of interface steerability, where the governing entity is limited to the computer-system. Alternatively, steering can be performed by humans. In such case, the computer system executes orders given by the user, and moves the interface accordingly. Because we believe that humans are central to HCI, we prefer to adopt a user-centric approach and describe steerability (or more general mobility) from user's perspective.

Interface definition. An interface is "the place at which independent and often unrelated systems meet and act on or communicate with each other"[1]. In the IT community, the term interface has multiple significations. Its meaning can vary from a plug that allows connecting peripherals to a computer, through an interaction device such as keyboard or mouse, to an application graphical front-end.

The user interface of desktop computers is usually associated with the three most common hardware devices used for interaction; a keyboard and a mouse for input and a display screen for output. In interactive environments the identification of the user interface is more difficult. For a table augmented with a GUI created using a video projector and a camera, the hardware components of the interface are the video projector, the camera and the table. However, from the user perspective only the

[1]Merriam-Webster Online Dictionary: `http://www.m-w.com`

table and the projected GUI form the computer interface, the rest of the hardware infrastructure is discarded.

We define the *user interface to computing* as an aggregate of means by which users interact with a computer. For the reasons of brevity, we refer to it as to the *interface*. The term "means" refers to entities that form the outmost perceivable boundary of the computer system. The boundary can be formed by physical entities, such as mice, keyboards, a tabletop, or by information content bound to physical entities, e.g. sound bound to the air as a medium for speech-based interaction, or images bound to a display surface for projection-based interaction.

Steerable interface definition. Steerability is closely related to "control". However, controlling an interaction resource does not imply that the corresponding user interface is steerable. For instance, a computer controlled lamp is an interaction resource creating an output user interface in the form of a light beam. Though the light intensity of the lamp can be controlled by the computer, the beam is not steerable. On the other hand, the control over a number of spatially distributed homogeneous interaction resources can result in interface steerability. The light beam becomes a steerable output interface, if the computer system implements a control function of mutually exclusive use of lamps, i.e. if by brightening one lamp the other is dimmed. Steerability is synonymous with "spatial controllability".

We propose the following definition of *steerable interfaces*: an interface to computing that can be relocated in a physical environment and its position during transitions and at each location is set by, or with the intermediacy of, a computer system. For brevity reasons, we denote *steerable interfaces* as SUI (Steerable User Interface).

In contrary to the definition presented by Pingali et al. [64], our definition encompasses approaches other than augmented reality. If a graphical interface rendered on a sub-area of a display can be moved on the display, it is considered as a steerable interface. The output interface is limited to the sub-area of the display used to render the GUI. The rest of the screen is not involved in interaction, thus is not considered as part of the GUI. In other words, a window dragged on the monitor display using a mouse pointer is also a steerable interface, or more precise a steerable output interface.

Portable interface definition. In order to better understand characteristics of steerable interfaces, we refer their description to that of more conventional portable interfaces.

By analogy to steerable interfaces, an interface is *portable* if it can be relocated in space, and its position during transitions and at each location is directly controlled by the user through physical contact with the interface.

Again, this definition can be applied for both input and output interfaces separately. For instance, a wireless keyboard is a portable input interface. Its portability can be used for more than convenience of use. In EasyLiving project [13] the position of a

wireless portable keyboard helps to determine which person interacts with the system.

2.2.3 Properties of steerable interfaces

Interface steerability defines the ability of an interface to be moved by the system within a physical space. Interface portability corresponds to the ability of a human actor to relocate the interface by grabbing and carrying it. Though both steerability and portability are examples of mobility, their nature differs in many ways.

In general, interfaces can be categorized according to their properties. Coutaz et al. [41] note that "property is the capability of an entity to fulfill a particular function", and that some properties of an entity can be inferred from attributes grounded on physicality. For example, the property of *portability* can be deduced from object attributes such as *weight, size and solidity*. On the contrary, the property of *steerability* is not directly related to physical attributes of the interface.

In contrary to portable interfaces, the location of steerable interfaces is not set directly by the user. It is always the computer system that renders the interface to a particular position in space, based on control signals given by the user. In this study, we presume that, when there are no control signals a steerable interface remains immobile. In this state, the SUI is much like a conventional interface with the notable exception of the ability to be moved. When control signals are issued, regardless whether the user generates them consciously or not, the interface changes location. Because a moving interface eventually stops, we consider movements of the interface transitory. In other words, steerable interfaces are likely to have a dual nature that results from their two basic states: standstill and transition or movement.

Steerable interfaces discern from conventional interfaces by their ability to move. Compared to conventional interfaces, the description of a steerable interface must be therefore extended with the characteristics of the interface in the transition state. We must also define how a steerable interface goes from standstill to transition state; i.e. the interaction techniques and the corresponding part of the interface. We discuss in more detail the issue of interaction techniques allowing the control of interface location in chapter 6.1. Here, we focus on the properties of interface in the transition state.

In terms of the transitory state of steerable interfaces, we identify three characteristics that are fundamental for user perception of the interface:

- *Observability* defines the extent to which the user is able to determine the location of the interface. In the case of a graphical interface, observability describes the user's ability to observe movements of the mobile interface.

- *Controllability* refers to the user ability to control the spatial position of the mobile interface between fixed locations. Controllability has two facets; it can be either continuous or discrete. A continuously controllable interface can be directed to a location any time, including when it is already on the way to a

location. Discrete controllability implies the existence of a state in which the user cannot control the interface.

- *Non-preemptiveness* corresponds to the ability of the interactive system to support the user in performing tasks in free order. In the context of mobile interfaces, non-preemptiveness describes whether the interface can support during its movement the user in executing tasks that are not related to controlling the interface location. In other words, a steerable interface is non-preemptive if it remains interactive during movement. Alternatively, an interface that forces the user to stop interaction for the transition period is preemptive.

In the following, we illustrate how these characteristics are reflected in the state-of-the art systems in which steerability is present. We chose three systems in which the interface steerability renders the environment interactive: the EasyLiving project [13], the 'Traveling Tic-Tac-Toe' [68], and the 'Access spotlight'[2].

The EasyLiving introduced in section 2.1.2, is a project exploring new interaction scenarios in a home-like environment augmented with large number of computing and interactive devices. Figure 2.2 shows a view of the EasyLiving environment.

Figure 2.2: The EasyLiving environment (from [13])

In terms of output devices, the EasyLiving environment is equipped with several flat-panel displays, including a wall-size back-projection screen, loudspeakers and computer controlled lights. For input, users can operate a wireless keyboard and mouse

[2] `http://www.accessproject.net/index.html`

placed on the table in the middle of the room, or they can use a keyboard and a mouse colocated with a display next to the entrance door. The EasyLiving environment is also equipped with a number of cameras that track both users and the wireless keyboard and mouse to allow for implicit interaction with the system.

In EasyLiving two interaction scenarios involve steering of the interface:

1. The user interface can be automatically redirected by the system to the interaction resources that are closest to the user, provided that the user already logged-in to the system, and that there is no other user blocking the interaction resources.

2. A user can send her/his interface to another set of interaction devices (i.e. display, mouse and keyboard), by selecting them on a graphical interface visualizing a model of the EasyLiving room.

The Traveling Tic-Tac-Toe is a pervasive version of the conventional 'Tic-Tac-Toe' game. A demo installation of this game runs in IBM Epcot Center demonstration area in Orlando, Florida [68].

In order to play a cross or a circle in the game, players touch the projected grid with a hand (figure 2.3). Each time a player touches the grid, the projected game board is moved to a randomly chosen location. To continue the game, next player has to first find the projected grid.

Figure 2.3: The Traveling Tic-Tac-Toe (source [68])

Both input and output interface of the Traveling Tic-Tac-Toe are steerable. The interface steerability adds to the game elements of physical exercise making the traditional game more attractive.

Access: A Video-based Tracking Spotlight is an art installation in which individuals are tracked with a steerable spotlight-and-sound system[3]. The installation

[3] `http://www.accessproject.net/index.html`

was shown publicly during several scientific and artistic events (e.g. Siggraph'03 conference).

The installation is composed of two interfaces; a robotic audiovisual output system and a conventional workstation running a web-browser (see figure 2.4). The web-interface allows users to pick a person from the crowd observed through a ceiling mounted camera. The chosen person is then automatically followed by the robotic spotlight and the acoustic beam that projects audio. The audio projector is an ultrasonic loudspeaker emitting a very narrow audio beam[4]. The tracked individuals do not know why they are being tracked, "nor are they aware of being the only persons among the public hearing the sound. The web users do not know that their actions trigger sound towards the target. In effect, both the tracker and the tracked are in a paradoxical communication loop."[5]

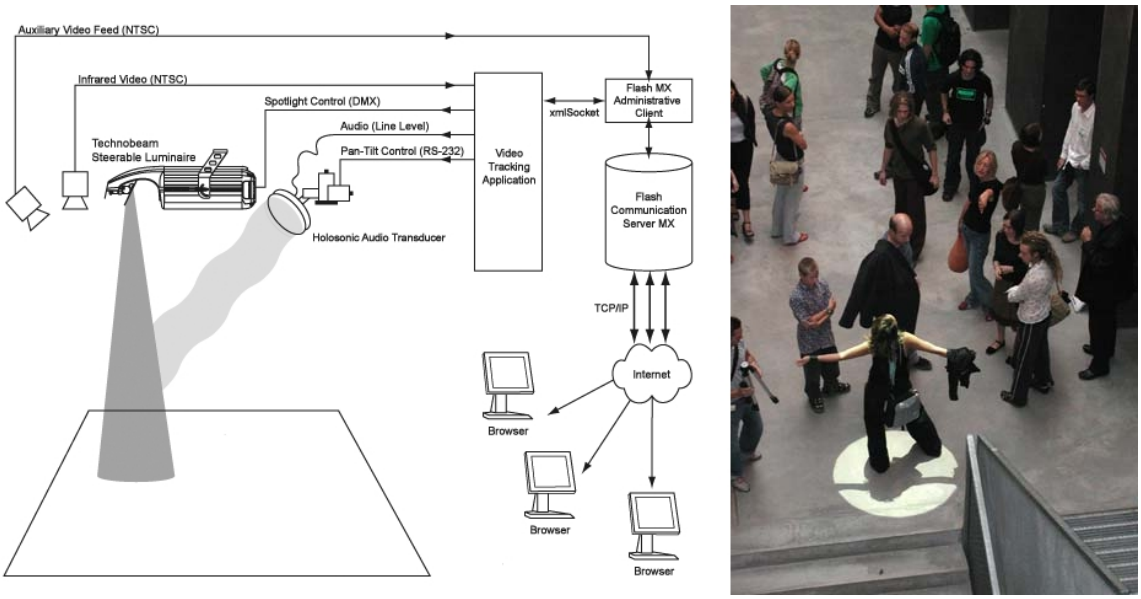

Figure 2.4: Diagram of the Access installation (source: `http://www.accessproject.net/`).

The 'Access spotlight' offers a very unusual interaction scheme. The web-user controls which person is tracked with the audiovisual spotlight, but is not fully aware of the resulting interaction (i.e. does not know that the person is also tracked with an audio beam). The tracked person can control the location of the audiovisual beam, but does not know why. Although its awkwardness in terms of interface design, 'Access spotlight' is an interesting application of interface steerability for novel forms of human-computer and computer mediated human-human interaction.

[4] `http://www.holosonics.com/`

[5] `http://www.accessproject.net/concept.html`

The steerable interface of 'Access spotlight' is limited to the robotic audiovisual spotlight.

Summary of the state of the art. While observability, controllability and non-preemptiveness are inherent to most portable interfaces, they are optional for steerable interfaces. For instance, in both EasyLiving and Traveling Tic-Tac-Toe the transitory state is not observable. In EasyLiving the observability of the interface during transitions is limited by the technical infrastructure, i.e. the graphical interface can be rendered only at locations where displays screens are available. Similarly, the input interface of EasyLiving can be relocated only between the two available sets of keyboards and mice. On the contrary, the Traveling Tic-Tac-Toe game is rendered on surfaces using the 'Everywhere Display' projector [67], thus potentially the game could be rendered also during movements. In this case, observability of the interface is omitted intentionally to make the game more demanding on player's skills. Finally, in the 'Access spotlight' project, the steerable interface is rendered observable by the projection of a light beam. Here, the observability of the interface is fundamental, because if the spotlight would be limited to a sound beam, it is likely that some of the tracked persons would not notice that they are spotted by the beam. As a result, the Access experience would be less interesting for both the tracked people and the web users.

The first of the EasyLiving interaction scenarios and the interaction with the Access spotlight, offer a similar degree of controllability; the interface follows its user. In Access spotlight project, the location of the steerable interface is coupled to the location of the tracked person. The tracked person can control the position of the beam by walking or running within the beam range limit. In fact, controlling the location of the spotlight is the only type of supported input for this steerable interface. On the other hand, the tracked individual cannot initiate nor stop the interaction; it is an other user of the system that chooses a person to track with the web interface. In EasyLiving, the ability of the interface to follow its owner (the user) aims to render the use of the interactive environment easier. In contrary to the above examples, the lack of control over the location of the interface is the main feature of the Traveling Tic-Tac-Toe. In fact, it is the key idea enhancing the game.

In the EasyLiving project, the interface can switch between ensembles of conventional interaction devices (display, keyboard and mouse), it disappears from one location to immediately appear at a new location. This type of interface steerability is known as *teleporting* [76]. The transitory state of a teleported interface is neglected.

In all three systems presented above, the interface in the transitory state is preemptive, i.e. once the action of relocating the interface is started, it has to be completed to use the interface for other tasks. One can imagine however, a moving interface that remains fully functional during transitions, and thus allows continuing tasks that are not related to the control of the interface position. For instance, an audiovisual

spotlight supporting speech-based input could be used to create such non-preemptive steerable interface. Similarly, a graphical interface following the user or moving at a constant speed could be used in motion.

The presented examples illustrate that steerable interface can be implemented in many ways. The properties of steerable interfaces should be used for classification purposes, rather than be considered as a set of desired characteristics for a "good" implementation. The issue of developing interaction techniques that are appropriate for controlling an interface merit further discussion. We present our approach to interaction with steerable interfaces in chapters 5 and 6.

2.3 Summary of the chapter

In this chapter, we defined the concepts of an *interactive space* and a *steerable interface*. Interactive spaces are closely related to the notion of ubiquitous computing, which in turn draws on concepts such as virtual and augmented reality. We briefly introduce the main ideas behind each of them, and present a simple classification space. We identify the steerable interfaces as a means enabling to render spaces interactive. Steerable interfaces form one of many possible ways towards ubiquitous computing.

Mobility is the common denominator for steerable and portable interfaces. It plays a tremendous role in modern information technology. Mobility applies to both physical entities such as people or interaction devices, and to virtual entities such as data or graphical objects. In this study, we are concerned with the mobility of an interface, an abstract figure build from physical entities, but not bound to them for its lifetime. In order to eliminate ambiguities, we provide definitions of: *(1)* the interface, *(2)* the steerable interface, and *(3)* the portable interface.

We refer our definition of steerable interface to the definition of Pingali et al. [64]. We point-out the differences between the two approaches. In particular, our definition is not limited to interfaces implemented using a particular technology.

We identify three properties that are fundamental for the description of steerable interfaces, in particular during transitions between stationary locations: observability, controllability and non-preemptiveness. These properties form a classification space for steerable interfaces, and a conceptual framework helping to identify the often subtle differences between implementations of steerable interfaces.

We illustrate the application of the presented properties by analyzing three state-of-the-art examples of systems in which interface steerability occurs. Though the properties are primary for the classification of interaction with steerable interfaces, their presence is not a necessary condition to classify an interface as steerable.

We note that observability of steerable interfaces is dependent on the technology used to create the interface. We discuss this issue in chapter 4.

Because, graphics is the most common output for computers, in the following we limit the scope of considered steerable interfaces to those having a graphical output.

Third Chapter

User-centered interactive system design

The implementation of an interactive system or interactive system's components is a software engineering problem, and as such, it can be structured into a number of activities organized around the software life-cycle. The life-cycle of creative engineering projects can be modeled in various ways depending on the objectives.

The "V-model" derives from the waterfall life-cycle model [79], and was developed as a part of software development standardization effort of the German Federal Authorities [62]. In fact, the V-model is the acronym for a standard that also regulates quality assurance, configuration management and project management, but it is often used to relate the software development (SWD) sub-model of the V-model standard.

Figure 3.1 shows a common representation of the V-model. It is composed of seven development activities, presented graphically on the branches of a V sign. The activities can be structured into three groups based on their nature: *(a)* conception, which comprises requirement analysis and specification definition, *(b)* implementation of the designed system, and *(c)* validation, comprising tests, evaluation and integration. In fact, the standard is more detailed and specifies nine activities, but for the need of this brief life-cycle models review the simplified version is more adequate.

Life-cycle models regulate the definition and order of activities required to develop a system. In the context of a research project, life-cycle models are rarely followed to the letter. For example, it is often unnecessary to elaborate detailed specification documents when the person developing the software is also its sole final user. In this study, the design of a software component is usually driven by brainstorming and discussions, and validation is limited to the integration and evaluation of chosen performance characteristics.

Process life-cycle models describe the "what and when", but do not define how different development activities should be realized. This chapter presents our approach to software design, development and evaluation. In the following section, we present the concept of Service Oriented Architecture (SOA). We propose a number of requirements

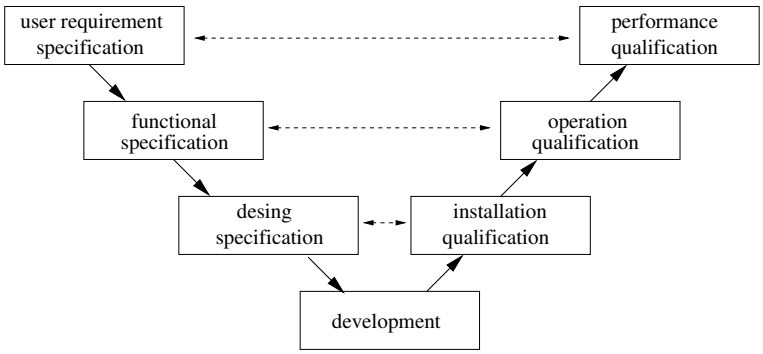

Figure 3.1: Common presentation of the V-model process life-cycle model

services must meet in order to be successfully used in interactive systems. Section 3.2 defines requirements that have to be meet in order to optimize usability. To position our work with respect to established software evaluation standards we refer to the ISO 25000 standard.

3.1 Services for interactive systems

The use of novel input and output devices makes interactive systems more complex. As each input method may have different requirements concerning its execution environment and hardware resources, the system integration is rendered difficult. Fast prototyping requires code reuse and modular decomposition of system elements. What's more, modules are likely to be development on different platforms, so interoperability is a must. These issues call for sound system architecture.

In this section, we present the concept of Service Oriented Architecture (SOA) design (subsection 3.1.1), together with a definition of a service and its properties.

3.1.1 Service Oriented Architecture

Service Oriented Architecture (SOA) is an approach to software development, where programs are structured to collections of interoperable services. Recently, the concept of services gained much attention within the web community. Several standardization bodies such as W3C, OASIS, and the Web Services Interoperability Organization work on developing a standard for web-services.

The W3C defines the SOA as a form of distributed system characterized by the following properties [99]:

1. Logical view: "The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it *does*, typically carrying out a business-level operation."

2. Message orientation: "The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves."

3. Description orientation: "A service is described by machine-processable meta data."

4. Granularity: "Services tend to use a small number of operations with relatively large and complex messages."

5. Platform neutral: "Messages are sent in a platform-neutral, standardized format delivered through the interfaces."

6. Network orientation: "Services tend to be oriented toward use over a network, though this is not an absolute requirement."

Based on these properties, Vogels [97] defines a service as a "software that can process an XML document it receives through some combination of transport and application protocols". The document "contains all the application-specific information that a service consumer sends to the service for processing". Note that, there are no constraints on the transport protocol, and the key to communicate with a service is the content of the XML document. Naturally, in order to understand each other, both service client and the service must share the same document description.

The W3C definition of SOA reflects the web-oriented perspective of this consortium. Similarly, Vogels' definition applies rather to web services than to services in general. Both should be considered as a particular case of service-orientation. In the following subsection 3.1.2, we detail our view on services and their properties.

3.1.2 Service definition

We define a service in terms of its properties, these are: abstraction, isolation and contract. In order to clarify the notion of a service, we discuss service properties in more details below, as well as their impact on our approach to software development.

Abstraction. A service provides an abstraction of the functionalities it offers, by defining the output (results) and necessary input information. A service offers its functionality to a service client (service requester), which implies that a service has some sort of communication abilities. There is no service that has defined neither input nor output, though both are optional.

When developing an interactive system, "abstracting the [user] input [is] the most time consuming and challenging piece of application development" [48]. The appropriate level of abstraction depends on the target user of the service. Services for interactive applications should provide only interaction-relevant output. For instance,

vision-based services should hide vision-specific information. Even though coupling processing results with a confidence factor might be richer, this information is, in most cases, of no interest to event-driven interactive system. For instance, in case of a robustness-related failure, the service should notify the client application, rather than provide incorrect or distorted information along with a (low) confidence factor.

A mouse driver can be implemented as a service. With the right level of abstraction, it would provide output corresponding to mouse movements and button state changes. Though, optical mice use normalized cross-correlation to determine the direction and speed of movement, the correlation coefficient is not send to service clients. The use of services can make the interactive applications agnostic about the nature of the input system: there should be no difference if the user realizes a pointing task by manipulating a mouse, moving their fingertips observed by a camera, or waving a laser pointer.

Consequently, the services abstracting user input should *(a)* render the sensor-related aspects invisible, and *(b)* generalize the input used for a given task.

Isolation. Services are autonomous with respect to their clients. There is no state sharing, i.e. the service behavior does not depend on the service requester state, but on its internal state and available input data (including requester message). On the other hand, services share with other services and clients a description of their abilities, i.e. the definition of their inputs and outputs.

Contrary to the W3C definition of Web Service Description (WSD) [99], our service description defines a service only in terms of its inputs and outputs, together with the corresponding transport protocols and the addresses. Message formats and data-types are optional information, which can be either specified in the service description or in the service documentation.

Services may need to be used by multiple applications, possibly running on different machines – for performance or geographic reasons. For instance, a video capture service might be used concurrently by a surveillance system located on a centralized server, a video-conference application and a motion capture service, both co-located to the camera. Since "a particular piece of input can be used for many different types of output" [48], information generated by a vision-based service must be shareable, and both remotely and locally accessible. The use of adapters, aggregators, or supervisor patterns allows service-based systems to be extensible and embeddable in other services or in applications. What's more, we impose on services the necessity of a mechanism for service discovery and inspection. Service discovery ensures that a client can find and identify a needed service. Inspection means that services offer a description of their capabilities to the entities that will use their services.

Contract. In order to satisfy the user experience, interactive systems have to fulfill sever constraints. This statement implies, that services designed to be a part of an

interactive application have to take these constraints into account. The user-centric criteria a service should meet depend on the functionality it provides. For instance, for positional input (mice, trackpads, laser trackers, etc.), the relevant criteria are latency, precision, robustness, and autonomy.

- *latency* is implicitly under a usability threshold for standard tracking input devices (the textbook value is around 50 ms);
- *precision* is also implicitly defined for a mouse/trackpad (under 1 display pixel, scaled with respect to the device gain)
- *robustness* is generally "absolute". In other words, the device or system either works or doesn't. For instance, an optical mouse "just works" as long as used on an adequate surface, and stops emitting positional events as soon as it is raised from the surface.
- *autonomy*, and in particular automatic setup and maintenance, is also tacit for traditional devices.

It is the role of the service designer to explicitly state in contract form its limitations with respect to relevant evaluation criteria. In section 3.2, we define software evaluation criteria that are relevant to services we develop.

3.1.3 Service implementation

In the previous subsection, we defined services as self-contained software that is well defined in terms of what it does, and what it needs. Services can be discovered and located by other services by publishing their description. Services do not depend on the state of other services. In SOA interactive system input devices (physical sensors) and output devices (actuators) are abstracted by corresponding services. In this section, we discuss the implementation of services.

The foundation of our services is the "BIP/1.0" protocol[1]. BIP stands for Basic Interconnection Protocol. Its implementation is structured in three layers, which implement the following features:

1. *The communication protocol*, which consists in an exchange of ASCII messages with single line message headers over a TCP or UDP socket.
2. *Service registration*, allowing services to advertise themselves using the DNS-SD[2] convention over Multicast DNS[3].

[1]Protocol specification is available at `http://www-prima.inrialpes.fr/prima/pub/Publications`

[2]DNS-Based Service Discovery: browsing and discovery of services using DNS queries. Described at http://www.dns-sd.org/

[3]http://www.multicastdns.org

3. *Service inspection and control* – an XML-based scheme for querying service description and controlling service parameters. This assures a mechanism for automatic service discovery based on the service description.

These layers can be used separately, so it is possible to implement a service without the ability to discover other services, or the ability to advertise itself. Provided that a client knows the TCP port of an input or output of a BIP-service, it can connect to the service directly without having to discover it via DNS-SD.

To allow for low latency while preserving reliability, BIP provides traditional socket-based (TCP and UDP) communications. The TCP link is used to guarantee the connection to a client, and to allow reliable, high-latency communications. The UDP link can be used for low-latency communications.

The BIP protocol allows development of services as independent, black-box processes that use only serialized communications. Nevertheless, it is the developer responsibility to ensure the right level of service abstraction and isolation, as well as to evaluate it and establish its specification (contract).

The next section provides insights into how software, such as a service, can be evaluated and what are the criteria relevant to interactive application design.

3.2 User-centered requirements

In the design of an interactive system the user and her/his needs must be in the center of attention. Since 1970 a design philosophy called user-centered design (UCD) has evolved around the idea of giving an extensive attention to the needs, wants and limitations of the end user at each stage of the design process. Approaches such as Co-operative design [35], Participatory design [80] or Contextual design [9], are variations of the UCD concept, and differ in the user's role and status in the design process. In general, attention is devoted to user's expectations at both the very beginning of the development process, when specifying user-requirements, and at the very end, when the software is evaluated against these requirements.

This dissertation describes a case of interactive system design. We devise the basis for interaction scenarios that have never been envisioned before. Therefore, in most of our work, user requirements are rather imprecise and are difficult to formalize in a specification. In a sense, we devise new functionalities without knowing if users will find them useful. In fact, we assume hypotheses concerning the use of functionalities and then verify them through implementation of prototypes. For this reason, we focus much attention on the "proper" implementation of our software, as a means to ensure the usability and minimize the impact of design flaws on user's judgments.

In the following subsection, we briefly present the ISO-25000 standard as a reference for software evaluation. Based on the standard, we identify requirements that are critical for the software we develop.

3.2.1 ISO 25000 software evaluation framework

ISO 25000 standard provides a reference framework that defines the evaluation criteria for a software. Typically, requirements are divided in two sets: functional requirements that define *what* software is expected to do, and non-functional requirements, which define *how* the software operates or *how* the functionality is exhibited [19]. "Functional requirements have typically localized effects, i.e. they affect only the part of software addressing the functionality defined by the requirements. On the other hand, non-functional requirements specify global constraints that must be satisfied by the software, e.g., performance, fault tolerance, availability and security" [78].

Together the functional and the non-functional requirements determine the *quality in use* of a software. *Quality in use* refers to the "extent to which a product used by specified users meets their needs to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use" [1] (see figure 3.2).

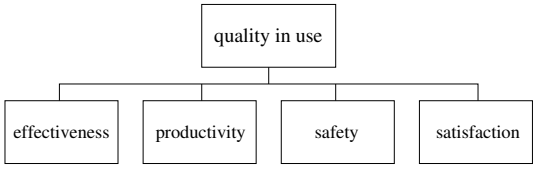

Figure 3.2: ISO 25000 quality in use definition

Quality in use is determined by the *external quality* and *internal quality* of a software. According to ISO-25000, external quality refers to the "extent to which a product satisfies stated and implied needs when used under specified conditions". Internal quality is the "totality of attributes (or characteristics) of a product that determine its ability to satisfy stated and implied needs when used under specified conditions". ISO-25000 refines both external and internal qualities into the same set of characteristics (figure 3.3).

3.2.2 Application of the ISO 25000 standard

In this study, we use non-conventional interaction devices like cameras, projectors and mundane objects to explore the potential benefits of interface steerability and portability in the context of HCI. As a consequence, we are concerned by the development of the whole interactive system; starting with subsystem(s) for sensing user actions, and ending at the computer-response rendering subsystem. These include not only software but also electro-mechanical devices. Evaluation of such devices requires a different set of criteria than that for evaluating software. Nevertheless, we consider the ISO 25000 standard as a reference, and based on it specify requirements relevant to both software and interaction devices.

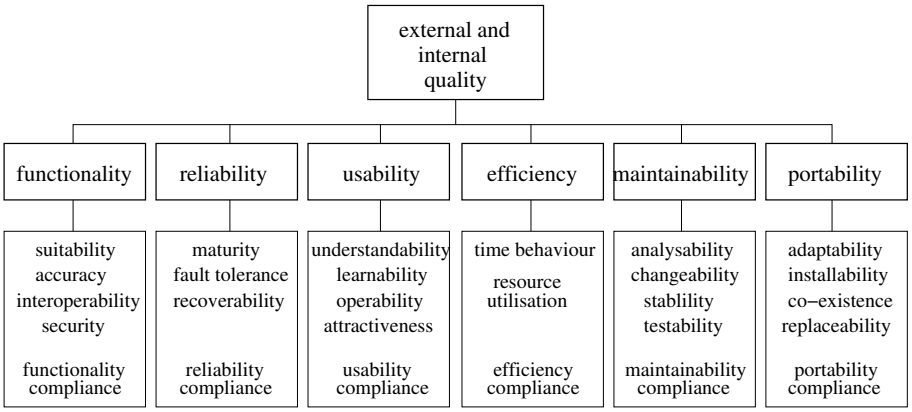

Figure 3.3: External and internal qualities as defined by ISO 25000

Software developed within this study is devised for two distinct types of audience: *(a)* interactive systems developers, and *(b)* naive end-users. The latter refers to users of the interactive system prototypes developed within this study. The adjective "naive" refers to their knowledge about the implementation and specification of our prototypes. The former are software developers interested in using our software to build interactive systems, i.e. they envision the use of computer vision as input modality, and/or projection (i.e. steerable projection) as output modality. This group of IT literate users can be refined to developers having expertise with vision and those unfamiliar with the field. As a result, our software can be grouped to: *(1)* components designed to be used by vision-expert developers, *(2)* components for interactive systems designed by developers without expertise in the computer vision domain, and *(3)* interactive system prototypes for naive users. Each of the users group has different requirements, and the defined by the ISO 25000 characteristics are not equally relevant to software quality evaluation.

Functionality is the extent to which software suits the task it was designed for. This applies both to the functional requirements compliance and to some non-functional characteristics that are vital for its correct execution, such as security and interoperability.

Due to the experimental-prototype nature of our interactive applications, the functional compliance is not critical. We can accept the loss of some functions in favor of enhancing system reliability or efficiency. On the other hand, functionality of software components is a binary quality, i.e. either the software is suitable for the task it was designed for or not, in which case the component is useless.

Interoperability of software services is enabled through the use of the BIP protocol (see subsection 3.1.3). At the same time, service-orientation implies the *autonomy* of services, both with respect to other services and to the user of the system. A service

used by an interactive application cannot require the user of the application to modify its internal parameters. This requirement is difficult to enforce for vision systems, as they usually require operator intervention for initial setup and maintenance. While acceptable in a lab setting, this is poorly suited for deployment, especially since a naive user has no necessary background to perform such operations. Therefore, we explicitly evaluate the extent to which our services are autonomous.

Reliability refers to software maturity, which in turn is defined by fault tolerance and recoverability. These criteria are of major importance for interactive systems as they influence user-experience. For instance, vision-based token tracking in [7] fails when the user hides a token, but as soon as the token reappears in camera image, it is detected and tracking is restarted allowing to continue the interaction. In this sense, reliability can be enforced during software design through identifying possible failure reasons.

Most software offers only partial reliability. It is very difficult to identify all possible sources of errors. Moreover, some failures are inevitable given certain usage conditions. Therefore, it is important to inform the user of software about the limitations of its usage.

Reliability is not limited to characterizing software. In general, reliability is the extent to which an experiment, test, or measuring procedure yields the same results on repeated trials[4]. We assume that, reliability of components responsible for sensing user actions, and rendering system response, determine the reliability of the whole interactive system. In this context, components are both the physical sensor or render device and its corresponding software controller. Reliability covers two important non-functional attributes of components:

1. *resolution*, a quality measure describing the minimal perceptible (or executable) variation of the controlled variable at given conditions.

2. *static stability*, which describes the variation of the measured value (or rendered output) of the controlled variable when it is constant.

In contrary to fault tolerance and recoverability, these characteristics can be measured and used to formulate quantitative non-functional requirements. For example, resolution of a standard computer mouse is around $0.127mm$, which roughly corresponds to the minimal movements that a person can do. The static stability of a mouse has to be small enough to ensure that the pointer rendered on computer screen would not move when the mouse is steady.

Resolution and static stability requirements apply to both sensors and actuators, and have to be determined for them separately based on how the sensor data and rendered output are used in an application. When pointing on a graphical user interface,

[4]Merriam-Webster Online Dictionary: `http://www.m-w.com`

the resolution requirements are different if the task is realized with a mouse, and if it is done directly with a finger.

Usability is the extent to which an interactive entity (software or physical object) satisfies users expectations with respect to the way the entity's functionality is exhibited and experienced by the user. It is the key feature describing interactive systems, as it defines the quality of interaction. It reflects, to some extent, all runtime characteristics of a system. Functionality, reliability and efficiency, all influence usability.

Rather than rely solely on ISO 2500, we propose to employ a usability evaluation framework. In [34] Gram and Cockton recapitulate guidelines for designing interactive software elaborated by members of the International Federation for Information Processing (IFIP). They relate properties influencing quality to software development. Therefore, they propose to analyze interactive systems from two complementary perspectives: the user's perspective and the software developer's perspective. Authors present two sets of properties against which software can be evaluated: "external properties" influencing user's experience (i.e. usability), and "internal properties" related to the developer point of view. In this thesis, we investigate how steerable interfaces could influence people working with computers, therefore we focus on the user's perspective. By "IFIP properties" or "IFIP usability properties", we refer to the "external properties" identified by Gram and Cockton.

Though Gram and Cockton state that the list of presented properties is not exhaustive, it can serve as a basis for the purpose of usability evaluation or classification. IFIP properties are structured into two factors: interaction flexibility and interaction robustness (see annex A). Each factor is refined to its own set of criteria.

Note that, Gram and Cockton provide a slightly different set of usability properties than that of ISO 25000. For instance, according to [34] *deviation tolerance* is closely related to *recoverability*, which in ISO 25000 is a property of software reliability. Similarly, in ISO 25000 the *pace tolerance* is a property influencing software *efficiency*. However, these are minor differences and we can consider IFIP properties complementary with ISO 25000. Both software reliability and efficiency influence user-experience and thus also influence usability.

Efficiency refers to the time behavior and resource usage of software. Interactive applications developed in this study are allowed to use all available resources. However, the applications are composed of a number of services that can be aggregated and that work simultaneously. In consequence it is important to ensure that services use only limited resources. This requirement is difficult to enforce in vision-based systems. For image processing-based services, an evaluation of resource usage is necessary.

Time behavior is evaluated in terms of latency. It is measured as the time between the user action and the reaction of the application. Interactive systems always place a constraint on latency of input devices or input systems.

In [6] Bérard analyzes latency requirements for tightly coupled interaction. Tightly coupled interaction is when the user action is directly influenced by the system response. The interaction is modeled as a loop in which both the user and the system observe and react based on each others actions (see figure 3.4). According to Bérard,

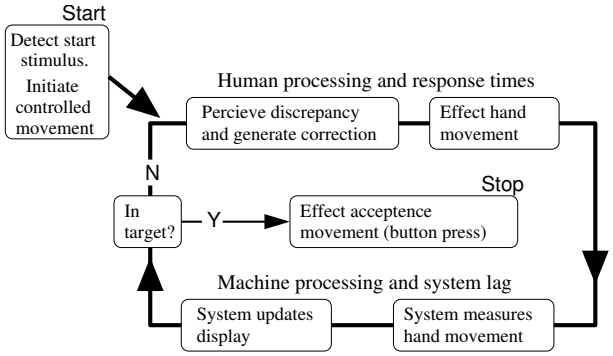

Figure 3.4: Tightly coupled interaction model for target acquisition task (after [100])

the system latency for tasks such as dragging or target acquisition, must be under 50 ms to optimize usability. He also notes that, loosely coupled interaction is much less demanding with respect to time behavior. For instance, a latency of 500 ms or more may be acceptable for a system monitoring the number of people present in a room.

The estimation of the total latency of an interactive system is an important step for performance evaluation. Measurements can be used later-on for anticipating and correcting the latency-induced errors. Additionally, an analysis of the time lags introduced by each component of the system can suggest improvements to overall design.

Maintainability and Portability. Maintainability is the extent to which software is easy to analyze, test and change. Portability describes the ability of software to be adapted for installation on different execution environments. Due to the experimental-prototype nature of our interactive applications these software evaluation criteria do not apply to our development.

3.3 Summary

Interactive applications based on non-conventional interaction modalities set additional challenges for developments; integration of interaction events from different sources, complexity of software and hardware setup, unequal timing-characteristics of used sensors, to name a few. These issues, together with the inherent need to identify elements critical for application performance, call for both clear software architecture and requirements identification.

Based on the W3C definition of service-oriented architecture and web-services, we define services for interactive systems. Their primary attributes detailed in this chapter are: abstraction, isolation and contract. Services are isolated, autonomous software that interoperate by exchanging messages using the BIP protocol. The implementation of the BIP protocol allows services to be advertised and discovered using DNS-SD.

We base the evaluation of services and interaction devices on the ISO 25000 standard, which specifies the quality of software in terms of: functionality, reliability, usability, efficiency, maintainability and portability. From these criteria, we derive non-functional requirements that apply to both software and interaction devices. We suggest evaluating software against qualitative requirements, such as autonomy or absence of tedious parameter setups, and recoverability. Interaction devices should meet quantitative criteria such as: resolution, static stability and latency. We define interaction devices as a compound of sensing or actuation hardware and their software abstraction, i.e. software services.

Having defined the approach to software development and evaluation, we can proceed with the design and implementation of software and hardware necessary to explore interactions based on the steerability and portability of the interface, as described in the next chapter.

Fourth Chapter

Experimental test-bed

This chapter describes tools and techniques developed in order to allow user-studies with a steerable interface. The design of our interactive space test-bed was one of the milestones of this study. Choices made during this phase defined the context of our work, and strongly influenced the nature of problems that we confronted. It is therefore important to understand our motivation for decisions we made.

In the next section, we briefly present different technologies that can be used to implement a steerable graphical interface, and motivate our choice of using steerable projection. We compare the two alternative approaches for creating steerable projection; mirror-based beam control, and a motorized assembly solution. We also present both the advantages of projected graphics and issues related to computer vision based interaction with projected images.

In section 4.2, we describe our Steerable Camera Projector (SCP) pair. The design of the SCP involved both the design of the electro-mechanical assembly and the software to control the SCP movements. The SCP software controller is designed to work as closed loop system, where feedback is provided by a vision based tracking system. We refer the characteristics of the SCP to usability properties described in section 3.2.

In section 4.3, we present a new display device created by combining steerable projection and computer vision based tracking of a handheld screen. A vision-based tracking system together with the SCP controller service allows maintaining the projected image on a cardboard-made Portable Display Surface (PDS). We present both the implementation of the PDS, and the evaluation of its performance.

4.1 Display devices for ubiquitous computing

Currently, display devices are available in four basic flavors:

1. front-projecting systems, such as LCD, DLP or LED video projectors
2. rear-projecting systems, including both CRT monitors and systems using video projectors

Display	Mobility of the device	Occlusion Problems	Space Requirements	Interactivity	Power Requirements
Rear-projection	Fixed	No	Large area, behind display	With separate hardware	High, mains
Front-projection	Mobile, ad-hoc creation	Yes	Large area, in front of display	With separate hardware	High, mains
Steerable camera-projector systems	Fixed but steerable image	Yes	Small area for projector, requires clear view of multiple display surfaces	With system camera, direct interaction with image	High, mains
Mobile projectors	Portable	No	Small handheld, needs light colored surface for projection	Potentially with gestures, movement, or direct interaction of image	Low, battery
E-Paper	Portable	No	Small handheld, flexible displays rollable for storage	Potentially write-on surfaces	Low, battery

Table 4.1: Comparison of Display Technologies for Ubiquitous environments (after [57])

3. rigid flat screens based on LCD, TFT or plasma technology

4. flexible flat screens - electronic paper

An analysis of their suitability to create ubiquitous displays can be found in [57]. In their paper, Molyneaux and Kortuem discuss issues related to each display technology and identify some of research problems that should be addressed in order to create environments enabled with ubiquitous computing. The summary of the identified main characteristics of various display technologies is shown in Table 4.1.

Though not perfect, front projection systems appear to be the most suitable technology for steerable displays. In particular, steerable projectors offer high flexibility in arranging the display position without significantly modifying the environment. What is more, they allow maintenance of the visibility of projected data during movements. For these reasons, we decided to design and develop a steerable video projector.

In the following section, we describe some of the advantages and issues related to the use of projection in interactive systems. In section 4.1.2, we compare two design options for a steerable projection.

4.1.1 Camera-Projector systems

Video projectors offer a number of advantages for AR systems. Projection is an ecological (i.e. non-intrusive) way of modifying the environment. It does not change the

augmented object itself, only its appearance. Moreover, by coupling a video projector with a video camera projected images can be rendered interactive.

Camera-projector systems can be used to supplement the functionality of objects, and henceforth to modify their role in the world. In [67], ordinary artifacts such as walls, shelves, and cups are transformed into informative surfaces. The superimposed projected image enables the user to take advantage of the information provided by the virtual world. The object becomes a physical support for virtual functionalities. An example of enhancing the functionality of objects is presented in [7], where users can interact with both physical and virtual ink on a projection-augmented whiteboard.

Though camera-projector systems are increasingly used in augmented environment systems [64, 73, 93], there is a number of problems inherent to projection. Here, we discuss three main issues: projection cone occlusions, image distortions and limited interaction techniques.

Any type of front-projecting system suffers from *occlusions* of the beam, either by some objects like furniture or, more likely, by users. Objects entering the projection beam cast shadows that break the illusion of surface augmentation. Moreover, if the projected image is observed with a camera, shadows can cause errors in image analysis. Techniques for shadow suppression [17] require at least two well calibrated video projectors. Fortunately, people naturally avoid obstructing projected images, so occlusion is less of a problem when camera and projector share the same point of view.

Image distortions can be twofold: geometrical and radiometric. Some geometrical distortion can be due to the optics of the video projector, but these are rather minor and almost invisible to humans. Extreme changes in the geometry of the displayed images can result from the geometry of the surface at which the image is projected. Unless a surface is planar and perpendicular to the projector beam, the resulting image is distorted.

For planar surfaces, the distortion can be approximated by a 3x3 projective transformation, known as a planar homography. In [88] Sukthankar shows how to automatically estimate the projector-screen homography, and how to use it to pre-warp the projected image to compensate apparent distortions. Yang and Welch [109] show that it is possible to display distortion-free images on non-planar surfaces. It is possible to compensate distortions observed from a given point of view as far as the display surface can be approximated with a continuous function in a reference frame aligned with the beamer's optical axis. However, the resulting beamer-screen calibration is valid only for a single point of view. Therefore, in this study we limit the display to only planar surfaces. Similarly, we ignore radiometric compensation as it requires precise radiometric and geometric camera-projector calibration.

Vision can render projected images interactive. However, vision-based interaction is still a developing area, thus it hardly offers supports for some tasks. For example, text input is difficult to realize on projected interfaces. Currently available projected keyboards like the Canesta Projection Keyboard [77] rely on hardware configuration which limit their use to a given position in the environment. While this issue is

important for the development of projection-based interfaces, it is out of scope of this work. In chapter 5, we discuss in more detail interaction modalities that can be used with projected interfaces.

4.1.2 Steerable beam vs. Steerable Camera-Projector pair

Currently, there are two families of steerable projection system implementations. The SCP, NEC's Active Projector [59] and the more recent Fluid Beam [15], render the image steerable by rotating the video projector together with its associated camera around two perpendicular exes using high-power electric motors. Alternatively, IBM's Everywhere Display (ED) [67] controls the beam direction using an of-the-shelf motorized mirror, see figure 4.1. These two technical solutions for interactive steerable projection each have specific characteristics.

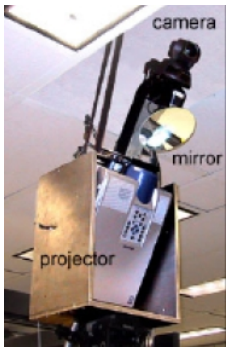

Figure 4.1: The Everywhere Display projector

One of the main reasons that motorized projectors have gained popularity over mirror-based systems is their ability to project images within a higher range of angles. Giving a projector two degrees of freedom allows the SCP to project images within a complete hemisphere. In contrast, the ED's actuation range is limited to a cone restricted by the mirror-projector configuration.

On the other hand, to project all around, the SCP must be mounted centrally in the room. This has two major drawbacks: *(1)* a rotating robotic device hanging from the ceiling is not really consistent with the concept of "disappearing computer", and *(2)* it is rather likely that in a multi-projector setup the centrally-installed beamer will obstruct the projection cones of other beamers. As a result, much care has to be taken when installing video projectors. In contrast, the ED can be discreetly placed in a corner or next to a wall of a room. It can be easily hidden in a wardrobe or behind a double wall by keeping only a window for the beam and the associated camera.

Rotating the whole video projector requires much more power than rotating only a mirror placed in-front of the beam. Higher power consumption results in higher heat

diffusion and more noise. Though in general heating is a drawback, in our office-like setup it is not much of an issue. On the other hand, higher noise level is a handicap in systems that try to produce an illusion of augmenting physical objects with digital images. We do not have any noise measurement data, neither for our SCP system nor for other steerable beamers. The SCP produces noise that for unfamiliar users is undeniably audible and most probably disturbing.

Using a mirror to redirect projection beam allows the use of heavy video projectors. Projectors lighter than 2 kg, such as the Epson EMP-720 used in the SCP, tend to be slightly louder, do not have motorized optics, have lower contrast and produce darker images than heavier video projectors. However, newer motorized projector designs, such as the Fluid Beam presented in [15], allow to operate projectors of up to 7.5 kg.

Both steerable-mirror-based and motorized projectors are usually equipped with a video camera. The camera use is twofold: *(1)* it allows to observe user's actions over the displayed image, thus making the projection interactive, and *(2)* input from the camera can be used to compensate projection distortions. In our SCP the camera is rigidly fixed to the beamer's chassis. Its view-cone overlaps constantly with the projection cone. The ED can observe the projected image using a steerable pan-tilt camera mounted above the steerable mirror, see figure 4.1. This means, that in order to observe the projection, the camera movement has to be coupled with the mirror control.

As the ED's mirror turns around its pan axis, the apparent image also rotates around the projection optical axis. In other words, when an image appears straight on a surface in front of the ED projector, it appears sideways on surfaces at the sides of the ED device, and upside-down on the surface behind it. Since, camera panning does not twist the observed scene in camera images, the ED projection system has not only to consider the transformation between the projector retina frame and the screen frame, but also the transformation to camera retina frame.

To summarize, the choice between steerable beam and steerable projector should be made based on requirements for a particular application. No single characteristic can be considered superior. For this study, greater actuation range and simpler control of the steerable projector were the decisive reasons for our choice.

4.2 The Steerable Camera-Projector

Comparison of different types of display devices lead us to design and build a Steerable Camera-Projector (SCP) platform. The design of the SCP was inspired by [59] and [67]. The SCP is a device that gives a video-projector and its associated camera two mechanical degrees of freedom, pan and tilt. While somewhat bulky (figure 4.2), our device anticipates the current trend of projectors to become portable devices, similar in shape to hand-held torch lamps [72]. The SCP is composed of an XGA-resolution video-projector, a computer controlled pan-tilt platform and a CCD, PAL video cam-

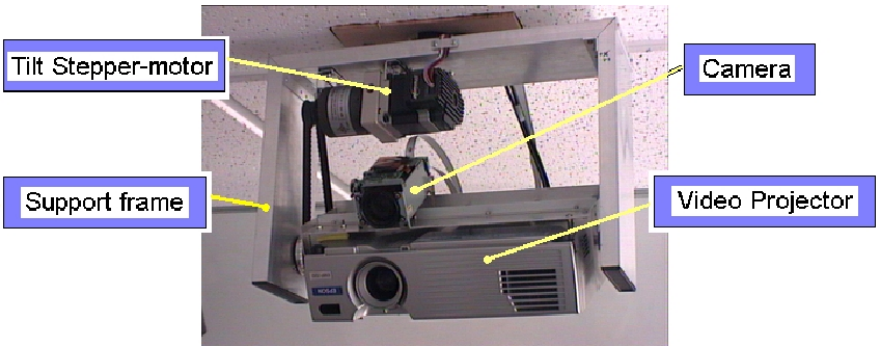

Figure 4.2: The Steerable Camera-Projector pair

era. In order to reuse the communication protocol and hardware interface from the pan-tilt cameras installed in our laboratory, we constructed our system by cannibalizing a widely used commercially available pan-tilt camera. The serial computer interface of the pan-tilt head was connected to high performance stepper motors, enabling rotation of a load of up to 2.5kg. The camera and lens assembly was detached from its pan-tilt base and rigidly fixed to the projector mounting. A mechanical assembly was designed so that the SCP could be mounted on the ceiling in an office.

The mechanical design of the SCP is described in appendix B. Note that the projector-camera pair is mounted in such a way that the projected beam overlaps with the camera view. Association of the camera and projector creates a powerful actuator-sensor pair enabling observation of users' actions within the camera field of view. With the ability to modify the scene using projected light, projector-camera systems can be exploited as sensors for the collection of data for modeling the environment (section 5.2).

4.2.1 Control of the SCP

The SCP is a robotic device allowing to project digital images on surfaces in the environment. Image position in space (i.e. beamer pointing direction) is defined by two variables ψ and θ, representing pan and tilt angles respectively. These variables can be set with the SCP's computer interface.

In order to point the projector at a given point $\mathbf{p} = [x, y, z]^T$ in the scene, the state variables can be calculated as:

$$\psi = \arctan \frac{y}{x}, \theta = \arctan \frac{z}{\sqrt{x^2 + y^2}} \tag{4.1}$$

The SCP can use its camera for object tracking (section 4.3). A typical application would be to track an object using the SCP camera and to project on it images using the SCP beamer. Based on the output from tracking algorithm, the SCP would follow the

object while it moves in the scene. In such case, the control task can be formulated in the following way: find the values of state variables ψ and θ, that would minimize the error criterion. We define the SCP pointing error as the Euclidean distance between the tracked object position and the center of the camera image. The distance is measured in the camera image plane. Position of an object in the camera image is usually calculated as the center of gravity of the object projection on the camera retina.

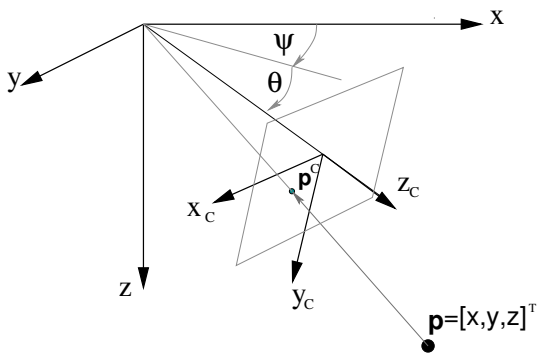

Figure 4.3: World and SCP camera reference frames

We assume a pinhole camera model, with the focal point in the origin of the world reference frame. Image reference frame {C} has its origin in the center of the cameras retina. Its X_c and Y_c axes lie in the image plane and are parallel to the horizontal and vertical image edges respectively. The coordinates of the object image in the camera reference frame are noted as $\mathbf{p}^c = [p_x^c, p_y^c, 0]^T$. The reference frame {C} follows the movements of the SCP chassis such that the Z_c axis is aligned with the pointing direction of the SCP and the X_c axis is always parallel to the XY plane of the world reference frame (Figure 4.3). The world coordinates of an object image can be therefore calculated as:

$$\mathbf{p} = \mathbf{R}_c(\mathbf{T}^c + \mathbf{p}^c) \tag{4.2}$$

where, $\mathbf{T}^c = [0, 0, f]^T$ is the translation vector of the camera reference frame from the retina to focal point, and $\mathbf{R}_c$ is the rotation matrix from camera to world reference frame. The rotation matrix $\mathbf{R}_c$ is a product of three rotations:

$$\mathbf{R}_c = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{4.3}$$

Given image coordinates of an object $\mathbf{p}^c$ and the current SCP parameters ψ and θ, the SCP orientation ψ' and θ', at which the object is in the center of the camera image can be calculated using equations 4.1 and 4.2 as:

$$\begin{aligned} \psi' &= \arctan \frac{p_x^c \cos\psi - p_y^c \sin\theta \sin\psi + f \cos\theta \sin\psi}{-p_x^c \sin\psi - p_y^c \sin\theta \cos\psi + f \cos\theta \cos\psi}, \\ \theta' &= \arctan \frac{p_y^c \cos\theta + f \sin\theta}{\sqrt{f^2 + (p_x^c)^2 + (p_y^c)^2}} \end{aligned} \tag{4.4}$$

Note that equations 4.4 not always provide a unique solution. This means that SCP has singularities in configuration space. A configuration singularity is a situation when for a given end-effector position (i.e. projecting to a given point), there exist more than one possible set of state variables (ψ and θ). In case of the SCP, when projecting vertically ($\theta = 90^\circ$ and $\mathbf{p}^c = [0, 0, 0]^T$ for example), solving equations 4.4 gives an infinite number of solutions for the ψ angle. Therefore, control software for the SCP has to cope with this singularity.

Though the model represented by equations 4.4 provides important insights about the SCP kinematics, it is inconvenient for practical implementation. It has singularities in solution space and relies on the camera intrinsic parameters such as focal length f. Instead, we propose to realize the control task as a closed loop controller, where input is given in pixel coordinates of the target, and the output (ψ and θ) is calculated based on F_{pan} and F_{tilt} transfer functions:

$$\begin{aligned} F_{pan}(\psi, \theta, \mathbf{p}^c) &= \psi + \arctan \frac{p_x^c \cos\psi - r p_y^c \sin\theta \sin\psi + k f_x \cos\theta \sin\psi}{-p_x^c \sin\psi - r p_y^c \sin\theta \cos\psi + k f_x \cos\theta \cos\psi}, \\ F_{tilt}(\theta, \mathbf{p}^c) &= \theta + \arctan \frac{r p_y^c \cos\theta + k f_x \sin\theta}{\sqrt{(k f_x)^2 + (p_x^c)^2 + (r p_y^c)^2}} \end{aligned} \tag{4.5}$$

F_{pan} and F_{tilt} are based on equations 4.4 and a rough approximation of the camera parameters.

While there exist a number of techniques for camera parameters estimation [113], a rough estimate of the camera's focal length can be calculated from its field of view. The SCP's camera horizontal and vertical view angles α and β are about 40° and 30° respectively. Given image width w and height h in pixels, focal length can be calculated as follows:

$$f_x = (w/2) \cdot \cot \alpha/2, \quad f_y = (h/2) \cdot \cot \beta/2 \tag{4.6}$$

Typically, the obtained value of f_x would be different than f_y, which is due to the fact that pixels are rectangular and not square. Thus the calculated f_x and f_y are expressed in different length units.

The pixel's width to height ratio r can be estimated as:

$$r = \frac{f_x}{f_y} \cdot \frac{\tan \alpha/2}{\tan \beta/2} \tag{4.7}$$

Knowing pixel's width to height ratio allows to express all camera-related parameters ($\mathbf{p}^c$ and f) in equations 4.5 in same length units, e.g. the camera pixel-width.

In principle, both the camera focal length and the pixel dimensions ratio can be estimated once for a given camera. Note however, that the focal length of a camera is determined by its optics, and changes as zoom or focus are set. What is more, the estimated focal length is very approximative. Therefore, we add to the equations 4.4 a gain coefficient k, that can be tuned to compensate focal length estimation errors. In order to avoid the overshot, the default k is such that the SCP never reaches the

target position in the first iteration of the control loop. The gain k can also compensate image resolution changes. For instance, when reducing image resolution by image subsampling the controller gain has to be changed by a factor of two.

To avoid the singularity in equations 4.5, the pan axis controller is blocked as the SCP is close to the vertical position. A block-diagram of the SCP controller is showed in figure 4.4. When the tilt angle exceeds 80 degrees, the pan axis controller is disabled. Still, any point within the dead-cone can be reached by the projection, due to the off axis projection of the SCP beamer. In fact, when the SCP panning is blocked the projection cone is approximately vertical. Hence, any image rotation can be realized by the display driver, without moving the SCP.

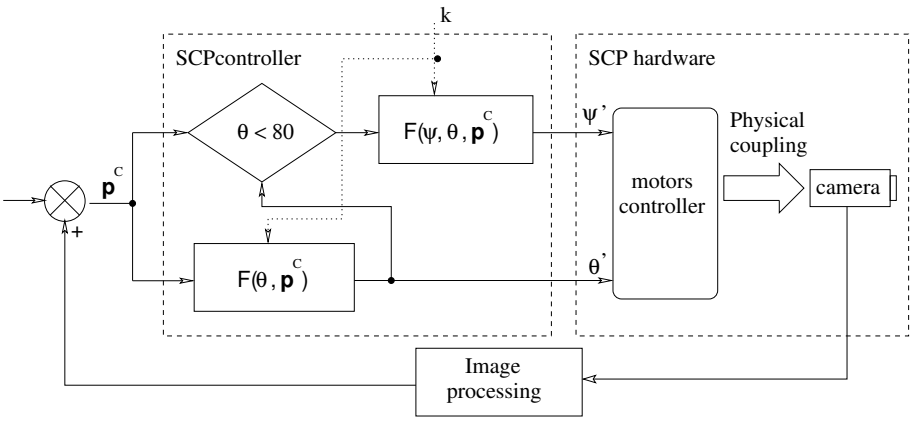

Figure 4.4: Block diagram of the SCP controller

The SCP controller, called *SCPcontroller* is implemented as a software service. It handles the communication between the SCP hardware and other programs. It receives commands via TCP/IP from clients, calculates corresponding change in SCP position, and sends it to the control hardware through the serial port.

To communicate with the SCP hardware, the SCPcontroller uses the VISCA communication protocol used by Sony EVI-D30/D31 cameras. In fact, the SCP was made by connecting powerful industrial stepper motors to the control hardware of an EVI-D30 camera. Sony hardware accepts only absolute position requests expressed with respect to camera's base position. There are 1680 discrete positions in pan and 480 positions in tilt. Considering the SCP movement range, this results in 0.11° and 0.18° pan and tilt resolution respectively.

The SCPcontroller communicates with clients using BIP protocol (see subsection 3.1.3). The command set contains both absolute position orders and relative movement orders considered by the SCP as feedback for the closed loop control. The latter are expressed in pixels and correspond to the target to camera center distance. The controller gain k, can also be set by client applications.

The SCPcontroller can accept connections from several clients. There is no synchronization mechanism between clients, and commands are treated concurrently. A client can lock the SCPcontroller to obtain exclusive control over the SCP. In such case, orders from other clients are refused and the SCPcontroller answers with an error message. Queries, on the other hand, are handled normally.

4.2.2 Performance evaluation

The SCP was designed as an element of an interactive system. It must therefore be evaluated against usability criteria.

The SCP can be decomposed into two subsystems: *(1)* camera together with the projector form the sensor-actuator basis of the SCP, *(2)* the mechanical steerable assembly together with the hardware controller and the SCPcontroller application form the SCP mobile subsystem. In this section, we only discuss the performance of the latter. The performance of the first subsystem is discussed later-on, together with applications that are using it.

Typical user-centric quality measures for interactive systems where defined in section 3.2. Here, we identify requirements that are relevant to the SCP device.

Latency is the interval between stimulation and response. In case of the SCP, latency is the time it takes to position the SCP from the moment it received the order to move. The reaction time is dependent on the distance the SCP has to be rotated. Except of tracking objects, the SCP is used in loosely coupled interaction, so the latency is not an issue. When tracking an object, the latency of the SCP is accumulated together with the latency of the vision-based tracking. In general the latency of the SCP is much lower than that of the tracking process. We estimate that the time between sending an order to the SCPcontroller and the beginning of its execution by the motors is about 30 milliseconds.

Reliability is the extent to which an experiment, test, or measuring procedure yields the same results on repeated trials. In case of the SCP, reliability can be defined in terms of: *(a)* positioning accuracy, *(b)* movement repeatability, and *(c)* image stability. Image position accuracy depends on the movement resolution (see table 4.2), motor positioning accuracy and the backlash of the gears. The SCP positioning error is due to gear backlash, and is about 0.25° for each axis.

A movement is said to be correctly repeated if the final position remains within the typical positioning error. In general, the SCP repeats its movements correctly, however occasionally it happens that a larger error appears. It is most likely to synchronization errors within the SCP hardware.

Image stability or image jitter, are small displacements that can be caused by vibrations of the SCP's structure. Though the SCP vibrates while it is rotated, image

	Pan	Tilt
Rotation range	$\pm 177°$	$+90°$
Angular resolution	$0.11°$	$0.18°$
Angular velocity	$146\frac{deg}{s}$	$80\frac{deg}{s}$
Response time	$\sim 2ms$	$\sim 3ms$

Table 4.2: Rotation platform mechanical performance

stability is not an issue, because it is unlikely that someone would actually want to interact while the projected interface is moving. At standstill the image is stable.

Robustness and autonomy. Robustness is the ability to withstand given conditions. Obviously, extreme conditions can cause the SCP to fail in realizing its task. For instance, the SCP is unable to project images on surfaces at extreme angles, nor on fast moving objects. However these situations exceed the working conditions the SCP was designed for.

Autonomy is the quality or state of being self-governing. Ideally, the SCP would receive and execute orders without any setup and maintenance. It is however necessary to: *(a)* set the SCP to its reference position; this is done manually, *(b)* adjust the gain of the SCP closed loop controller.

Summary The SCP was designed to enable experiments with steerable projected interfaces. It meets most of the performance requirements discussed above. It requires however some setup at startup and occasional position recalibration. The positioning accuracy limits the application of the SCP as sensor. We will discuss this issue in chapter 5.

4.3 Portable Display Surface

In [89] Szalavari and Gervautz introduced the Personal Interaction Panel (PIP); a handheld interaction device allowing users equipped with see-through glasses to manipulate 3D data using a stylus. The PIP couples the displayed data to a wooden tablet. Both the tablet and the user are equipped with magnetic tracking sensors. The interface is superimposed on the image seen by the user wearing a head mounted display (HMD).

In [112] Zhang et al. present the *Visual panel*, a handheld tablet on which users can interact with the system using a finger. Both the tablet and the finger are tracked using a camera. The tablet geometry together with the position of user's finger is mapped to the computer screen allowing the user to use her/his finger like a mouse pointer.

Both PIP and Visual panel are tracked by the computer system and can be therefore used as input for interaction. In this section, we describe our implementation of a Portable Display Surface (PDS) that combines elements of the PIP and Visual panel concepts.

The PDS is designed to allow both interaction with fingers as with the Visual panel, and information visualization on its surface, as with the PIP. The steerable camera-projector pair uses its camera to track a handheld panel, and the beamer to display images on the panel, automatically correcting for 3D translations and rotations as the handheld screen is moved. The SCP camera can be also used to detect when the user places his fingers or hand over a part of the screen in order to select or manipulate projected information. In contrary to PIP, the PDS is actually augmented with projected images, so it can serve as a display for multiple observers. On the other hand, images shown on the PDS are displayed only in 2D.

The PDS can be made from any flat rigid tablet that is small enough to be held in a hand. The screen should be uniform and light-colored. In order to enhance the robustness of the PDS tracking the surface can be delineated by a clearly marked border. An example of a PDS is shown in figure 4.5. It is a piece of white cardboard with black boarders.

Figure 4.5: A hand-held screen used to construct a portable display surface

Compared to conventional display devices, such as PDAs or tablet-computers, the PDS has several advantages:

1. It is ecological, in that it does not require the user to wear any additional markers or beacons.
2. It is lightweight, thus extremely portable.
3. It can be used as an interaction tool. Both the position and orientation of the PDS can be exploited as interaction input.

On the other hand, the use of a steerable projector to augment the surface of a portable screen implies some usage restrictions. In particular, the PDS:

1. has restricted orientation range - it can not operate at extreme angles with respect to the projection beam,

2. suffers from shadows cast on its surface,

3. has restricted movement speed, determined by the maximal speed and acceleration of the SCP assembly.

In the following sections, we detail the implementation of the PDS. First, we review the state of the art in vision-based object tracking (section 4.3.1). Next, in sections 4.3.2 and 4.3.3 we present two tracking algorithms implemented for the PDS application. In section 4.3.4, we explain how the projected image is transformed to fit the surface of the PDS. Finally, we compare the two presented implementations in terms of tracking performance (section 4.3.5).

4.3.1 Tracking of the Portable Display Surface

Monocular object tracking is one of the most common applications for vision-systems. Tracking an object is the task of locating it in the camera image based on its model. The model can be categorized to one of two main classes based on the information type it contains: *(a)* geometrical models [55, 44, 3, 50], *(b)* feature-based models, like color histograms [7] or image-intensity profiles [103, 20].

A tracking application must implement both object detection and re-estimation (i.e. the tracking process). Typically, object detection is used only for initializing tracking. In most tracking systems, the tracking process can be divided into two stages: feature extraction and validation of the observation. The former usually implies the use of a feature model, and is often preceded by image segmentation. The latter consists in determining whether the extracted data match the tracked object (i.e. the model of the object). Both stages of the tracking process can be driven by predictions based on previous observations.

Feature extraction. Features can be simple such us color, luminance and luminance derivatives, or can be more complex based on combinations of simple features. Williams et al. [103] uses normalized image-intensity patches to detect and track faces. Luminance is a simple feature available at almost no cost, but it is seldom used in tracking applications due to its high sensitivity to changes in lighting conditions.

Color is another simple feature that can be easily extracted from images. Color is often used in real-time tracking for interactive systems [7, 106, 45]. However, color is sensitive to illumination changes and tracking applications relying on stationary color-models require controlled lighting conditions. Adaptive color models [108] can be used to minimize this drawback, but they tend to be computationally expensive.

Letessier and Bérard [50] use color-based time derivatives for finger tracking. They perform image differencing in a normalized chromaticity space to extract foreground

objects. Here, authors use normalized color as a means to eliminate shadows from image difference map.

Luminance spatial derivatives are fast to compute, are calculated on grey-level images and are robust to light changes. In [3] Armstrong and Zisserman use a 1D Canny filter [16] to extract points susceptible to lie on a tracked contour edges. In [44] Jurie uses contour images as input for a tracking algorithm, however he does not specify the method used to extract them.

Complex image features such as scale-invariant feature transform (SIFT) are more discriminative compared to simple features, and allow object identification [51]. However, they tend to be computationally expensive and therefore are not well suited for real-time tracking applications.

Observation validation is the process of filtering extracted image features to find subsets that correspond to tracked objects. In [50] extracted foreground objects are compared to a simple geometrical model of a fingertip. Heuristic rules allow eliminating image regions that do not fit the model. Remaining regions are associated to previously detected fingers or declared as new finger appearances.

Armstrong and Zisserman [3] use "random sample consensus" (RANSAC) algorithm [29] to fit contour lines on high contrast points extracted from the image. To cope with errors in line detection, the object geometrical model is successively projected on detected lines with one line removed from the set. For each projection, a distance from the line and its projected correspondence is calculated. If the distance is high, the line is not validated as a correct detection.

The Pfinder tracker [106] models humans as an ensemble of elliptic blobs. Each blob represents the statistical parametric representation of the image position and color of human body, head and limbs. Detection of pixels that belong to the tracked object is based on the likelihood that the pixel belongs to one of the blobs. At each frame, blob parameters are updated.

Williams et al. [103] use Relevance Vector Machines (RVM) [90] to track objects by learning the mapping from thumbnail images of the object to its displacement in the image. The observation is periodically validated using a Support Vector Machine (SVM) [94] trained to recognize the tracked object.

In [44] Jurie presents a probabilistic approach to match a 3D object model to extracted line segments. Lines are said to correspond to object elements if they satisfy a probabilistic error model.

Tracking initialization While many authors neglect the initialization stage of tracking applications, it is determinant for the usability of a tracker. Even the best tracking system will yield bad user-experience without proper initialization.

To initialize tracking, the vision system has to detect, identify and locate the object in the image. Object recognition is a difficult task and is subject of intensive research

effort of its own. Therefore, most tracking applications limit the detection of objects to some constrained conditions. In [106] and in [45] it is assumed that the target corresponds to the largest area segmented in the image. Letessier and Bérard [50] do not recognize fingers, but assume that any finger-like object is a target.

In [103] and in [44] objects are recognized at the cost of violating the real-time computation constraint. Moreover, these applications are limited to the detection and tracking of one target at a time.

In [3] and [55] the problem of initializing tracking is left for further investigation.

Dealing with projected light. Light projected on a tracked object changes its appearance. This can deteriorate the performance of vision-based tracking. Typically, algorithms that rely on color information are susceptible to fail as projected light modifies the apparent colors. Similarly, background subtraction can be error prone as the projection can quickly modify the scene, thus making the background model invalid. However, there are several "tricks" that can help to cope with projection-related problems.

In contrast to the human retina, most cameras have uniform gain for all sensor surface. Therefore, if the dominant source of light is ambient, the projection is hardly visible in the camera image. In other words, if projecting low-contrast images their influence on the camera image is marginal, and can be ignored. This simple approach can be satisfying for certain applications like the Magic Table [7], where the projected image is composed of a relatively small number of thin strokes.

Letessier and Bérard [50] note, that when projecting on white surfaces the image appears brighter than physical objects around it. Therefore, by augmenting the camera gain projection becomes saturated (uniformly white) in the camera image and changes in the projection are less visible to the camera. However, augmenting camera gain disables camera's automatic gain regulation. For this reason, this approach is only feasible for setups with controlled lighting conditions, as the camera in manual mode would not cope with changing conditions.

When projecting on surfaces having only specular reflections, projected light can be filtered-out using polarization filters. This approach is commonly used in stereoscopic projection systems for virtual reality applications. However, as most surfaces of physical objects diffuse light, this method is unadapted for real conditions.

The retina of common cameras is sensible in a wider range of frequencies than the spectrum of visible light. Cameras are especially sensible to low frequency radiation such us infrared light. On contrary, beamers emit only visible light. Projected light can be therefore eliminated from camera images by mounting a low-pass infrared filter on the camera optics. Typically, one would use infrared lamps to lighten the scene, and reflective paper markers put on tracked objects to facilitate image processing [32]. To make the camera more sensitive to infrared light, one can replace the high-pass infrared filter commonly mounted in cameras with a low-pass filter. Such solution

eliminates the need for additional source of infrared light and reflective markers.

When using IR filters, the camera image looses color information as only infrared luminance is captured. In [104] Wilson uses an infrared light source and two low-cost web-cameras modified as described above to make a vision-based back-projection touch screen called TouchLight. Thanks to the spectrum shift, the cameras can see through the TouchLight display surface and detect objects that are close to it.

Thermoscopic cameras can be used to detect humans or they limbs under any lighting conditions. Oka et al. [63] present a real-time robust finger tracker. However, thermal-vision is a bad choice for tracking "cold" objects. What is more, thermoscopic cameras are 10 to 100 times more expensive than standard cameras.

Summary A successful tracking application combines carefully chosen feature extraction, validation of observation, and tracking initialization procedures. For interactive systems, the tracking process must meet HCI-centric performance requirements to ensure a satisfying user-experience. Typically, tracking has to be performed at camera rate with minimal latency. The choice of employed algorithms is a compromise between robustness, generality, reliability and performance.

The PDS is a rigid flat object observed under dynamic lighting conditions. Therefore, we choose to use image spatial derivatives, because they are relatively robust to changes in lighting conditions. An additional advantage of luminance derivatives is that they can be calculated using both standard and infrared cameras. As the object we track has very simple geometry, we implement simple rule-based matching techniques that allow also real-time initialization and recovery. Our two implementation of PDS tracking are presented in more detail in following sections.

4.3.2 Hough space based tracking

This subsection describes our first implementation of the PDS tracking. It is partially inspired by the tracking algorithm of the Visual Panel [112].

Tracking of the PDS is based on the position of the four corners of the hand held screen. However, the corners, themselves may be occluded. Rather than try to directly detect the corners, we use a Hough transform to estimate and track the four screen boundaries. The corners are then determined as the intersection of the four dominant edge-lines subject to constraints on their difference in position and orientation.

To extract edges, we use a Canny filter [16] implementation of Mike Heath, adapted for use with Imalab [52] development environment. Points of high contrast are chained into continuous edges (figure 4.6). The Canny edge detector requires its user to set the following filter parameters: *(a)* the size of the Gaussian smoothing mask, *(b)* edge high threshold, *(c)* edge low threshold. All these parameters are determined experimentally to maximize the PDS tracking performance. We use a 3 pixel width smoothing mask. The edge high threshold, which determines the hysteresis-based chaining algorithm starting points, is set to a value found at about 80% of a histogram

Figure 4.6: Enhanced image of the gradient (thresholded)

of gradient magnitudes of all points in the image. The edge low threshold, which is the stopping point for the chaining algorithm, is set to 10% of the high threshold value.

The edge map is transformed into Hough space to estimate the position of straight line segments. Each edge point "votes" for lines in the Hough space according to the following equation:

$$\rho = x \cos\theta + y \sin\theta \qquad \theta \in [0; \pi] \tag{4.8}$$

Points lying on straight edges produce maxima in the Hough space, represented in figure 4.7 with red crosses.

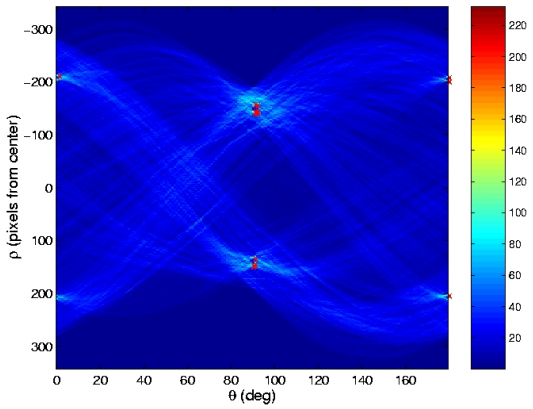

Figure 4.7: Hough space with the fifteen highest local maxima represented by red cross

Hough transforms have relatively large cost in memory and computational time. Nevertheless, lines are only detected in regions of interest and not in the whole image, thus speeding up the computation and reducing the memory requirements.

Initially, the PDS tracking searches the Hough space for two pairs of similar maxima that form a parallelogram. A configuration of such maxima corresponds to four pairwise similar-length and pairwise parallel line segments in the camera image, such

as edges of a rectangular object. This search is limited to rectangles that are roughly aligned with camera image borders and lie in the image center.

Once a rectangle has been clearly detected, the tracking algorithm is started. In contrary to [112], where the Visual Panel is tracked in the camera image, the PDS tracker follows targets as points in the Hough space. The aim is to continuously track 4 local maxima in the Hough space, predict their new locations, re-compute the Hough space for the new image and test among the new local maxima the best four which define the new rectangle location. Segments are tracked independently, giving for each segment one or more possible new locations (figure 4.8). A Kalman filter is applied to the region of interest around each tracked maximum in Hough space and to the region of interest where the PDS was found in the image. The list of possible new

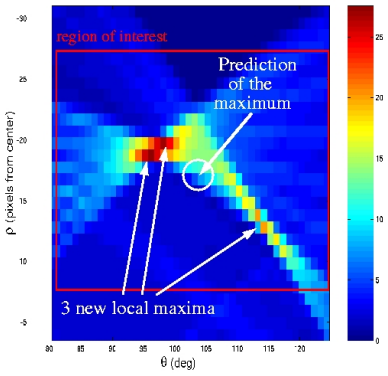

Figure 4.8: For one of the four maxima tracked: list of hypothetic new local maxima compared to its assumed position given by the prediction

locations is sorted by their Euclidean distance in the $\rho\theta$ plane from the predicted new location. Additionally, maxima that differ more than 20% from the tracked maximum are deleted from the list. If there are no candidates for new location of a maximum, we keep the old one.

The result of line tracking selects the four most likely lines defining the new segments position. We test whether the new estimate of the PDS pose is feasible, i.e. we apply constraints on relative movements of tracked PDS borders. The estimated new position of the PDS is validated only if:

1. None or at least two of the PDS borders changed their location.

2. Angles in all PDS corners are within $[20; 160]$ degrees.

The tracking is stopped if the estimation cannot be validated for three consecutive image frames. The PDS tracker falls back to initialization mode described above.

Our voting-based implementation creates a discrete approximation of the Hough space. In order to locate the sub-pixel position of a maximum in Hough space, regions of interest around the new predicted maxima positions are smoothed. Note that in order to speed up the algorithm the Hough space is only computed for angles θ that belong to the local maxima regions of interest.

The tracking algorithm runs as follows:

1. Compute the edge map image for the predicted region of interest.
2. Find maxima in selected regions of the Hough space and for each segment of the previous rectangle location: list and sort the matching lines.
3. Select the 4 new maxima, and apply geometric constraints to the possible lines adjustments.
4. Update new characteristics and predict the new positions of each 4 local maxima in the Hough space defining four new regions of interest in the Hough space.

Tracking of the PDS in the Hough space is naturally robust to partial occlusions of tracked edges. However, the tracking may loose precision and even fail, if edges in the background are aligned with borders of the PDS. In the next subsection, we propose a second alternative implementation of the PDS tracking that is robust to background clutter.

4.3.3 Segment approximation based tracking

Both of our implementations of the PDS tracking are based on edge detection. In our second PDS tracking implementation, the border segments are tracked in the camera image, and their intersections determine the sub-pixel position of the PDS corners. Line segments corresponding to edges are extracted from the image using an adaptation of Canny's [16] edge detection algorithm. Extracted segments are matched to the tracked segments based on their position and mean gradient magnitude.

Gradient magnitude and direction are computed by convolving the camera image with a Sobel operator. The gradient magnitude calculation is followed by a non-maximal suppression. As a result, a raw edge map is obtained. A chaining algorithm is then applied to approximate edges with line segments (figure 4.9).

Since the PDS border is composed of straight lines, we only consider edges that are well approximated with line segments. A chain is started when three adjacent points on an edge have the same gradient direction. This direction determines a rough estimate of the line segment direction. Points are appended to the chain only if their corresponding gradient direction is within a 10 degree tolerance with respect to the estimate. Because Sobel-based gradient direction is a noisy measure, gaps of up to three pixels are tolerated in a chain. Gaps can be also due to missed edge points.

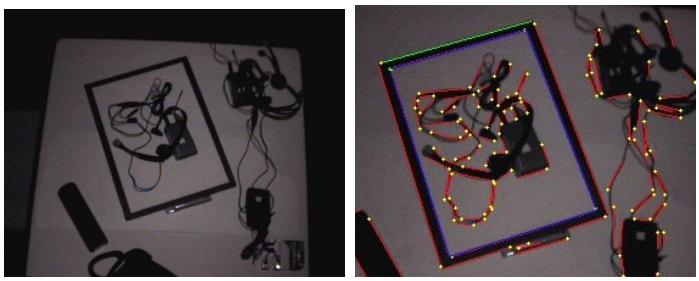

Figure 4.9: Edges approximated with line segments (images taken in infrared light)

As edge points are appended to the chain, first and second moments of the chain are calculated. A mean gradient magnitude of a chain is also computed and memorized. A principal component analysis is performed to estimate the standard deviations σ_1 and σ_2 of the points set along its principal axes, which in turn allows to evaluate the linearity and the mean direction of the chain. A set of adjacent points is considered to be a line segments if the following is true:

$$\frac{\sigma_2}{\sigma_1} \cdot pointsNo < 1.5 \quad or \quad \sigma_2 < 0.8 \tag{4.9}$$

The numerical values in equations 4.9 are set arbitrary.

In order to facilitate the setup and use of our line segments extraction implementation, all its parameters are either fixed or dynamically auto-adjusted. In order to reduce noise and to eliminate the need for hysteresis in chaining algorithm, small gradient magnitudes are set to zero when Sobel operator is applied. The threshold value is set to two third of the median of gradient magnitudes computed in previous frame.

At each frame, the tracking application searches the image for clearly delineated quadrilaterals. The list of extracted line segments is searched for closed-loop sets of at least four segments. Each set of segments that make a closed contour is then ordered by the segments direction. All segments must be approximatively oriented in exactly four directions. Finally, to form an acceptable quadrangle, angles between consecutive segments must be either close to zero (for segments lying on the same quadrangle side), or have to be between 45 and 135 degrees for segments belonging to adjacent sides of a quadrangle. If all the above conditions are fulfilled, the segment set is removed from the list of extracted line segments and the new PDS candidate is memorized. This candidate is registered as a new object only if it does not match any of the previously detected quadrilaterals.

The shorten list of extracted segments is analyzed to track previously detected quadrilaterals. First, new positions of each side are estimated independently of each other, then geometrical constraints are applied on the new quadrilateral shape to verify its validity.

Each quadrilateral is modeled as a set of four line segments. Sides of a quadrilateral can be sectioned by occluding objects into several collinear segments. For the sake of implementation simplicity, only the longest of these segments is memorized in the quadrilateral model.

For each segment modeling one side of the quadrilateral, new segment are matched from all line segments having similar slope, based on a weighted sum of appearance and position similarity measures, where the priority is given to the position similarity.

Once segment candidates are found, the same geometrical constraints are applied like in our first implementation. The estimated new position of a quadrilateral is validated only if:

1. Either none or at least two of the borders changed their location.

2. Angles in all quadrilateral corners are within $[20; 160]$ degrees.

A quadrilateral is removed from the list of tracked objects if the estimation cannot be validated for three consecutive image frames.

4.3.4 Projection on the PDS

Visual tracking of the PDS allows to steer the SCP so that the handheld screen remains in the projection cone of the SCP. However, unless the projected image is deformed before being displayed it would not appear on the PDS correctly. A mapping from the projector retina to the PDS has to be found.

Sukthankar et. al [88] exploit camera screen detection and homographic image pre-warping to rectify projected images in a static setup. Here, we show an implementation of dynamic image pre-warping. The relation between a pixel of the projector image in homogeneous coordinates $[x_1 w, y_1 w, w]^T$ and its corresponding point on the PDS $[u_1, v_1, 1]^T$ is a projective transform with eight degrees of freedom. This transformation may be expressed in homogeneous coordinates as a 3 by 3 projective transformation, H. The nine coefficients of H can be exactly determined by the observed position of 4 points, combined with the constraint that $|\overrightarrow{H}| = 1$. These four points must be non-collinear. That is, no subset of three points may lie on the same line in the observed image. In this case, H is a homographic projection that preserves the mapping from one plane to another, and is thus invertible. The four corners of the tracked PDS provide exactly such set of four points. What is more, by defining the coordinate frame of the PDS such that its corners in homogeneous coordinates are $P1_{pds}[0, 0, 1]^T$, $P2_{pds}[1, 0, 1]^T$, $P3_{pds}[0, 1, 1]^T$, $P4_{pds}[1, 1, 1]^T$, we can express the nine coefficients of the PDS to camera homography H^c_{pds} as analytic expressions of the PDS corners coordi-

nates observed in the camera image $P1_c[p1_{cx}, p1_{cy}, 1]^T$, ... , $P4_c[p4_{cx}, p4_{cy}, 1]^T$:

$$
\begin{aligned}
h_1 &= p2_{cx} \cdot (h_7 + 1) - p1_{cx} & & \\
h_2 &= p2_{cx} \cdot (h_8 + 1) - p1_{cx} & a &= p2_{cy} + p3_{cy} - p1_{cy} - p4_{cy} \\
h_3 &= p1_{cx} & b &= p2_{cx} + p3_{cx} - p1_{cx} - p4_{cx} \\
h_4 &= p2_{cy} \cdot (h_7 + 1) - p1_{cy} & c &= p4_{cx} - p3_{cx} \\
h_5 &= p3_{cy} \cdot (h_8 + 1) - p1_{cy} & d &= p4_{cx} - p2_{cx} \\
h_6 &= p1_{cy} & e &= p4_{cy} - p3_{cy} \\
h_7 &= (a \cdot c - b \cdot e)/(f \cdot c - d \cdot e) & f &= p4_{cy} - p2_{cy} \\
h_8 &= (a \cdot d - b \cdot f)/(d \cdot e - f \cdot c) & & \\
h_9 &= 1 & &
\end{aligned}
\tag{4.10}
$$

Observing the four corner points of the display screen provides the homographic transformation, H^c_{pds} from the screen to the camera. However, to project on the PDS, we need to know the homography from the screen to the projector H^p_{pds} (figure 4.10). The H^p_{pds} can be calculates as:

$$H^p_{pds} = H^p_c H^c_{pds} \tag{4.11}$$

Since the camera is rigidly mounted on the projector, the transformation H^p_c is roughly constant and can be calibrated during system setup. Thus, the observed homography to the camera can be converted to the required homography from the screen to the projector at runtime. Due to this simplification misalignment errors may appear when the PDS is moved away or closer to the SCP assembly. These errors are visually acceptable, but when more precise mapping is required the homography between camera and projector H^p_c should be updated on the fly. In practice, when the camera is used to render the PDS interactive, we estimate the H^p_c transform separately for several positions where the user is likely to interact with the system. These matrices are handled by a separated "Calibration Service" described in section 5.2.4.

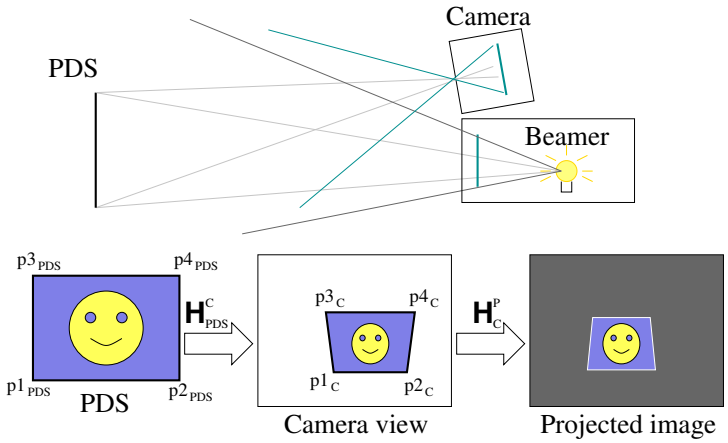

Figure 4.10: Image projection onto the PDS

4.3.5 PDS performance

In previous sections we detailed the algorithms for the PDS application, here we evaluate and discuss its performance with respect to the usability-related requirements defined in section 3.2.

Reliability For the PDS application, the reliability criterion applies to the PDS recognition, tracking and projection alignment.

In all experiments performed with the PDS during this study there was only one PDS operational at a time. What's more the tracking initialization is interactive. In other words the user has to explicitly show the PDS to the system to start tracking. For these reasons, the PDS recognition is not an issue.

The second implementation of the PDS tracking allows the detection and tracking of several display surfaces at a time. For such scenario, the reliability of matching handheld screens in consecutive frames would be determinant for the usability of the system. However, we did not performed such experiments and therefore do not consider this problem.

Since corners are computed with sub-pixel resolution as intersections of tracked edges, it is difficult to evaluate the precision of the PDS border tracking. In fact, manual corner labeling in images is less precise than our tracking implementations, and thus cannot be used as the ground-truth for evaluation. Therefore, we consider that the PDS tracking precision is visually satisfactory.

As described in section 4.3.4, we assume that mapping from camera to projector view is constant. This simplification results in projection misalignments when PDS is moved from the position for which the camera-projector mapping was estimated.

To eliminate these errors the camera would have to share the optics with the video projector. Alternatively, one could estimate the PDS distance from the SCP and adjust the camera-projector mapping accordingly. However, this would require either the use of a separate range sensor or a different approach to PDS visual tracking. It is possible to estimate the distance of an object to the camera either by using stereo vision or by using precise object and camera models. We find both approaches cumbersome compared to the gain they offer, and prefer to calibrate the SCP camera to projector mapping in several key-positions and dynamically switch between these mappings as the PDS is moved.

Robustness The robustness of the PDS application is influenced by three main factors: *(1)* lightning conditions, *(2)* clutter in the scene, both in background and on the PDS itself, *(3)* occlusions of the PDS.

Both of our PDS tracking implementations can work with visible or infrared light. When working with the former, the presence of projection on the PDS can lower tracking robustness. Projecting very bright images can result in high contrast changes on the projection border. As our tracking algorithms are based on high contract edge

extraction they can be mislead by the projection and instead of tracking the PDS start to follow the projection. For this reason, we consider the PDS tracking robust only if the projected image is not the dominant source of light.

When treating infrared intensity images, projected light is not visible, and the tracking application is robust to lighting changes as long as the PDS is visible.

Clutter is not much of an issue to our second implementation as long as there are no clutter edges that are similar and close to the PDS edges. Our Hough space based tracking implementation is more sensitive to clutter. Because the tracking is performed in the Hough space, the information about the extracted edge segment position is lost. Each of the images presented in figure 4.11 segment configurations has the same representation in the Hough space, and would initialize tracking. For the same reason, clutter in the background could mislead the PDS tracking. Clutter on the PDS does not influence the tracking as the corresponding peaks in Hough space are smaller than that of the PDS borders.

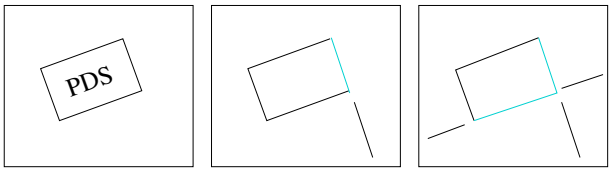

Figure 4.11: Segment configurations having the same representation in Hough space.

Both tracking implementation are robust to partial occlusions of the PDS borders. On the other hand, none of them can track the quadrilateral with one side hidden.

Autonomy Our tracking applications are implemented as services (see section 3.1). They offer client applications a transformation matrix that maps a unit square to the camera coordinates of the PDS. Matrices are broadcasted to all clients each time the PDS moves in the camera image. The PDS service can be paused and resumed by a client, but the tracking cannot be reinitialized on demand. Likewise, clients do not have any control of the image treatment parameters. In this sense, the PDS service is autonomous.

Latency In our experiments with the PDS, we have observed that the projected image follows the PDS with a certain delay. The visible projection error is due to the latency of our system and is proportional to the movement speed. Although, the projection delay does not break the interaction experience, it is disturbing for the user. Moreover, as soon as the user moves the cardboard, the augmentation illusion is destroyed and the user realizes the superposition of the physical screen and the projected image.

The PDS application hardware is composed of the following elements:

1. CCD, PAL-resolution video camera.
2. Video acquisition card.
3. Main processor unit with the process memory.
4. Graphic card supporting OpenGL direct rendering.
5. Video projector.

The total latency of a system is the sum of delays introduced by each of its components.

We estimate the latency of the PDS application with the procedure described below. The experiment setup is presented in figure 4.12. Images of a plastic bar rotating at a

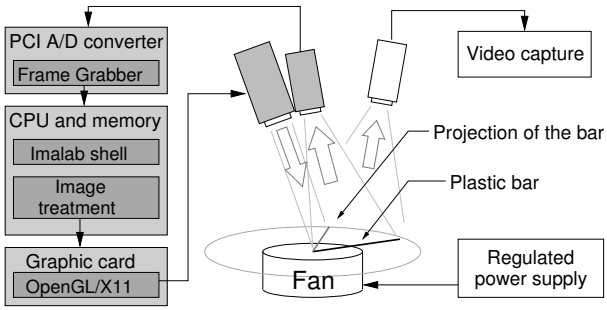

Figure 4.12: Hardware setup for latency evaluation.

constant speed are captured with the SCP camera and re-projected on the tabletop by the SCP's beamer. Both the plastic bar and its projection are captured with a second, independent video camera. The video sequences taken with the second camera are analyzed to estimate the latency.

The delay between image capture and its projection can be estimated based on the motor rotation speed and the angle between the physical bar and its apparent projected image. We estimate the angular velocity of the bar by counting the number of revolutions it makes per a given number of frames of the video sequence.

To acquire sharp images of the rotating bar and thus minimize measurement errors, the capture of each frame should be infinitely short. Therefore, the shutter of the reference camera is set to $\frac{1}{8000}$ of a second. The camera captures test sequences at 30 frames per second. The shutter speed of the camera on the SCP is set to $\frac{1}{5000}s$. The SCP camera provides 25 frames per second.

Such setup allows to evaluate the latency between image capture and image projection. To gain more insights about the delays introduced by all components of the PDS application, we estimate the total latencies of the following chains of system hardware and software components:

1. Camera + video projector (analog image transmit)

Table 4.3: PDS tracking latency estimation

Component chain *No.*	Measured angle *degrees*	Bar rotation speed $\frac{deg}{s}$	Estimated latency *ms*
1	52	3000	17.3 - 17.6
2	72.2	2250	32 - 32.4
3	125	2250	55 - 58
4	7.1	100	70

2. Camera + frame grabber + *Imalab shell [52] + X11 rendering* + video projector
3. Camera + frame grabber + *Imalab shell + image treatment algorithm + X11 rendering* + video projector
4. Camera + frame grabber + *Imalab shell + image treatment algorithm* + OpenGL rendering + video projector (all components)

We noticed, that in all sequences the estimated latency, represented by observed angle between the bar and its projection, vary in a certain range. This is due to the fact that both image capture and display are discrete processes running at different frequencies. As the camera takes an image each 40ms and the beamer refreshes the display each 11ms, the observed latency variation amplitude is 51ms. In order to minimize the influence of the frame-rate-induced lag on the process latency estimation, we take advantage of the projector's retinal persistence and look for images containing the projection of the bar from two consecutive images taken by the camera. By doing so, we are certain that the projected image has been just captured and sent to the projector.

By measuring the minimal angle between the bar and its projection in several chosen images, we can estimate the base latency of a process. Table 4.3 shows the measured angles and the corresponding latency estimations for the first PDS tracking implementation.

The results show that the image treatment algorithm and the OpenGL image display application introduce a lag of about 22 and 12 milliseconds, respectively. This is consistent with our estimates based on CPU cycle consumption. The 17.3ms latency induced by the camera-projector hardware is relatively low compared to the frame-rate-induced latency (up to 51ms). However, when considering that the image treatment and rendering calculations add no more than 38ms to the total latency, and that the hardware-induced delay is about 32ms without mentioning the frame-rate-induced lag, one can argue that our hardware infrastructure is inappropriate for projection-based augmentation. In fact, an obvious bottleneck of our system is the frame-rate of the camera. Moreover, both the camera and the video projector use analogous image transmission, which results in numerous time-consuming analog/digital and

digital/analog conversions. We can expect a considerable latency reduction when we replace the currently used camera with a high frame-rate IEEE 1394-compliant camera.

4.4 Summary of the chapter

In this chapter, we described the components of the interactive environment that allow us to create and experiment with mobile interfaces. Out of the available display technologies, we consider steerable video projection the best adapted for that purpose.

Steerable projectors offer high flexibility in arranging the display position. They are also easy to install without significantly modifying the environment. Projected images can be used to augment surfaces of mundane objects. Combined with the ability to move, projected images can transform an environment in a continuous display. What is more, combined with computer vision, projection becomes interactive.

We designed a Steerable Camera Projector pair, which allows us to display interactive image in a hemisphere around the device. The design included both the electro-mechanical assembly, and the software for controlling the SCP's position. The SCP is composed of the serial interface of a commercially available steerable camera, connected to high-power stepper motors driving a custom-made platform for the projector-camera pair. The software controller is designed as a service allowing clients to position the steerable assembly by sending either absolute position commands, or by coupling the projector to follow objects tracked with the camera. The latter mode of the SCPcontroller is used to maintain a projected image on a cardboard-made Portable Display Surface (PDS).

We present two implementations of the PDS tracking. Our first implementation tracks edges of the PDS as points in a two dimensional Hough space. The algorithm works at camera frame-rate using approximately 60% of the CPU cycles on a 2.8GHz Pentium IV computer. However, in practice the Hough space based representation of the scene is often not enough discriminative, and the tracking is not robust enough for the PDS application. In particular, lines appearing in the background can be easily confused with the borders of the PDS.

Our second implementation of the PDS tracking relies on line segment tracking in the image plane. Contours are filtered-out and approximated based on gradient directions. The PDS is detected as a set of concatenated segments forming a quadrilateral. Each border of the PDS is tracked separately based on similarity measure combining mean gradient of the segment and its position in the image. The algorithm allows tracking at camera rate several quadrilateral objects and is insensitive to background clutter.

The PDS tracking software is also implemented according to the service oriented architecture approach. We also provide a user-centric oriented evaluation of the PDS service performance. The SCP together with the PDS forms the founding for our experiments on mobile projected interfaces.

Fifth Chapter

Interaction modalities for steerable interfaces

In this chapter, we present issues corresponding to the two fundamental components of interactive systems: the input subsystem, i.e. components that allow users to express commands to the computer, and the output subsystem responsible for providing the feedback to the user.

In the first part of this chapter, we present how steerable projection is rendered interactive. We identify input modalities available for projected interfaces and discus how they can be exploited for mobile interfaces. We describe in more detail our implementation of vision-based interactive widgets. Together interactive widgets and the finger tracking of Letessier and Bérard [50] allow us to render our projected mobile interface interactive.

The second part of this chapter concerns issues related to projecting digital images on arbitrary planar surfaces, and integration of interactive applications with our system. We start by presenting a model of the geometrical relation between a display surface and the projector, which allows us to rectify projected images even while they are in motion across a planar surface. We describe our approach to environment modeling and to model data acquisition. Finally, we present services for building interactive applications deployed with the SCP as the main input/output device.

5.1 Input modalities for projected interfaces

In chapter 4.1.1 we discussed problems related to the use of projected light and vision to create graphical interfaces. Issues such as occlusions of the projection beam, or geometric and radiometric image distortions resulting from projecting on uneven surfaces, are inherent to projection systems. On the other hand, projection allows enhancement of ordinary objects with digital images, and projected images can be rendered interactive. As a result, video projectors can be used to augment both the visual aspect and the function of ordinary objects.

In this section, we discuss interaction modalities that are suitable for use as input for interactive projection. We start by describing computer-vision-based finger tracking, as the most flexible input for projected interfaces. Then, we present our implementation of vision-based basic interactive widgets, such as buttons and sliders. Such widgets can be used to build simple projected interfaces. Finally, we briefly discuss the use of interaction modalities other than computer-vision.

5.1.1 Finger tracking

Hand gestures can potentially provide a rich vocabulary of gestures to express commands to a computer system. Notable examples are [37, 96, 36], where employment of commercial motion-capture systems allowed CHI researchers to conceive interaction techniques based on 3D hand and finger gestures. However, motion-capture systems rely on marker tracking and complex setup of hardware. To interact with a system using motion-capture, users must be tagged with markers. Though, 3D gesture recognition allows interesting interaction schemes, in this study we avoid tagging users with markers. While, it is hardly possible to implement 3D gesture recognition using the SCP with its single non-calibrated camera, the SCP setup is suitable for vision-based finger tracking.

Vision-based finger tracking allows users to interact with projected images without wearing markers or actuating any hardware. However, tracking fingertips under projected images is challenging for the image processing algorithms. In case of steerable projected interfaces, the difficulty is even higher because the properties of surfaces on which the GUI can be projected can vary. As a result, most state-of-the-art finger tracking systems are not suitable for our purposes. They require either more complex hardware configuration, or they are unable to cope with the potentially changing and cluttered background. For instance, the Visual touchpad [53, 54] requires a uniform dark background, the PlayAnywhere [105] interface also requires a uniform background, but in the close-infrared domain. Although, the finger tracking system proposed by Oka et al. [63] can reliably track multiple fingers without any constraints on the background, it works only with a thermoscopic camera.

Letessier and Bérard [50] present a relatively robust implementation of monocular finger tracking. The tracking algorithm is based on image differencing and shape filtering. Fingers are found as elongated objects having a circular shaped end, based on a probability map reflecting the similarity of each pixel to the background model. To obtain the similarity map, each image frame is subtracted from the model image (see figure 5.1). This differencing is performed in the color (chroma) domain. The background is continuously updated based on the current image frame and the similarity map, such that regions that differ only slightly from the model are integrated to the background model. Finger tracker is implemented as a service, which outputs a flow of *appear*, *motion* and *disappear* events with corresponding image-space coordinates. The tracker is able to track multiple fingertips at camera frame-rate and with

an average latency of 80 *ms*.

Figure 5.1: Segmentation example. *Left:* Background model. *Center:* Current image, *Right:* Similarity map. (source [50])

The finger tracker of Letessier and Bérard has several advantages over other implementations:

- The implementation requires a simple setup (i.e. only one camera is needed), and is based on off-the-shelf hardware, the results reported in [50] were obtained with 1.4 Ghz PowerPC G4 machine and a middle-class FireWire camera.
- The tracking algorithm can cope with cluttered background and to some extent with changes in lighting conditions. Because image processing is done in the color domain, it is relatively robust to changes in lighting intensity. As a result the tracking system does not "see" shadows that can be cast by fingers hovering over projected interfaces.
- The finger tracker is implemented within a user-centric and service-oriented framework. In fact, our service-oriented framework described in chapter 3.1 is inspired by the work of Letessier and Bérard.

Note that the presented finger tracking implementation is not able to cope with important changes in the background. This is a major drawback, in particular for mobile interfaces. Each time a steerable interface moves, the background model of the finger tracker has to be reinitialized. The initialization of the model consists of capturing an image of the background without any foreground objects, which is very difficult to enforce in real conditions.

We use the finger tracker of Letessier and Bérard to build a drawing application that runs using the SCP hardware on either the PDS or on any other planar surface in the environment. The drawing application described in chapter 6, allows us to test different interaction techniques for passing the control over a portable and a steerable interface between collaborating users.

5.1.2 Simple pattern occlusion detectors based widgets

Steerable projected interfaces provide a very challenging context for vision-based user interface design. The view field of the camera of an SCP overlaps with the projection cone of the SCP's video projector. In consequence, objects observed by the camera are likely to be surrounded by shadows cast in the projection beam. Furthermore, the appearance of objects is modified by the projected light, and the observed scene can change as the SCP assembly moves. As a result, most vision algorithms allowing direct interaction with projected images are not suitable for deployment with the SCP.

In the previous section, we presented a finger tracking system that can accommodate some of the issues related to vision based interaction with projected interfaces. Nevertheless, the presented system still requires reinitializing the background model after each major change in the observed scene. Furthermore, the computational cost of a finger tracker is rather expensive, and unjustified for simple WIMP-like interfaces. An alternative is to process images only locally, around the interactive interface elements (i.e. interactive widgets).

State of the art implementations of visual interactive widgets. Kjeldsen et al. [47] propose a computer vision based implementation of touch buttons and sliders. Each image frame is subtracted from the previous frame, to obtain a similarity map. After simple post-processing for noise reduction, the similarity map is analyzed only in predefined areas around interactive widgets. Fingertips are detected by convolving a fingertip template thumbnail image with the widget area. The detection is limited to one fingertip in a widget area at a time. In the case of multiple detections, all fingertip candidates except the furthest one are discarded. Interaction events are detected based on the analysis of the fingertip trajectory in the widget area. For instance,"a button touch event is defined to occur when a fingertip [...], travels away from the user to a point within the button and then returns toward the user."

The described image processing is a part of an architecture for dynamically reconfigurable vision-based user interfaces (VB-UIs) [46]. In [46] Kjeldsen et al. propose to separate the application from the vision engine. The core application controls the vision engine by sending XML messages specifying the current interface configuration. The interface is composed of interaction widgets defined at the level of functional core by their function and position in the interface. As end-users actuate widgets, the image processing engine sends to the application's functional core messages that correspond to interaction events.

The approach and implementation of Kjeldsen et al. proved to be successful in a number of prototype applications, such as the Traveling Tic-Tac-Toe described in section 2.2.3 or the client guidance system for an interactive retail store [87]. However, the relatively simple image processing has a number of shortcomings. In particular, it relies on an assumption that users reach the interactive widgets from only one side of the interface. This is difficult to enforce when the GUI is projected on a horizontal

surface such as a table. The interface orientation with respect to users is also an additional parameter to set during the system setup.

Ye et al. [110] present a framework for vision-based interactive components design. Both Kjeldsen et al. and Ye et al. propose to analyze camera input only in regions of interest around interactive "zones" of the interface, thus making the vision engine more economic in CPU usage. However Ye et al. go a step further, and instead of applying one vision algorithm per widget type, they propose to use a set of different detection techniques to obtain visual events called Visual Interface Cues (VIC). Each VIC detector called also *selector* examines a small part of the camera image for visual events that can be of different nature, like color, texture, motion or object geometry. Selectors are structured into hierarchies, and each selector can trigger one or more other selectors. Interaction events are detected based on sequences of selectors output. For instance, a touch sensitive button uses only motion detection at first. Once motion is detected, a VIC-based button switches to color-based image segmentation and to segmentation-based gesture recognition.

While the VIC paradigm presented in [110] is appealing for widget-oriented interface design, it lacks the analysis of VIC selectors suitability for interactive systems. Indeed, the VICs framework is presented from the perspective of an expert in computer vision. It does not consider setup and maintenance issues related to each selector such as threshold setting or lighting requirements. In the following paragraphs, we present a VIC-based implementation of basic interactive widgets that was conceived to respect the user-centric requirements presented in section 3.2.

In [12] we presented an appearance-based implementation of touch sensitive projected buttons which we called "Sensitive widgets". The presence of an object over a button on the interaction surface was detected by observing the change of perceived luminance over the button center area with respect to a reference area (see figure 5.2). By defining the reference area around the central-region, the button is made robust to complete occlusion, and sensitive to appearance changes made by oblong objects. The very simple image processing algorithm makes it possible to run several dozen sensitive widgets simultaneously at camera frame rate (PAL-size images at 25 Hz). Moreover, it is robust to lighting changes, thus suitable for front-projection setups.

Implementation and evaluation of several interface prototypes based on sensitive widgets demonstrated that, from the user's perspective, robustness to partial occlusions is also necessary. Indeed, a user pointing at a part of the interface would likely hover her/his arm over a part of a button, thus triggering it. A partial, unsatisfactory solution was obtained by deactivating partly occluded widgets based on the input from widgets placed further away from the user. The idea of combining inputs from several sensitive widgets led us to re-think the touch detection approach.

We choose to assemble atomic occlusion detectors, which are to be placed within and around widgets, in a way allowing to distinguish some simple occlusion patterns.

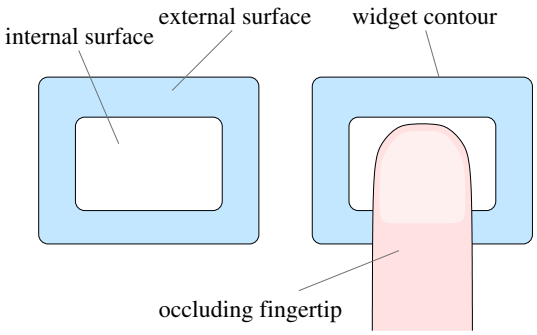

Figure 5.2: Surfaces defined to detect touch-like gestures over a widget

The geometry of the detectors (called "striplets") is simplified to a rectangular strip. These detector federations are used through the SPOD (Simple Pattern Occlusion Detector) service. The SPOD service is internally divided into two separated layers: the image treatment layer called the Striplet Engine (SE), in charge of image processing, and the Vision Events Interpretation Layer (VEIL), in charge of input abstraction. Both of them were designed to meet the requirements described in section 3.2.

Striplets are defined as sensitive patches on the interface. Their response is calculated as the integral of the perceived luminance multiplied by a gain function over the surface of the striplet. The gain function has to be chosen so that its integral over the striplet equals zero. In the current implementation the gain function is a symmetric step function with positive value over the central part of the striplet and negative value at both ends of the striplet (figure 5.3a).

Striplets are designed to detect occlusions by oblong objects. Each striplet is 40 millimeters long and 10 millimeters wide on the projected interface. These dimensions are chosen to ensure a maximal response to occlusions made by finger-sized objects occluding the striplet's central area. Occlusions of any extremity of a striplet are intentionally ignored.

The camera coordinates of a striplet are calculated based on its position in the interface as given by the VEIL service, and the camera-interface mapping. The event trigger threshold, on the other hand, is estimated individually for each striplet by the SE without any control from the client application. It is dynamically set to a positive value, based on the maximal measured response of a striplet during interaction. The only assumption is that fingers contrast with the interface (i.e. fingers are darker than the projection surface), which is true in most setups.

The VEIL is the "brain" of the SPOD service. It *(a)* translates widgets coordinates defined by the client application to a set of striplets coordinates, and *(b)* analyzes the occlusion events generated by the SE and issues interaction events when appropriate.

Currently two types of interactive widgets are implemented: touch buttons and sliders. This allows to build simple WIMP-like user interfaces. The button widget is composed of six assembled striplets: two crossed in the center and four other surrounding the button center (figure 5.3b). Touch events are issued only if occlusion is detected by at least one of the center striplets and no more than one surrounding striplets. Sliders are obtained by simply assembling multiple partially overlapping buttons.

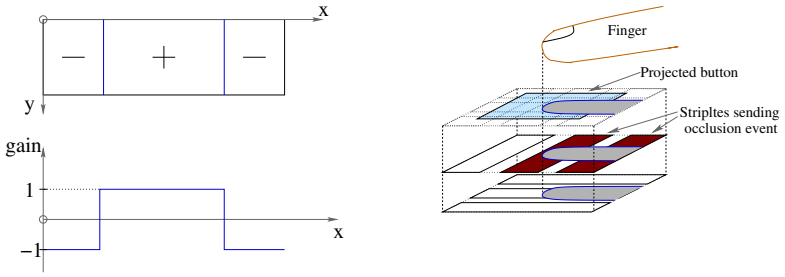

Figure 5.3: *(a)* Striplet gain function. *(b)* Button widget made of 6 striplets.

Since monocular vision systems cannot detect when a finger actually touches the interface, interaction events are generated only after detecting a short pause of the finger over a widget. To detect when a finger touches an interactive widget, one could use disparity information obtained using a second camera. Such technique is presented for vision-based interactive widgets in [28]. However, currently the SCP is equipped with only one camera, so such solution is not feasible with our hardware.

The dwelling period corresponds to the button-down event for mouse-based interaction. To move a slider, the user has to first "press" it and then drag it. This is coherent with existing WIMP interface behavior. In contrast, the dragging task requires 2D movement coordination from the user. If the user's finger exits the SPOD slider area, dragging is stopped.

By assembling striplet detectors in more sophisticated way, it is also possible to develop different types of interactive widgets, for instance crossing-based menus [2]. An extreme would be to cover the whole interface surface with SPOD-button-like structures, thus making a SPOD-based finger tracker. However, SPODs where design for the particular purpose of simulating standard interactive widgets and the resulting finger tracker would be weak compared to other implementations.

Service API. The SPOD service requires the client application to specify the position of each interactive widget in a normalized coordinate frame of the interface. Additionally, the SPOD service needs to know the mapping between the camera view and the interface, as well as a rough estimate of the number of pixels per unit length of the interface in the scene. Both the mapping and the scale are provided by a calibration service (discussed in section 5.2.4). The communication between SPOD service

and calibration service is invisible to the client application. The SPOD service exclusively sends to the client application a stream of interaction events. All communication occurs via TCP/IP connections, using the BIP protocol (as described in section 3.1.3).

Inter-layer API. Both the SE and the VEIL are implemented as independent services, running in independent processes. Their communication also is asynchronous and event-driven, using BIP. For each interface configuration, the VEIL sends to the SE coordinates of all striplets together with the interface-camera mapping matrix. The SE layer sends state-change events that result from user interaction with the system.

Contract for the SPOD service. The initial SPOD service implementation can handle up to 300 striplets at camera frame rate (30 Hz) with images of 320x240 pixels size on a 2.8 GHz Pentium IV processor. In terms of widgets, this means the system can handle roughly 50 SPOD buttons simultaneously. Since actuating buttons is not a form of close coupled interaction, latency is less of an issue. In fact, the VEIL makes the distinction between accidental occlusions and intentional actions based on a dwell time. On the other hand, the SE service is implemented to minimize latency.

Striplets provide only coarse resolution for finger position. The resolution can be enhanced by averaging the position of several striplets detecting the same finger. Our implementation of a slider widget achieves a resolution of about 5 millimeters.

The SPOD service is made autonomous (i.e. except for the UI-camera mapping and scale there are no parameters to set), at the expense of robustness to certain condition changes. In particular, the SE layer would fail to detect occlusion from a finger on a dark background. It would also fail if the image contrast decreases due to a change in camera setting.

While the SPOD service was designed to respect the developer-centric requirements, it does not fully meet user-centric requirements. In particular, the simplistic automatic threshold estimation results in occasional false positive touch detections. However, the SPOD service can be described within the Visual Interface Cues (VICs) framework [110], with local luminance changes as the only visual cue. Using a similar approach for spacio-temporal gesture recognition like in [110], we can hope to alleviate the need of setting an occlusion threshold. Instead the set of striplet responses would be fed to a neural network based VEIL.

5.1.3 Interaction mediated with tools

Finger tracking and SPOD service allow for direct interaction with a GUI. An alternative to direct manipulation is interaction mediated with tools. This type of interaction is ordinary for conventional computers.

The mouse and the keyboard are the most common tools that mediate our interactions with computers. Other tools such as track-balls, track-pads or styluses are

also popular. In fact, compared to tool-based interaction, the use of direct interaction remains marginal.

Most of the conventional interaction tools like mouse and keyboard are not adapted for use with steerable interfaces. Both the keyboard and the mouse have been developed for use with stationary computers, and thus are rather cumbersome for mobile computing.

Nevertheless, small, lightweight devices can offer an interesting alternative to direct interaction with steerable projected interfaces. For instance, the fact that an interaction tool does not operate directly on the display surface, can be an advantage when interacting with a large or distant screen. In section 6.1 we propose to use a laser pointer to control the position of a projected interface.

Vision-based laser pointer tracking is used as input for interactive systems in a number of projects [25, 43, 56]. Detection and tracking the laser pointer dot can be implemented easily. Because laser light has a high intensity, a laser spot is the only visible blob in an image captured with a camera set to low-gain. The laser detection is then obtained by thresholding the intensity image, and determining the barycenter of the connected component. Robustness against false alarms can be achieved by filtering out connected components that have aberrant areas.

As for other tracking systems, the output of a laser tracking is a flow of *appear*, *motion* and *disappear* events with corresponding image-space coordinates. Robustness can be increased by: *(a)* generating *appear* events only after the dot has been consistently detected over several frames (e.g. 5 frames at 30Hz), *(b)* similarly delaying the generation of *disappear* events. Because the camera is set to low gain, varying lighting conditions and shadows are not an issue. Note that modifying the camera gain allows for very simple laser detection, but it can deteriorate other image processing algorithms that run on the same camera.

While direct manipulation and laser-based pointing can substitute the mouse input, efficient text input for projected interfaces remains an unsolved problem. Specialized portable or wearable keyboards could provide a remedy for this issue. In fact, combining portable and steerable interfaces opens a vast field for innovation. For instance, the input interface of a portable computer can be used to access and control a steerable display service embedded in the environment [8].

In chapter 4.3, we introduced the Portable Display Surface, a handheld display screen made of a cardboard. Whereas the principal purpose of the PDS is to serve as a portable display, it can also be used as an interaction tool. Both the location and the orientation of the PDS can be exploited to express commands to the system. In section 6.1 we present how the PDS can be used to move the steerable projected interface from one location to another.

5.1.4 Other interaction modalities

Direct and tool-mediated manipulation are the most conventional interaction modalities. However, steerable interfaces are a particular class of UIs that are likely to profit from more unusual ways of expressing commands. For instance, in the Access spotlight project (see section 2.2.3), users interact with the system by changing position in a room. Although, moving within the Access space has no other goal than just making the computer system follow the user, it shows that steerability of interfaces allows for new forms of interaction.

The use of particular interaction modality depends on the task the interface was designed for, and on the context of use of the system. For instance, in a retail store the position of the user can be used as input for a system to create a personal information display that would follow the user and display information relevant to the store section that the user enters [87]. In such environment, vocal input would be rather unadapted. On the other hand, in a home environment, where location of the user could raise privacy issues, vocal commands could be a good solution.

In section 6.2.2, we present an application for collaborative image drawing in which the interfaces location is controlled using speech detection. Speech is used only to determine the user who leads the conversation at a given moment. Based on this information the system makes the drawing interface available for this person. We do not analyze the content of the speech, but only the timing of vocal contributions of each user.

5.2 Interface rendering

In previous sections, we described different input methods suitable for use with projected interfaces. Here, we address some issues related to the primary output of the system, i.e. to projected images. We present our approach to dynamic image distortion compensation, based on modeling the geometric relation between the video projector and the screen. We discuss how planar surfaces suitable for display can be found and integrated into a sensor-centric model of the environment. Finally, we outline implementation considerations with regard to interface rendering software.

5.2.1 Projector-screen distortion model

Steerable video projectors offer an inexpensive means to transform any convenient surface into an information display. However, unless a surface is planar and perpendicular to the projector beam, the resulting image would be distorted by projection. For planar surfaces, the distortion from projection can be approximated by a 3x3 projective transformation, known as a planar homography. A planar homography is a general class of transformations that includes rotations, translations, scaling, and arbitrary affine transformations. In section 4.3.4 we have shown that a homography can be easily estimated by observing with a camera four points on the display surface. Unfortunately, most of the planar surfaces in the environment are not well delineated, and finding four suitable points is usually impossible. Other methods for finding the homography need to be devised.

There are two alternatives for determining a pre-warp without a clearly delineated display screen: *(a)* manual, interactive alignment of the projection with the display screen, or *(b)* modeling the geometrical relation between the video projector and the screen. The former, though time consuming and suitable only for fixed setups, was successfully used in a number of demonstrators involving steerable projection [67, 87, 68, 8]. The result of such projector-screen calibration is usually expressed as a homography. The advantage of this approach is that the image distortion model is expressed in the image space without the need of explicitly modeling the geometry of the projector-screen system. The principal drawback of such methods is that the image distortion model is valid only locally, for a given projector-screen configuration. Therefore, manual calibration can be used only for fixed display locations. The later alternative, offers more flexibility, but requires modeling the 3D geometry of the projector-screen system.

A number of methods for acquiring a model of the display screen can be found in the literature. Raskar et al. [73] propose to use imperceptible structured light to acquire a depth-map of the screen. They use some of the projectors energy to generate light patterns, and a camera to recover the geometry of the display surface. This method can be used online, while the projector displays graphics. However, it requires a calibrated and synchronized camera-projector pair. An alternative method

based on image feature detection is presented by Yang and Welch in [109]. Yang and Welch also use a calibrated camera, but instead of inserting structured light patterns between projected images, they match image features between the projected graphics and its image captured with the camera.

Both Raskar et al. and Yang and Welch model the display as a 3D mesh. This allows rendering images on screens with complex geometries. However, images rendered on non-continuous display surfaces appear rectified only from a limited number of view points. It is questionable whether such functionality is useful in the context of interactive spaces where multiple users might observe the projected graphics from different locations.

The approaches of both Raskar et al. and Yang and Welch require a significant amount of time to acquire the depth map of the screen. As a result, they can only be used to maintain an already known model of the projector-screen configuration, but are not well suited for online model acquisition. Online solutions should be able to acquire the model instantly, before projecting any image on a new display surface. For these reasons, we decided to limit screen geometries to planer surfaces. As a consequence, we can consider alternative approaches to modeling the projector-display configuration.

We propose to parameterize the pre-warp as a function of the angles between the projector's optical axis and the screen plane normal. In other words, we seek to determine a transformation M from source image X_{si} to a projected image X_{pi}

$$X_{pi} = MX_{si} \tag{5.1}$$

where M is a function of the projector optical parameters and the orientation of the screen with respect to the projector. This transformation should be such that after projection, the projected image appears as if the source image had been projected to a perpendicular, planar screen, up to a scale change S.

$$SP_0X_{si} = PX_{pi} \tag{5.2}$$

We define the origin of the projected image frame R_{pi} and the source image frame R_{si} as the intersection of the projector optical axis with the image. The origin of a world reference frame R_W is in the focal point of the video-projector optics. The x_w axis is aligned with the vertical rotation axis of the projector. The z_w axis points to the planar display surface in the environment, and is parallel to the normal of the surface (figure 5.4). The two angles ψ and θ defining the position of the SCP equal zero when the projector optical axis is aligned with the z_w axis of the world frame. Therefore the projection P of a point in the projected image frame R_{pi} can be decomposed as:

$$X_S = PX_{pi} = P_pCX_{pi} \tag{5.3}$$

Where the matrix C corresponds to a transform from the projected image frame R_{pi}

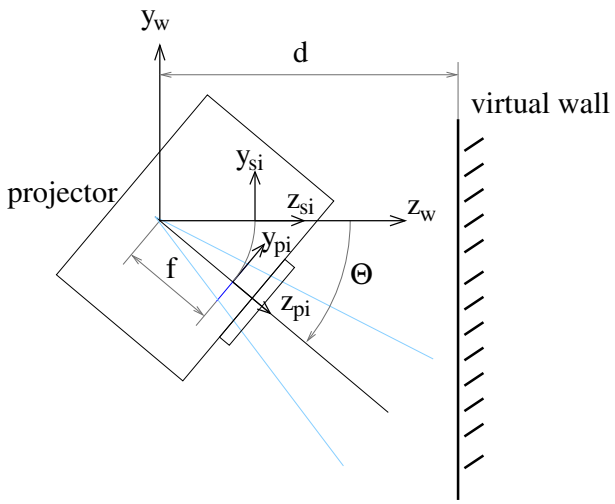

Figure 5.4: Top view of the pinhole projector model and the corresponding reference frames

to the world frame R_W. It has the following form:

$$\begin{bmatrix} x_W \\ y_W \\ z_W \\ w \end{bmatrix} = \begin{bmatrix} & R(\psi,\theta) & & \begin{matrix} 0 \\ 0 \\ f \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{pi} \\ y_{pi} \\ 0 \\ 1 \end{bmatrix} \tag{5.4}$$

Where f is the focal length of the projector optics and R is the projector rotation matrix. The projector projection matrix P_p is composed of the distance d of the screen to the focal center of the projector:

$$P_p = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.5}$$

The unknown distance d can be chosen arbitrary, since this results only in a change of scale. One can say that transformations C and P_p project points from the projected image frame onto a virtual screen at a distance d from the projector focal center. The boundaries of the projection are determined such that the corners of the projected image coincide with the corners of the virtual screen.

Within these boundaries the system can display a rectangle (or any other figure) X_{Sr} that users would like to see on the real screen. The rectangle size can be adapted so that apparent size is independent of the orientation or position of the screen surface by calculating the transformation from the four screen corners.

In order to determine the new coordinates of a rectified image X_{Sr} one can translate the image X_{S0} that would result from a perpendicular projection P_0 by a translation T.

$$\begin{bmatrix} x_{Sr} \\ y_{Sr} \\ z_{Sr} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d\tan\psi \\ 0 & 1 & 0 & d\tan\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{S0} \\ y_{S0} \\ z_{S0} \\ 1 \end{bmatrix} \tag{5.6}$$

Which can also be expressed as:

$$X_{Sr} = TP_0X_{si} \tag{5.7}$$

Finally the desired figure X_{Sr} can be back-projected to the R_{pi} (projected image frame).

$$X_{pi} = T_BP_BR^TX_{Sr} \tag{5.8}$$

Where $P_0 = P_pC$ for $\psi = 0$ and $\theta = 0$. The matrix R^T is the transpose in homogeneous coordinates of the rotation matrix R. Matrixes P_B and T_B assure the back-projection from the scene to X_{pi}.

$$P_B = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} T_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -f \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

Since we know the parameters of the projector and the orientation angles, it is sufficient to apply the mapping M to the desired image shape X_{si}.

$$X_{pi} = T_BP_BR^TTP_0X_{si} = MX_{si} \tag{5.10}$$

If the resulting image X_{pi} exceeds the projector's pixel array, we have to apply an appropriate scaling.

Thanks to this approach we are able to dynamically rectify the image while sweeping the image over a planar surface without having to recalibrate the projector-screen system for each position of the SCP. However to do so, we need to know the location and orientation of the planar display in the environment. In the following section 5.2.2, we discuss methods suitable for planar surface detection and modeling.

5.2.2 Finding the screen

In the previous section 5.2.1, we discussed methods for image rectification when projecting on surfaces that are not perpendicular to the projection beam axis. Here, we focus on the problem of locating potential display surfaces in an environment. Optimally, we would like to have a method able to instantly locate and measure both the geometric and radiometric characteristics of surfaces as they appear or move within

an environment, preferably using only the SCP. Unfortunately, to our knowledge there is no such solution.

Raskar et al. [73] and Yang and Welch [109] propose methods suitable for modeling the display geometry, but both of the methods are time consuming and use calibrated cameras for triangulation purposes. Note, that the SCP camera is mounted vary close to the projector optics and thus cannot be used for disparity maps calculation. In fact, we tried to align the optics of the camera and the projector to minimize errors in the PDS application (see section 4.3). As a result, the SCP is not adapted for stereo-vision-based display modeling algorithms.

In [72] Raskar et al. propose to either use structured light or to tag display surfaces with visual markers. Lee et al. [49] propose to embed photo-sensor-based communicating tags in display surfaces. Similarly to the active tags presented in [71], the tags of Lee et al. determine their position with respect to a video projector based by detecting patterns projected by the beamer. Lee et al. propose to use frequency encoded patterns that are imperceptible to human eyes. Though interesting from the technical point of view, tagging the environments is comparable to manual calibration of each surface.

Tokuda et al. [91] propose to use a laser range finder able to scan its surrounding in a coarse resolution. The range image is then analyzed to locate planar surfaces in the environment, which are then measured with a camera-projector pair using structured light techniques. The range scanner can provide relatively fast updates of the model describing the location of potential display surfaces in the environment, and a rough estimate of the surface geometry.

The approach of Tokuda et al. can be extended with the feature detection based method of Yang and Welch [109], as a means to refine the model while projecting on the new surface. While such combined approach could be used for online acquisition and update of available display surfaces, it requires the use of a laser range finder and a calibrated camera-projector pair mounted sufficiently apart. None of the above hardware is available in our laboratory.

We propose an off-line method for finding display surfaces using cameras installed in our laboratory. During the first phase of our method we scan the environment with the video projector to find planar surfaces and determine their size in the projector orientation angles. In the second step, we evaluate the orientation of detected planar surfaces with respect to the SCP.

A composition of two perspective transformations conserves straight lines. This feature can be exploited to provide a simple method to determine surfaces that are suitably close to planar to be used as a display surface for a rectified image. A grid pattern is projected by the video projector and captured by any of the available video cameras. Using algorithms described in section 4.3 we find the shape of the pattern in the camera image. Any discontinuity of the projected lines indicates a corner-like distortion of the screen. A curved appearance of the lines in the pattern indicates that the surface is not planar.

The system records projector orientation angles corresponding to each detected surface. As a result, a two dimensional model of the environment is created in which each surface has a representation as a region.

Conservation of lines is also used to estimate the orientation of the detected planar surfaces with respect to the SCP. For this, two patterns of three aligned dots are projected and moved across the surface within the sight of a distant camera. The dots lie on one of two lines (a vertical and horizontal with respect to the image borders) crossed in the optical center of the projector optics. Each dot is color coded in order to identify it in each camera image.

A vertical aligned dot pattern is projected when establishing the angle between the screen normal and the optical axis of the projector in the vertical plane. While rotating the projector around its vertical axis, the points are detected in the camera image. Each of the points follows a trajectory which curvature is a function of the projector-screen-camera configuration. A linear regression algorithm is applied for each sequence of colored points seen by the camera and a mean point-to-line distance is calculated. The minimum distance corresponds to a trajectory aligned with the intersection of the screen with a perpendicular plane passing through the focal point of the projector. Thus allows determination of one of the orientation angles at which the video projector is perpendicular to the screen. In a similar manner, the second orientation angle is found using a color-pattern of horizontally aligned dots.

Having found the orientation of the display surfaces with respect to the projector, we can use the projector-screen distortion model described in the previous section 5.2.1 to rectify the image. However, the presented planar surface measurement technique can be applied only to limited projector-screen configurations. In particular, it can be used to find the orientation of large-area surfaces, but fails or provides erroneous estimations when applied on small surfaces. In practice, our method is able to detect surfaces such as walls and the floor. As a result, we prefer to calibrate display surfaces using the PDS (see section 4.3) as a portable calibration grid. Figure 5.5 shows both surfaces detected automatically and calibrated using the PDS.

5.2.3 Environment modeling for interactive spaces

In previous sections (5.2.1 and 5.2.2), we discussed the problem of modeling the display screen geometry. In this section, we present our user-centered approach to environment modeling for interactive systems. We compare our approach to the approaches present in the three steerable interface examples described in section 2.2.3.

We define the steerability of an interface as the ability to relocate the interface in a physical environment (see section 2.2.2). This definition yields that a system supporting interface steerability is bound to the environment and thus must have an internal representation of the environment, i.e. a model of the environment.

When modeling interactive spaces, the level of complexity of the model depends on a number of factors. We identify three characteristics that influence the approach to

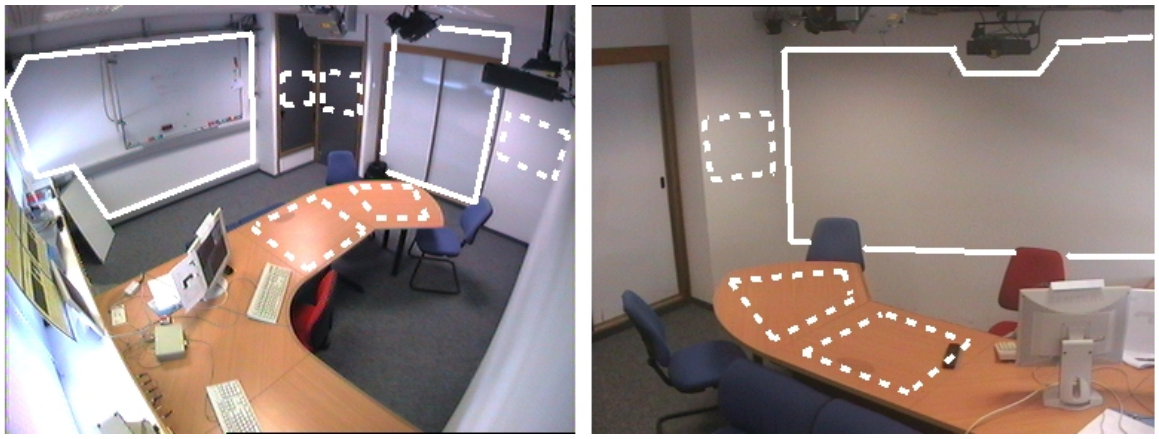

Figure 5.5: Planar surfaces in the environment: surfaces detected automatically are delineated with continuous lines.

environment modeling of systems supporting interface steerability: *(1)* the extent to which both the user and the system need a common understanding of the environment, *(2)* the type of control over the interface location (explicit vs. implicit), and *(3)* the extent to which the computer system is aware of the user. In the following, we illustrate how these factors influence the models of the environment in the example systems presented in section 2.2.3.

Common understanding of the environment Coutaz et al. [41] highlight the duality of interactive systems, where both the user and the system are actors. The dual nature of interactive systems also implies a dual understanding of the perceived physical space. When building an environment model, the system typically generates a sensor-centric environmental representation (see section 5.2.2), but this abstraction is not necessarily comprehensible for the human actor. Therefore, when a common understanding of the environment is needed, the model needs to be translated into a user-understandable representation.

In the EasyLiving project [13], the environment is modeled as a three dimensional Cartesian space, where each entity is described in terms of its geometrical relations to other entities. The representation of the environment in form of transformations between entities is difficult to understand by humans. Therefore, when presented to the user, the model is projected on a 2D map of the EasyLiving space.

In the Traveling Tic-Tac-Toe example [68], there is no need for the system to share with the user a common understanding of the space. Each time a user touches the projected game interface, the system moves the display to another surface stored in the environment model. The model stored by the Traveling Tic-Tac-Toe is simply a list of projecting directions and the data describing corresponding display surfaces. There is

no need to translate the projection coordinates to a representation comprehensive for humans because the interface movement is only triggered by the user and is controlled by the computer system. In fact, for the human actor building a mental representation of the space known to the system is part of the game.

Similarly, in the Access project[1] the system does not translate the sensor-centric environment model to a human-comprehensive representation. Here, the environment is modeled as a plane on which human actors are detected and tracked. The interaction is limited to a direct coupling of the system output with the detected position of the user. Therefore, the user does not have to reason about the interactive space to move the interface.

The lack of the necessity to provide an easy to understand representation of the environment allows the Tic-Tac-Toe and the Access spotlight to maintain fairly simple environment models. On the other hand, in the EasyLiving example the system must support the user in expressing where the interface should move. To do so, the system must translate the sensor-centered model to a human-friendly representation. Such operation usually induces an overhead in the model complexity.

Implicit vs. explicit interface steering The model can also depend on the extent to which the user is involved in controlling the interface location. The control of the location of an interaction resource can be realized through answering three questions: *when?*, *where?* and *how?*. We define interface steering as explicit when the user provides at least the *when* and *where* parameters for the interface control. The steering is implicit when the system infers at least the *where* parameter from the situation context.

In the Access project, the user can control the location of the spotlight beam by walking in the Access space. In other words, the user can specify *where* the interface should move. On the other hand, the user has no control *when* the interaction starts nor *when* it ends, as these parameters are controlled by distant web-interface users. As a contrast, in the Traveling Tic-Tac-Toe, the user can trigger the movement of the interface (specify the *when*), but has no control over the movement execution (the *where* and *how*). Nevertheless, in both examples the interface control can be considered implicit.

The EasyLiving project implements both implicit and explicit interface steering. To explicitly move the interface, the user can choose a display from a list of resources (i.e. list of available displays). The interface can also be steered implicitly to the mouse-keyboard-display closest to the user. To provide the implicit steering the EasyLiving system has to map the position of the user onto the environment model.

In general, implicit control, i.e. when the system infers where to move the interface based on observation of users actions, requires from the system better understanding

[1] `http://www.accessproject.net`

of the environment and the user. On the other hand, explicit interactions require to provide representation of the environment model that can be understood by the user.

User-awareness Implicit control of the interface location requires the system to be aware about the user. The location of the user can be used to determine where to move the interface to remain accessible while the user is moving. Such approach is presented in the EasyLiving project or in the "User-following displays" of Pingali et al. [65]. In both examples the environment and the location of the user is modeled in a 3D space.

The user-awareness does not have to be limited to the position of the user. In section 6.2.2, we present a drawing application where the position of the interface is determined based on both the location of the user and on her/his role within the group of collaborating users.

Implicit control of steerable interface location requires to include the user as an entity in the environment model. Support for explicit interface steering yields modeling the environment in a human-comprehensible manner, for instance as 2D map of the space. These two requirements can be satisfied when modeling the environment geometry as a two or three dimensional space.

Although 3D environment models have many advantages for applications involving the use of steerable interfaces, they are difficult to create and maintain. Instead, we store a model that reflects the available sensor capabilities of the infrastructure installed in our laboratory. The model consists of two layers: *(a)* a labeled 2D map of the environment in the SCP's orientation angles coordinate system, and *(b)* a database containing the acquired characteristics for each detected planar surface. The model can be acquired automatically using a method exploiting the SCP's ability to modify the environment by projecting and controlling images in the scene (section 5.2.2), and can be extended by adding surfaces calibrated manually using the PDS. In practice however, the PDS-calibrated surfaces are stored separately. The surface characteristics stored in the model are twofold; *(a)* data describing the geometrical relation of the surface to the SCP, these are either orientation angles for surfaces detected by the system or homography coefficients for surfaces calibrated using the PDS, *(b)* images of the surfaces and their surroundings highlighted with the SCP. The images can be used to provide users with contextual information which facilitates the mapping between the sensor-centric environment model and the physical world. We use these images to generate interfaces for explicit control of the display location (section 6.1).

While our model is sensor-centric, we take care to provide an appropriate user comprehensible translation. Rather than showing the user a symbolic representation of the world such as a map or a 3D model, we enrich the sensor-centric model with contextual cues that facilitate mapping from an abstract model to the physical environment.

5.2.4 Display and calibration services

In previous sections, we discussed the issue of modeling the display screen and the environment. Here, we present implementation considerations of services abstracting these two models; the Calibration Services (CS). In projection-based interactive systems, the principal clients of calibration services are the vision-based input subsystem and the graphics rendering subsystem. Therefore, we also discus software architecture issues related to projection-based mobile interfaces. In particular, we focus on the design and implementation of the rendering software.

Projected mobile interactive displays can potentially be used as an extension of conventional operating systems. However, projecting images on arbitrary surfaces can result in a distorted image. Neither legacy applications nor graphics drivers of conventional operating systems support graphics pre-warping. In consequence, to compensate projective distortions when displaying legacy applications requires applying a geometric transform on the output of the operating system.

Figure 5.6 presents the software components charged with rendering applications on a display of both UNIX/Linux and Microsoft Windows operating systems. In both

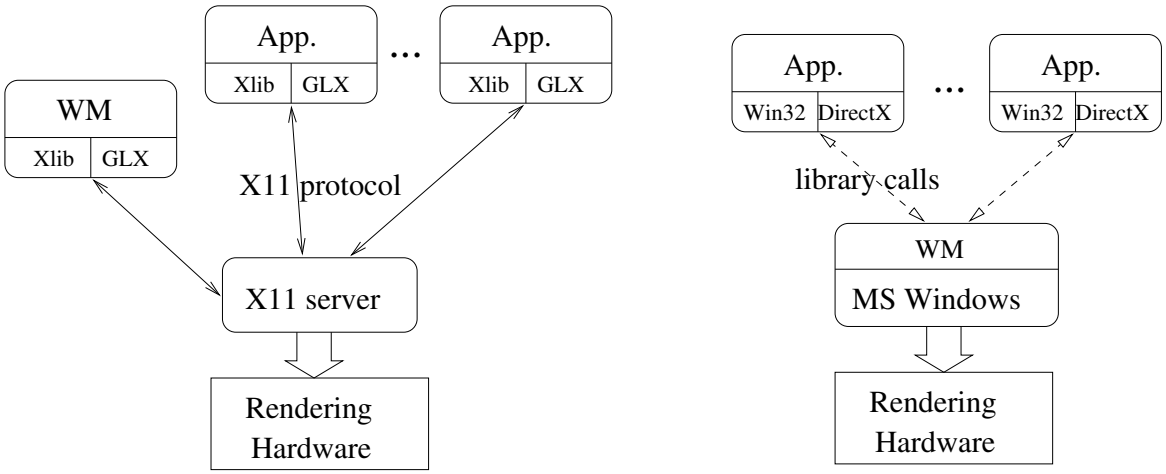

Figure 5.6: General architecture of application rendering in current oprating systems: *Left:* X11-server-based systems (Linu, some of UNIX systems), *Right:* Microsoft Windows

systems it is possible to render the output into a texture, which can then be deformed arbitrarily [18]. However, this solution is very inefficient in terms of resource usage; each texture is first rendered by the GPU of the graphics card, then send back to the main memory just to be send again to the graphics card. Alternatively, one could consider to either write a graphics card driver or to modify the operating system to

deform windows on demand.

In Microsoft Windows, low level graphics drivers are proprietary software with no available documentation. The widow manager of Windows is integrated with the operating system, which makes it difficult to modify. The implementation of pre-warping seems to be easier on Linux operating system where the system output is produced in the X11 server based on the input from applications that use X11 protocol. Unfortunately, X11 servers are large complex software packages, and modifying their code is an ambitious and time-demanding goal.

While pre-warping the output of the operating system allows displaying conventional applications on mobile projected displays, it is not clear whether legacy applications can be adapted for such usage. Current applications were designed for use with the mouse-keyboard-display trio as principal interaction resources. Yet, mice and keyboards are stationary devices and can hardly "follow" a steerable display. Adding mobility to conventional software interfaces may change the way people use them, but validating this thesis requires deploying such systems in natural conditions. The question of legacy applications usability on projected interfaces remains open.

The alternative to system-level pre-warp implementation is to develop custom applications featuring geometric transformation of the GUI. While this solution impedes the use of legacy applications, it offers more possibilities in exploiting interface mobility. An application aware of the applied pre-warp or display parameters can adapt its interface to the given context. What is more, the implementation of a custom application supporting pre-warping is much simpler than hacking the system.

The rendering of applications developed for the SCP is implemented using the OpenGL[2] library. This solution makes it possible to easily take advantage of hardware accelerated rendering. Some of the main features of the OpenGL library graphics model is the support for arbitrary 3D transformations, double buffering, alpha blending and texture mapping. Adding a transformation to the output of an OpenGL-based application is a matter of one or two lines of code. On the other hand, OpenGL offers only basic drawing primitives, such as points, lines, polygons and textured polygons. Therefore, our interfaces are rather simple compared to the eye-candy interfaces of modern applications.

Our first applications such as the Statusman demo[3] were implemented as "monolithic" software. These combined interactive functions, GUI rendering and communication with both a calibration service and the SCPcontroller (see section 4.2.1) in one piece of software. Such an implementation is advantageous for stand-alone low-latency applications. On the other hand, it breaks principal concepts of Service Oriented Architecture; abstraction and isolation of functions.

Figure 5.7 shows a SOA for applications deployed on mobile projected interfaces. The diagram shows the composition of services for an application aware of its location

[2] `http://www.opengl.org/`

[3] Video available on the PRIMA web-page `http://www-prima.inrialpes.fr/`

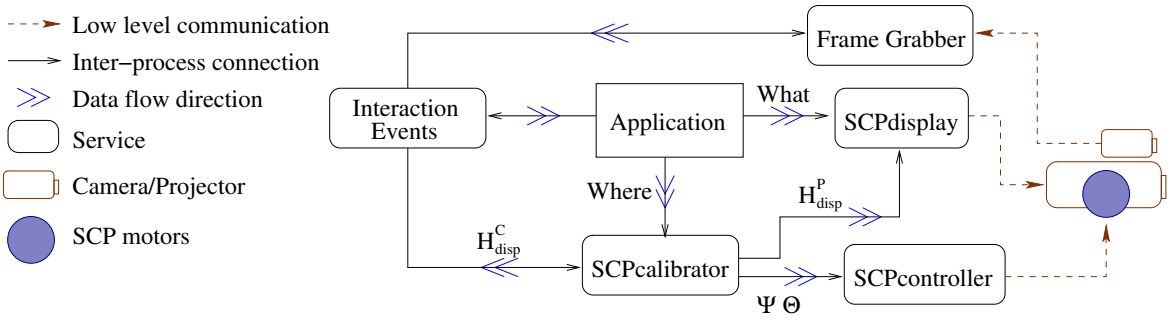

Figure 5.7: General architecture of an SCP-based system supporting interface steerability

in the environment. The application communicates with both the display service (*SCPdisplay*), to specify what to render on the GUI, and with the calibration service (*SCPcalibrator*) to specify where to render the display. The application is also a client of the Interaction Events (IE) service, which in turn is a client of a Frame Grabber (FG) service providing images from the camera. The calibration service sets the pre-warp matrix of the display service and the camera to projector mapping of the IE service.

SCPdisplay, SCPcontroller and FG services can be considered as software drivers for the SCP abstracting its hardware. These three services are bound to the hardware, and thus they cannot be multiplied in the system, i.e. there can be only one SCPdisplay service, one SCPcontroller and one FG per a SCP device. On the contrary, multiple interaction events services can use the same camera image to detect user interactions with the system. Similarly, multiple applications can provide different functions and render their output on the same display through the SCPdisplay service. Note, that only one application at a time should control the location of the display by sending commands to the SCPcalibrator.

The architecture presented in figure 5.7 encapsulates different functions of the system into separated services. Unfortunately, OpenGL does not support display sharing between multiple applications. While there are frameworks such as [39] or [23] that overcome this problem, we prefer to avoid the related software integration and latency overhead. In order to ensure a degree of modularity and code reuse, we propose to implement interactive applications as plug-ins for the SCPdisplay component. The plug-in is a dynamically charged shared library object implementing an OpenGL-based display function. The display function contains all OpenGL calls necessary to render the application GUI. A separate service called *AppLoader* is used to notify the SCPdisplay which shared library to load or unload. The loading order determines the order in which applications are rendered by the SCPdisplay.

In figure 5.7, the interactive application indicates the SCPcalibrator “where” to dis-

play the GUI. The application has to specify at least the direction to which the SCP should point. Because, the SCPcalibrator knows both the planar surfaces saved in the environment model (section 5.2.3), and the optical parameters of the beamer, it can calculate the necessary pre-warp H^p_{disp} for the SCPdisplay service (see section 5.2.1). Similarly, the SCPcalibrator could calculate the display-camera mapping H^c_{disp} necessary for vision-based input services. In practice however, we do not exploit this possibility, i.e. our interactive applications rely on a camera-display mapping stored in the environment model.

The orientation angles defining the SCP display direction are given by the application in radians. The SCPcalibrator transforms them into hardware-specific position units which can be send to the SCPcontroller. If in the SCP's environment model there is no planar surface corresponding to the display direction send by the application, the SCPcalibrator sends a default mapping matrix to the display service. The default matrix applied on the projector output results only in a down-scale of the image without any rectification.

In order to send display direction orders to the SCPcalibrator, the application must know its environment. Because the model known to the application can be different and potentially richer than the one of the SCP, the SCPcontroller accepts optional parameters defining the desired display geometry. These parameters allow the SCPcalibrator to calculate the display mapping matrix H^p_{disp} (and potentially the camera mapping matrix H^c_{disp}) without checking the environment model.

The SCPcalibrator is also needed when displaying images on the PDS. The PDS tracking algorithm described in section 4.3 is encapsulated in a service called *PDStracker*. The PDStracker sends to its clients a homography matrix that transforms a unit square to the camera view of the PDS. This homography is the mapping between the PDS and the camera, i.e. the H^c_{disp} matrix. While the H^c_{disp} matrix can be used by vision-based user-input services, it differs from the display to projector mapping H^p_{disp} needed by the SCPdisplay service.

In section 4.3.4 we show how to transform the H^c_{disp} matrix to the H^p_{disp} matrix using a camera to projector homography H^p_c. The H^p_c matrix is estimated off-line and is valid for a particular display-SCP configuration, i.e. moving the PDS results in projection misalignment. In order to maintain a satisfactory projection precision, we estimate the H^p_c matrix for several key locations of the portable display. These matrices are stored in the environment model used by the SCPcalibrator. As the PDS moves, the SCPcalibrator picks the H^p_c matrix estimated for the closest PDS location and sends it to the SCPdisplay service.

To choose the right camera-projector homography, the SCPcalibrator must know current position of the PDS. While the PDS is moved, repositioning requests are send to the SCPcontroller at camera frame rate. To ensure good reactivity, the SCPcontroller overrides all previous orders. As a result, when tracking the PDS, only the SCP hardware controller knows the orientation of the SCP. Unfortunately, querying the SCP hardware blocks the control of SCP position, thus should not be very frequent.

In current implementation, when the PDS is moved, the SCPcalibrator queries the SCP for current position two times per second.

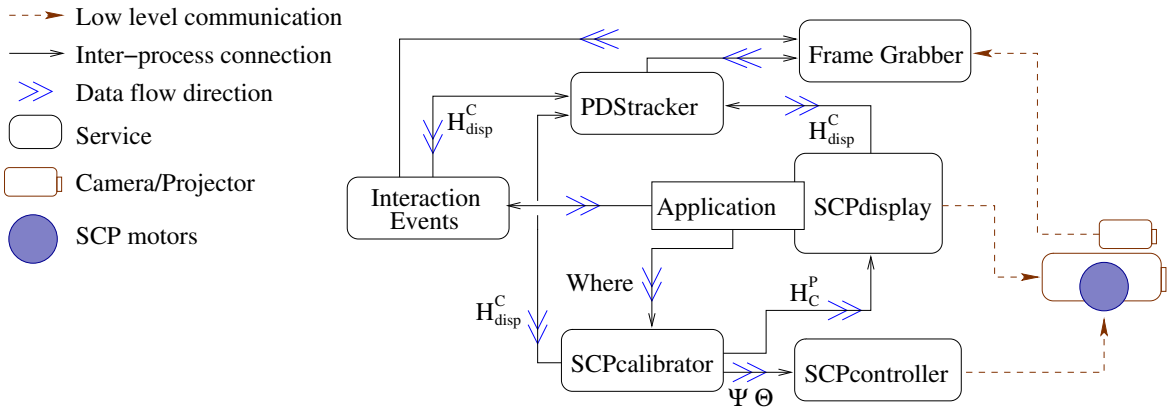

Figure 5.8: Services connections for an application dispalyed on the PDS

Figure 5.8 shows connections between services composing a sample application using the PDS as primary display. Note, that the PDStracker sends the H^c_{disp} matrix directly to the SCPdisplay service, and the H^p_{disp} mapping is calculated by the SCPdisplay. While this solution minimizes the overall latency of the PDS application, it requires both the SCPdisplay and the SCPcalibrator to support two operating modes; one for projection on parameterized surfaces, second for PDS-calibrated surfaces. To switch between these modes, the application sends to the SCPcalibrator either an order to follow the PDS or orientation coordinates for the SCP.

The SCPcalibrator service is also responsible for generating orders for the SCP-controller when tracking the portable display. Based on the H^p_c homography, the SCPcalibrator calculates the position of the PDS center with respect to the camera image center. PDS center coordinates can be then send to the SCPcontroller service as relative positioning orders as described in section 4.2.1.

5.3 Summary of the chapter

In this chapter, we presented two key elements of an interactive system: *(1)* the input subsystem, detecting user actions and providing applications with abstract interaction events, *(2)* the output subsystem, providing the user with feedback about the state of the application.

We present a number of interaction modalities that are suitable for use with mobile projected interfaces. The use of a particular modality depends on the context in which an application is deployed. In this study, we focus on direct interaction with projected images. We present our implementation of vision-based interactive widgets, which

enables us to build interfaces based on buttons and sliders. The interactive widgets together with the finger tracker of Letessier and Bérard form the basis of our interactive applications, which we present in the next chapter.

The use of a steerable video projector as the primary output for our system results in a number of interesting technical challenges. In particular, it was necessary to deal with projective image distortions when displaying on planar surfaces at arbitrary angles. To cope with this issue, we adapt a pinhole projector model and calculate a pre-warp rectifying the projected image. To use this model, we must first find planar surfaces in the environment. We present an off-line automatic surface detection method exploiting the SCP's ability to display patterns. We also discuss our approach to environment modeling, in which we highlight the relation between the type of supported interactions by the system and the necessary features of the environment model.

Finally, we define the services composing interactive applications deployed on the SCP. We discuss certain implementation detail resulting from the basic concepts of the SOA on one hand, and from both software and hardware limitations on the other. The services described in this chapter form the basis for development of interactive applications supporting interface mobility.

In the next chapter we turn from the description of the development of the hardware and software infrastructure to interactive applications made possible to experiment with steerable and portable interfaces.

Sixth Chapter

Interaction with steerable interfaces

In previous chapters, we defined steerable and portable interfaces, and presented the hardware and software developed to experiment with mobile interfaces within a unified framework. In this chapter, we focus on interaction techniques allowing users to control the location of an interface, and on the evaluation of interface mobility as a means to enhance interaction with computers. We also identify the application fields in which steerable interface can enhance user experience with information technology.

In section 6.1, we address the issue of modeling mobile interfaces from the HCI perspective. We propose to use the framework for coupling interaction resources introduced by Barralon et al. [60]. We apply this to a number of interaction techniques for controlling the location of a steerable interface designed in this study. The framework of Barralon et al. helps to structure the interaction techniques for moving a mobile interface and to analyze their usability.

In section 6.2, we discuss possible fields of application for steerable interface. To illustrate the potential of mobile interfaces in the field of Computer Supported Collaborative Work (CSCW), we present a prototype application for collaborative authoring of presentations. The application interface is build using vision-based projected widgets and projected on the PDS.

The prototype authoring application allowed us to test the technology for projection-based interfaces, and helped to prepare a user study for evaluating mobile interfaces in the context of collaborative work. The results of the experiment show, that users prefer to work with a mobile interface than with a fixed one. However, we were unable to determine whether people in general prefer portable or steerable interfaces.

6.1 Spatial control of graphical user interfaces

The key feature of interactive spaces supporting interface steerability is the ability to move the display between multiple physical surfaces. In order to provide such function, adequate interaction techniques for controlling interface position must be devised. In this section, we propose a number of such techniques and present a framework for

analyzing the involved interaction.

6.1.1 Modeling interaction with steerable interfaces

Steerable interface should alleviate the spatial constraints of interacting with desktop computers. The interaction need no longer be bound to a single display mouse and keyboard. However, moving the interface is an additional and potentially difficult task a user must deal with. It is therefore important to avoid making the control of interface position a burden.

Moving a portable interface equals to moving the physical object, e.g. a PDA, to which it is bound. Moving physical objects is a very common activity and requires little cognitive effort. On the contrary, moving a steerable interface involves issuing commands to a computer system that executes the movement. The user is forced to elaborate requests that are comprehensible to the system. The language in which the user communicates with the system is defined by the interaction technique designed for that purpose.

In order to design an easy to apprehend interaction for controlling the location of an interface, it is necessary to understand the task it addresses. In section 2.2.3, we note that steerable interfaces have two basic states; stationary and mobile. The latter is considered as a temporary transition between two stationary states. To move an interface, the user has to order the system to change the state of the interface from standstill to movement, and then define new location where it should fall-back to standstill. These two actions are essential for interface position control.

Coutaz et al. [41] present a framework for reasoning about multi-surface interaction. Their framework describes interactions including multiple surfaces used simultaneously, and does not explicitly address the issue of interface steerability. Nevertheless, Coutaz et al. identify entities involved in interaction, their properties and their relations. The framework is based on the observation of a symmetric relation between the computer system and a user, which they call *artificial actor* and *natural actor*, respectively. Each actor has sensors and actuators to observe and act on interaction resources.

Coutaz et al. define interaction resources as physical entities that play the role of mediators between the artificial and natural actors. They identify two classes of interaction resources: instruments and surfaces. Instruments mediate actions of an actor, and surfaces serve an actor to observe information content.

Coutaz et al. note that interaction resources, such as instruments or surfaces, can be coupled to information content. Considering such a model, the key action of moving a projected interface is equivalent to un-coupling the interaction content from one surface and coupling it to another.

Some steerable interfaces can render information locally in space, and not only on surfaces. For example, in the Access Spotlight project sound can be heard only by the user interacting (followed) with the system. In this case, the information content is

coupled to a location in space rather than to a surface. Therefore, it is more adequate to consider steerability as a process of coupling information to a locus. For instance, a projected image can be shifted by a couple of centimeters on a wall size display. The information content, i.e. the projected UI, is uncoupled and then re-coupled with the same surface but to different locations.

Coupling of interaction resources was recently addressed by Barralon et al. in [60] and in [21]. Though, Barralon et al. focus on coupling of interaction resources defined as physical entities, their analysis can be applied to abstract entities such as information content and location in space. They define coupling as "act of binding two entities so that they can operate together to provide a new set of functions that cannot be provided individually by the entities" [21]. Coupling information content and sensing abilities to a location, such as a tabletop surface or a volume above the table, creates a new set of functions, i.e. the user interface.

In conventional interaction with computers couplings of interaction resources is pre-packaged and immutable. The output of the system is coupled with the display screen, and input to the keyboard and mouse plugged to the workstation. In contrast, in interactive spaces where computing is pervasive couplings change dynamically.

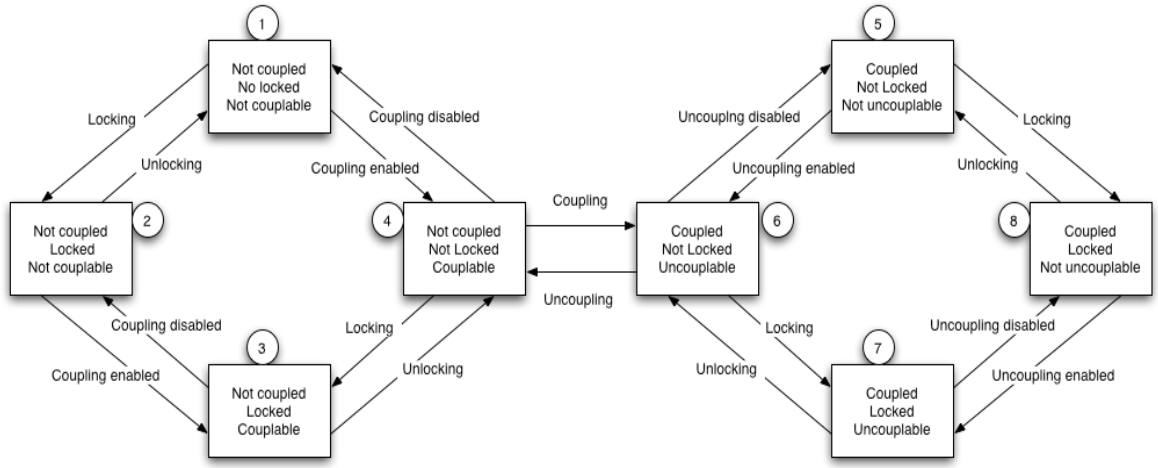

Figure 6.1: Coupling finite state automaton. For sake of readability, transitions betweeb states 1 and 3, 2 and 4, 5 and 7, 6 and 8 are not shown. (source[21])

Barralon et al. denote the coupling of two entities e1 and e2 as (e1, c, e2). **F** denotes the set of functions that result from coupling (e1, c, e2). Barralon et al. propose to model coupling as a finite state automaton shown in figure 6.1. The automaton is composed of two sub-automata whose states are defined by three predicates:

- *Coupled* is true for a pair of entities only if their coupling results in a new non-empty set of functions $\mathbf{F} \neq \emptyset$.

- *Locked* is true if the state of the coupled entity e1 does not permit to modify the state of the coupling (e1, c, e2) from one sub-automaton to the other. In other words, when (e1, c, e2) is locked, the predicate *Coupled* cannot be changed.

- *Couplable* refers to conditions, other than *Coupled* and *Locked*, that determine whether the coupling (e1, c, e2) can happen. For instance, the predicate *Couplable* can express the compatibility of shapes or roles between two entities.

The finite state automaton model of coupling can inform the design of interactive systems. Re-locating an interface is a process of decoupling the steerable interface (hereafter SUI) from the starting location L1 and coupling it to the target location L2. Therefore, moving a SUI is modeled by at least two couplings (SUI,c1,L1) and (SUI,c2,L2). In case of surface-oriented interaction, locations L1 and L2 can be referred as surfaces S1 and S2. Modeling the couplings c1 and c2 as final state automata can help to understand the corresponding interaction process. What is more, each state of the coupling automata can be evaluated against a usability framework such as the IFIP usability properties (see appendix A) which provide a reasoning framework.

In the next section, we present a number of interaction techniques for controlling the position of a steerable projected interface. We analyze them using the finite state automaton model of coupling, and discuss how the coupling model together with usability properties can provide design insights. Our analysis allows us to spot some issues of the coupling model that needed to be refined.

6.1.2 Choosing a location for the interface

Changing the location of an interface is similar to moving a digital object in a virtual environment. Most of the interaction techniques developed for VR applications are based on metaphors inspired from physical actions, such as pointing or grabbing objects with a hand. However, Virtual Environments (VE) allow displaying objects in a continuous volume, thus adapting VR interaction techniques for steerable interface control is not always possible. For instance, when grabbing an object with a virtual hand, the hand can float freely in space and follow a trajectory from the user to the object. In projection-based augmented environments such visualization is not possible, because the display is limited to a set of surfaces scattered in the environment. Nevertheless, VR applications can inspire the design and classification of interaction techniques for display manipulation.

In VR literature, object manipulation techniques are often separated into two categories: exocentric and egocentric [70, 4]. This simple taxonomy reflects the two fundamental reference frames for interaction with objects in immersive environments. With egocentric techniques the user manipulates objects with respect to her or his current position. For instance, both selecting a digital object with a virtual pointer projected as a ray in the virtual environment, and grabbing a physical object with a

hand, are tasks performed from an egocentric point of view. On the contrary, exocentric interactions provide the user with an outside view of the situation. A typical example of exocentric visualization is the World In Miniature (WIM) [82], where the user can move objects in the virtual environment by manipulating proxy objects in a miniaturized model of the environment.

We present three interaction techniques for moving a steerable interface: *(1)* a direct manipulation technique in which a menu lists available interface locations, *(2)* a laser-pointer interaction for choosing display position at a distance, and *(3)* the PDS as a "phicon" for interface re-location. The nature of the interaction techniques varies from exocentric to egocentric.

In a recent study on manipulation of 2D objects in an projection-augmented environment, Voida et al. [98] indicate four issues that should be considered when designing interaction techniques for moving digital objects:

1. Depending on *distance to the object*, users may prefer to touch the interface or point at it. A successful interaction technique should support control from both hand-reach and larger distances. Voida et al. suggest that users expect the distinct gestures used for different distances to blend seamlessly into a "pointing continuum".

2. *User's spatial model of surfaces* describes how users envision the space around them. Voida et al. identify two distinct models: "surface-oriented and continuous". They note that, some manipulation paradigms are "inherently within-surface methods", like "drag and drop", while others like "pick and drop" are "inherently between-surface" methods. While "within-surface" methods can be adapted for manipulating data between surfaces [42], and "between-surface" methods can be used on one surface, the question when to use which interaction style remains open.

3. *Indication of discrete events*, such as object selection or the release of the held object, can be realized differently depending on the tools involved in the interaction. Some instruments such as mice, electronic pens or wand support indication of discrete events by construction, i.e. through embedded buttons. Others, such as user's fingers or hands, require to set a vocabulary of signs defining discrete events.

4. *A user's willingness to move* depends on both the distance to the object, and on the user's propensity to move within space. A user's willingness to move may also depend on the potentially higher accuracy and reliability of interaction up close.

Voida et al. performed their study using the Wizard of Oz technique in which some of the functions of the interactive system are simulated by a human operator. They proposed a number of interaction techniques for manipulation of digital objects in

space based on hand gestures and voice commands. They also asked users to invent interaction techniques for that purpose.

The results of the study show that users prefer hand pointing and touching over voice commands. Direct touching of objects is used when the object is within the subjects' arm reach. Users preferred "pointing gestures using small hand movements and involving minimal body movement" for manipulating objects at larger distances.

Voida et al. identify interaction techniques that are likely to fit user expectations with respect to the control of steerable interfaces. However, the implementation of the evaluated interaction techniques is not feasible given the state of the art in vision-based gesture recognition. We propose a set of prototype implementations of interaction techniques for controlling the location of an interface.

We devise simple graphical interfaces, referred to as *control interfaces*, that allow a user to re-locate an interface. During the design of the control interfaces presented below, we considered the following requirements:

- Minimal footprint. Interface steerability can be added to standard applications as an additional feature allowing people to use them in a more natural way. Hence, the control interface should not interfere with the rest of the interface and disrupt the user primary tasks.

- Maximal control. Users should be able to access the control interface at any time during interaction with the system. The control interface should enable the user to explicitly order the system to switch between the possible states of the steerable interface; standstill and movement. Our interaction techniques allow only explicit control of interface location. Though, they could also be used as initialization for implicit interface steering. For instance, a menu could contain a button to switch the interface to a user-tracking mode.

- Ease of use. Similarly to moving a portable device, controlling the steerable interface should be intuitive and simple for the user. The cognitive load necessary to re-locate an interface should be minimal. The interactions must be easy to learn and simple to execute.

Menu-based interaction In desktop-based interfaces pop-up and scroll-down menus are known for at least twenty years. A menu contains a list of available commands or resources. Since planar surfaces in the environment can be seen as potential resources, it is natural to use a menu as a means for choosing a location for the interface.

The menu-based control interface is added to the display as a semi-transparent button in the lower left corner of the GUI. The button is sensitive to touch-like movements of the user's fingertip. When the user touches the button, a list of available screen locations appears (figure 6.2). The list of surfaces is generated automatically based on the model of the environment. As mentioned in section 5.2.3, each planar surface

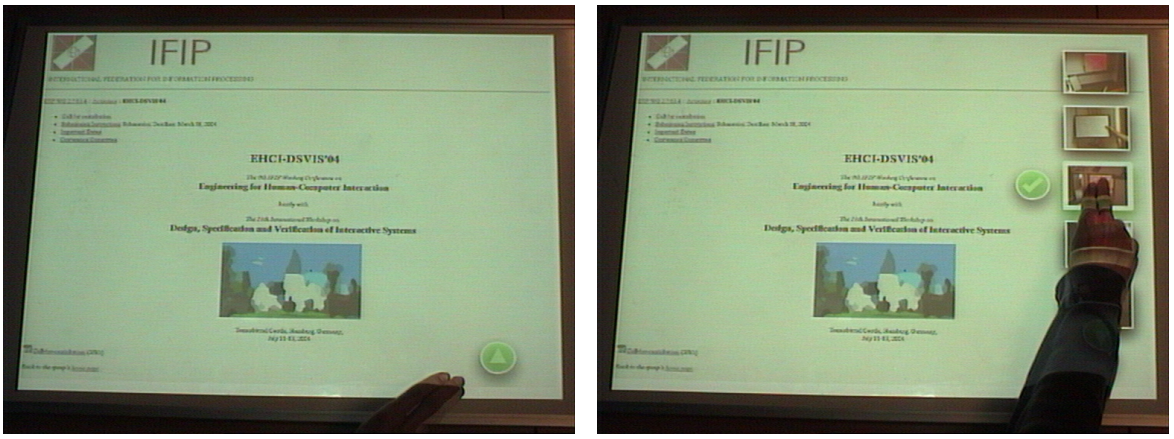

Figure 6.2: Interacting with a list of displays (envisionment)

can be stored in the model together with an image showing the surface as it appears in the environment. Therefore, we display each list item as an image thumbnail.

The images show the available interaction surfaces together with their surroundings, which facilitates the user to map the presented surfaces to their physical prototype. The user can chose a new location for the interface by moving a finger over the corresponding image. In order to avoid accidental selection, we add a "confirm" button. The user can cancel the interaction with the control interface by touching the initialization button again. The list disappears also if there is no interaction for a fixed period of time.

The menu is implemented using the vision-based projected buttons presented in section 5.1.2. Our computer vision system cannot tell whether a finger touches a button or only hovers above the button. Therefore, false positive button touch detections may occur. Unwanted movement of the interface completely breaks interaction with the system, and is thus unacceptable. Therefore, even though this makes the interaction longer, we require the user to validate the selection of new interface location by pressing a confirm button.

The menu-based control of interface location can be categorized as exocentric interaction. Exocentric representation requires the user to map the abstract model to the physical space. The menu shows an overview of the available positions, and the user has to mentally map the items from the list to physical locations.

Mapping images from the menu to surfaces in the environment is a task involving user's visual cognition. To choose the item showing the desired display surface, the user must match the image with his or her view of the surface. If the surrounding of the surface is very discriminant, the task requires little effort. Optimally, the user should be able to map the thumbnail image to the corresponding surface based on a single visual feature. If more features must be analyzed, the mapping task becomes

inefficient. What is more, if images in the menu are similar, the user is forced to carefully compare them before taking the decision of moving the interface. Much care must be taken to provide sufficient number of easy to discriminate landmarks in the images composing the menu.

A more conventional exocentric approach to resources control is presented in the EasyLiving project [13]. The EasyLiving system shows a 2D map of the environment on which the user can choose objects and modify their state. This interface is used for the control of lighting. Choosing a location for an interface on a map involves spatial reasoning. There is a number of factors influencing the cognitive load of understanding maps; number of details, the orientation of the map, indication of the user location, to name a few. In general, the quality of the map depends on the system's ability to model the environment.

While the world in miniature or map-based approach is likely to remain usable with a higher number of available locations, it requires a more complicated model of the environment. Our menu-based interface directly reflects the sensing abilities of the SCP and cameras located in our laboratory.

The menu-based interaction is modeled by two finite state automata, corresponding to the two couplings (SUI,c1,L1) and (SUI,c2,L2). L1 refers to the starting location (i.e. starting surface), and L2 to the target location. The life-cycle of the couplings (figure 6.3) models the interaction process of re-locating the steerable interface from L1 to L2. Each encircled number in figure 6.3 corresponds to a state of the coupling finite state automaton shown in figure 6.1, and is defined over the predicates *Coupled*, *Locked* and *Couplable* (section 6.1.1).

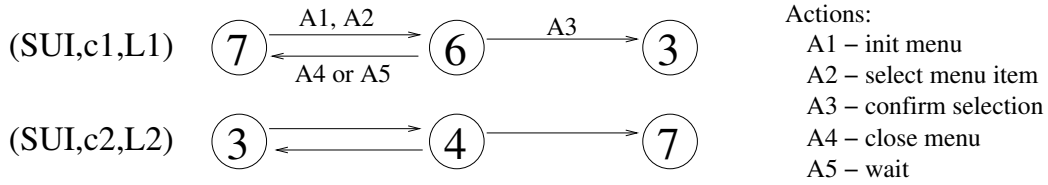

Figure 6.3: Coupling finite state automata for the menu-based control interface

Transitions between states are defined by user actions. Note that, a change of state in coupling c1 results in a state change in coupling c2. Couplings c1 and c2 are indirectly coupled because they share a common entity, i.e. the SUI. Therefore, corresponding transitions in coupling c2 are defined over the same actions as for coupling c1.

At first, coupling (SUI,c1,L1) is in state 7, and coupling (SUI,c2,L2) is in state 3. The SUI is coupled to the location L1 (surface S1), thus it is not coupled to location L2. Because the SUI cannot be coupled to an other surface the predicate *Locked* is true for both couplings c1 and c2. The SUI is also uncouplable from L1 and couplable to L2, because it is projected with the SCP and thus can be moved between surfaces.

When the user initializes the control interface and selects the image corresponding to L2, the couplings c1 and c2 become unlocked. The coupling c1 changes to state 6 and coupling c2 changes to state 4. For both c1 and c2, the predicate *Coupled* can be changed by one gesture of the user, i.e. touching the confirmation button. The couplings are locked again, if the user either touches the menu-close button, or stops interacting for a while.

If the user confirms the selection, the predicate Coupled(SUI,c1,L1) becomes false and inversely the predicate Coupled(SUI,c2,L2) becomes true. Because, the menu-based control interface does not appear after the SUI moved, both couplings are locked, thus c1 is in state 3 and c2 in state 7. The interaction is completed and so is the life-cycle of the couplings.

The menu-based interface allows users to send the interface to a distant location. However, to perform the interaction, the interface must be within user's arm reach. What is more, the user can move the interface only between discrete locations. The interface presented below addresses these issues.

Interacting at a distance Having a large display or multitude of display locations demands methods for interacting at a distance. We propose an interface designed to be initialized and operated with a laser pointer.

Interaction from a distance using a laser pointer has been experimented [25, 43] and evaluated [56] by several authors. Most of them translate the laser-pointer movements to events similar to those generated by a mouse. According to Myers et al. [56], pointing at small objects with a laser is much slower than with standard pointing devices, and less precise compared to physical pointing. However, the resolution of hand pointing detection from a distance [40] is rather low, and stereoscopic vision is required. Standard pointing devices like the mouse or track-ball provide interaction techniques that are suitable for a single screen setup, even if the screen is large, but are difficult to adapt for multiple display environments with complex geometry [42].

In our system, we use laser-based interaction exclusively as a tool to move the steerable interface. Users are free to use their laser pointers in the conventional way. They can point at anything in the room, including the projected images. The system does not respond unless a user makes an explicit sign.

In our application, interaction is activated with a double sequence of switching the laser on and off while pointing to roughly same area on the projected image. If after this sign the laser point appears on the screen and does not move for a short time, the control interface is projected. During the dwell delay of the laser-point, the jitter of the hand is estimated in order to scale the control interface appropriately.

The interface shown in figure 6.4 is a semi-transparent disc with arrows and thumb-nail images. The arrows point to physical locations of the available displays in the environment. Similar to the menu-like control interface, the images placed at the end of each arrow are taken from the environment model. They present each display surface

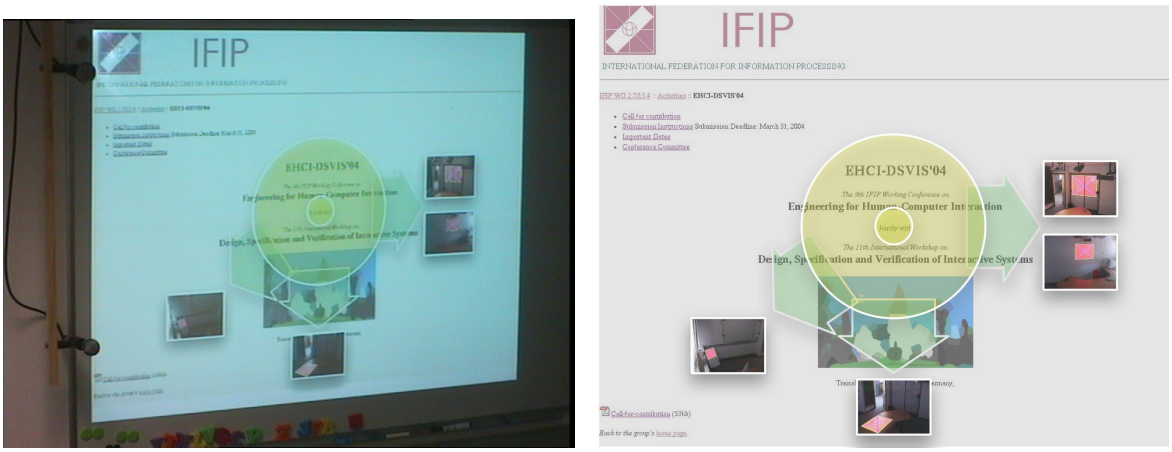

Figure 6.4: Laser-based control interface (envisonment)

as it appears in the scene. The size of the images and the width of arrows is a function of the measured laser point jitter. So is the size of the dead-zone represented by the small internal disc, in which the laser dot can stay without reaction of the system. The control interface is semi-transparent in order to avoid breaking users' interaction with the application, in case of a false initialization.

In order to avoid unwanted system reaction, the control interface is inactive when it appears. To activate the interface, the user has to explicitly enter and dwell for a short time in any of the GUI's elements, i.e. in any arrow, image or disc. As the user moves the laser point within the yellow outer disc, the system moves the interface and follows the laser point with the center of the disc. This movement is limited to the area of the current display surface. Interface movement is slow and allows fine adjustments of the steerable interface position.

When the laser goes outside the yellow disc or enters an arrow, movement halts. The user can then place the laser dot in the image of choice. As the laser point enters an image, the application interface immediately moves across the room to the corresponding surface. The control interface does not appear on the newly chosen display until the user initializes it with the double-blink sign. At any time during the interaction process, the user can cancel the interaction by simply switching off the laser pointer.

Compared to a menu, the laser-based interface is more egocentric. The user can see all available locations for the interface, together with hints for their relative position with respect to the current one. As a result, matching the physical surface with its image becomes much easier, because the user can immediately eliminate most of the images.

The laser-based interface provides more cues to help the user mapping physical locations to the sensor-centric environment model. It also allows tuning the position

of the interface on large surfaces by following the laser dot. What is more, the control interface allows users to interact also when it is out of user reach. On the other hand, our implementation requires the use of a laser pointer as an actuator.

The necessity of a dedicated instrument to control the interface is an important constraint. However, Voida et al. [98] note that 3 out of 9 participants of their study on object manipulation techniques, "defined techniques involving the study conductor's laser pointer". Subjects asked to use a laser pointer, even though they "were told the system would be looking for gestures and/or voice".

Voida et al. explain users' willingness to use a laser pointer through their preference to make small hand movements when pointing. However, other reasons may also contribute. For instance, people might feel uncomfortable when pointing with a hand because they become conscious of the system observing them. Hand pointing might be also disfavored for social reasons. What is more, a laser pointer provides an easy way to indicate discrete events by switching the beam on and off. Finally, pointing with a hand is more ambiguous as the pointing direction can be defined by either the forearm direction or the line passing through user's eyes and the hand [61]. The laser dot highlights the pointed object.

The laser-based interface can be used either to move the SUI to another surface, or to adjust the position of the SUI within a surface. In the former case, the coupling-based interaction model is very similar to the menu-based interaction model. Only the actions to move between states of couplings (SUI,c1,L1) and (SUI,c2,L2) are different (figure 6.5 top). In the later case, the interaction covers 4 states of the couplings automata.

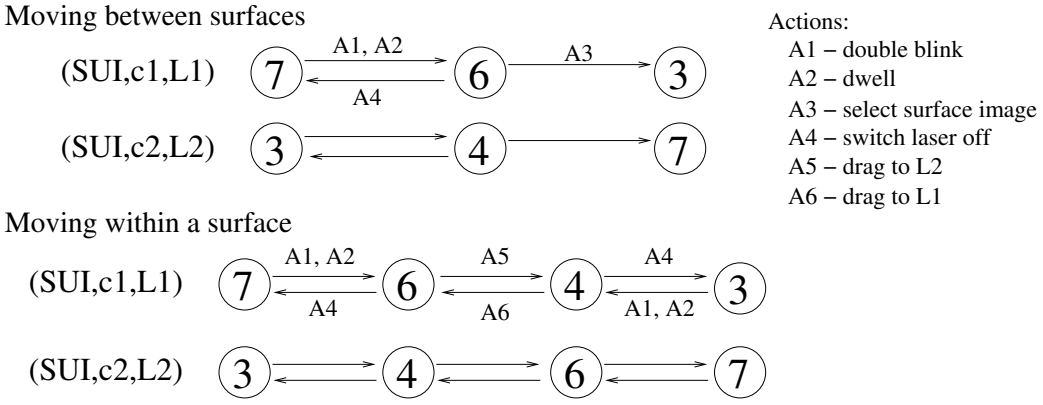

Figure 6.5: Coupling finite state automata for the laser-based control interface. *Top:* moving the SUI between surfaces. *Bottom:* moving the SUI within a surface.

When moving the SUI on a surface from location L1 to L2, the couplings c1 and c2 are at first in state 7 and state 3, respectively. As the user activates the control interface by double-blinking the laser and dwelling it in one of the interface's elements,

the SUI becomes unlocked. Coupling c1 is in state 6, and coupling c2 is in state 4. To move the interface, the user can now point with the laser within the larger disk. The interface follows the laser dot and moves away from location L1, thus the predicate *Coupled* of coupling c1 becomes false. As the interface reaches the location L2, coupling c2 goes to state 6, i.e. the SUI is coupled to L2 but it is not locked. Only when the user turns off the laser pointer, the control interface disappears, and both couplings get locked.

Note, that in this interaction, the couplings life-cycles are symmetrical and reversible.

The laser-based interface makes it possible to position the steerable projected display on any surface known by the system. Yet, the configuration of available surfaces can change and new surfaces can appear in the environment. We propose to use the PDS as a tool allowing to freely control the location of the interface, without the limitation of the environment model.

Interaction using the PDS In section 4.3 we present the PDS, a cardboard-made lightweight display. We can exploit the portability of this device to use it as a tool for explicit control of the display location. The PDS can serve as a temporary vehicle for the steerable interface, like the electronic pen is a means of transportation for data in the "pick and drop" paradigm [74]. Note that, the PDS is perceived by the system as yet another planar surface, thus the mobile interface can remain interactive while projected on the PDS.

To relocate the interface with help of the PDS, the user needs to start the tracking of the cardboard. Note, that one of the images composing the menu-based control interface shows the handheld screen (figure 6.2). When the user chooses the PDS image in the list of available surfaces, the display becomes darker and the PDS tracking algorithm starts to look for the cardboard surface. When the PDS appears over the projected image, the interface is transferred to the PDS. If no rectangular object appears in the SCP camera view within a fixed delay, the system cancels the interaction and returns to its previous state.

The user can move the interface projected on the PDS. To stop the tracking algorithm, the user has only to touch a "freeze" button projected on the PDS. The location of the PDS together with the corresponding pre-warp matrix is thereby added to the environment model as new display surface. This mechanism allows the system to dynamically update the model.

The PDS-based control of interface location is based on portability. The PDS as a portable device offers a very natural way of spatial control. The choice of interface location is not limited by the number of surfaces known by the system, because the user can position the interface anywhere. The model of the environment stored by the computer system is more transparent to the user. As a result, the user is freed from learning the relation between the sensor-centric model and the physical space.

On the other hand, moving the interface requires the user to carry the PDS to the new location.

The PDS-based interaction involves a two stage un-coupling-coupling process. First, the SUI is un-coupled from the starting surface S1 and coupled to the PDS, second, the SUI is un-coupled from PDS and coupled to the target surfaces S2. As a result, the interaction is modeled by three couplings (SUI,c1,L1), (SUI,c2,PDS) and (SUI,c3,L2). The evolution of these couplings during interface re-location is shown in figure 6.6.

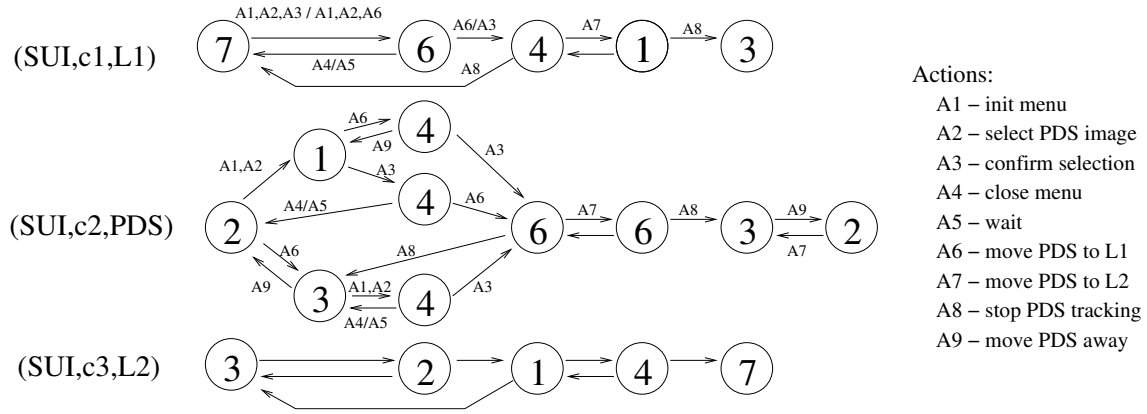

Figure 6.6: Coupling finite state automata for the PDS-based location control

Note that, the presented interaction requires the use of the menu-based control interface for initialization. Thenceforth, the interaction path is long. Alternatively, the PDS tracking can run continuously in the background, and the SUI can be moved from fixed surface to the PDS when the latter is put on the projected image. The user has only to place the PDS on the current interface position, without touching any interactive widgets. As the PDS is detected by the tracking software, the display is dimmed to provide detection feedback for the user. After a short dwell the SUI is uncoupled from S1, and coupled to the PDS. The delay allows the user to make corrective actions in case of a false detection of a PDS-like object. The SUI can be re-coupled to the target surface by touching the "freeze" button. Once the interface is moved from the PDS to S2, the tracking is disabled until the user moves away the PDS. The corresponding coupling life-cycles are shown in figure 6.7.

Coupling and usability properties Barralon et al. [60] propose to evaluate each of the automaton states against usability properties. This evaluation can inform the design of the interactive system. We illustrate this process by applying a subset of IFIP usability properties to our running examples.

Reachability describes the extent to which the system permits users to reach a desired state. Applied to interface steering, reachability describes how many states of the coupling model are covered using a given control interface. While small coverage of

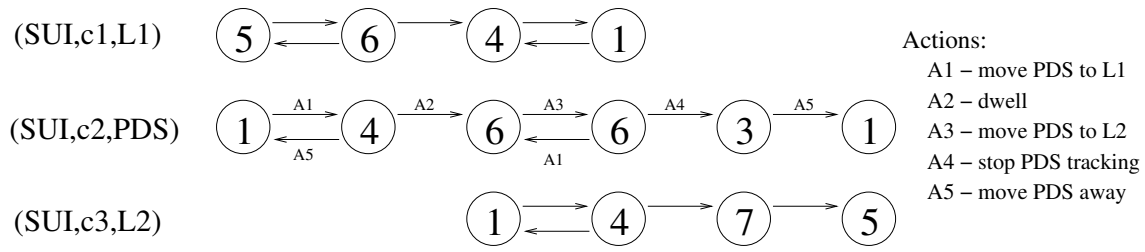

Figure 6.7: Coupling finite state automata for the menu-free PDS-based location control

the automaton may result in simple and efficient interaction, accessing more states may structure the interaction. Higher coverage of the automaton may also reflect support for a greater number of functions. For instance, using the menu based interface makes it possible to move the SUI between two surfaces and the life-cycle of couplings c1 and c2 cover 3 states of the coupling automaton each. On the other hand, interaction with the laser-based interface allows both fine-tuning the position of the SUI on a surface, and moving the SUI between surfaces. The corresponding coupling life-cycle is composed of up to 4 states. Similarly, a coupling model of the PDS-based interaction covers 5 states of the automaton, and the interaction allows the most flexible positioning of the SUI.

Non-preemptiveness refers to the extent to which the system allows users to choose the order of actions to perform a task. In this sense, only the PDS-based interaction offers a degree of non-preemptiveness during re-coupling the SUI from surface S1 to the PDS (see figure 6.6). Other interaction techniques are designed as immutable sequences of user actions.

In our examples the interaction always starts at location L1. All interactions require the user to first unlock the coupling (UI,c1,L1) and then to define the new location L2 for coupling (UI,c1,L2). Alternatively, a non-preemptive interaction technique would allow the user both to "send" the interface, and to "call" it to the new location.

Non-preemptiveness can also denote the ability of the system to directly reach any state of the coupling life-cycle. Coutaz et al. [21] explain; "this means that the automaton of (e1,c,e2) is completely connected (transitive closure)". While transitive closure is a desired property for sub-automata having the same value of predicate *Coupled*, it cannot be enforced between the two sub-automata. For instance, in the menu-based interaction, coupling (UI,c1,L1) can go from state 7 through 6 to state 3, but there is no direct connection between states 3 and 7. In fact, the state of the predicate *Coupled* can only be changed if predicates *Locked* and *Couplable* are set appropriately. States 4 and 6 of the automaton (figure 6.1) are exit-gates for the corresponding sub-automata. Therefore, a coupling can reach the state 4 only from states 1, 2, 3, and 6. Similarly, the state 6 can only be reached from states 5, 7, 8,

and 4. On the other hand, from both states 4 and 6 any other state can be reached. Figure 6.8 shows the fully connected model of a coupling.

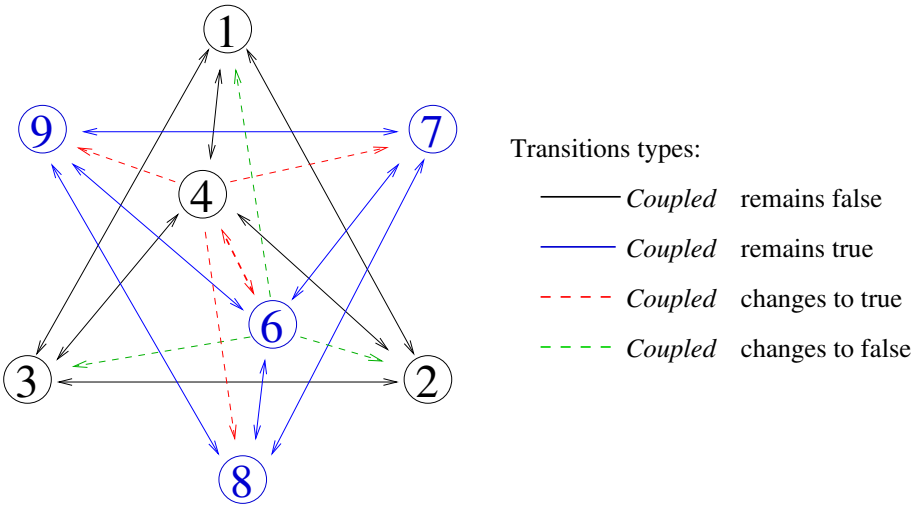

Figure 6.8: Coupling finite state atomaton model

Deviation tolerance or recoverability, describes the system's ability to support user-issued corrective action once an error is recognized, i.e. an undesired state has been reached. In the context of steerable interfaces, recoverability is ensured by the existence of a reverse transition for each state of the coupling life-cycle. Recoverability is supported by the laser-based interface, when using it to adjust the position of the interface. The user can move between all states of the coupling life-cycles. The interaction is fully reversible, i.e. the system can be set back to its departure state by performing actions in the reverse order. Note that, if an interaction for re-coupling an entity is fully reversible, the corresponding coupling life-cycles are symmetrical (figure 6.5). Note also, that some user action may be irreversible, because the coupling automaton is not transitively closed.

Recoverability is partially supported by the remaining interaction techniques, as all state changes, except the final coupling switch, are reversible. The user can recover from accidentally initializing the control interface. However, the interfaces do not support recovery from unwanted coupling the SUI to a new location (L2). It is therefore important to avoid such situations by, for example, asking the user to confirm the irreversible action before execution, like in the menu-based interface.

Observability refers to the extent to which the system provides support for the user to evaluate the internal state of the system from its perceivable representation. Applied to our interaction examples, this property denotes the ability to determine current state of a coupling. Coutaz et al. [21] argue that, "this property requires that every state of the automaton be made observable to users". Observability of every

coupling automaton state may help to explore the control interface by novice users. Note however, that the laser control interface is invisible until the user makes a sign with the laser pointer. As a result, a novice user may have difficulties to initialize the interaction. On the other hand, once the user learned how to use it, the invisibility of the control interface at standstill is less of an issue. Even if, in certain conditions, some states of the coupling automata may be left unobservable, it is important to provide adequate feedback for each major state change, i.e. state change that occurs simultaneously in both coupling c1 and c2.

Summary The presented interfaces offer a variety of interaction methods to move a SUI. We illustrated how to use the concept of coupling interaction resources to model steerable interfaces. Our analysis, revealed some shortcomings of the presented interactions, such as the existence of irreversible transitions in the coupling automata. It also allowed us to spot some issues that where not clear in the model presented by Barralon et al. [60], and Coutaz et al. [21]. For instance, we show that some of the states in the coupling automaton cannot be connected. Based on our analysis, we can draw some observations that apply to the design of SUI control interfaces:

- The fact that the locations L1 and L2 are coupled to the same entity (the SUI) results in an indirect coupling between L1 and L2. This coupling is reflected in the model by a level of symmetry between the coupling c1 and c2. A state change of coupling c1 results in a change of state in the coupling c2.
- State changes should be made observable to the user.
- Coupling states can be made observable, especially for non-expert users.
- If all state changes are reversible and the couplings c1 and c2 are composed of the same states but in reverse order, the interaction is reversible. Reversible interaction allows recovery.
- Some state changes are irreversible.
- Non-preemptive control interface should allow both sending the interface to a location, and calling from a location. Note, that such interaction requires the control interface to be unbound with the SUI.

While the presented analysis helps to understand the mechanisms involved in interface control, it does not provide an evaluation of the interaction techniques. In the next section, we evaluate interaction techniques for controlling an interface in the context of collaborative work.

6.2 Collaborative work through interface mobility

Steerable and portable interfaces will diminish the spatial constraints we are used to, and will allow new forms of interaction with computers. However, it is not clear how exactly steerable interfaces can support our activities. Without extensive user-studies, it is very difficult to anticipate how people will take advantage of interface steerability at everyday basis. The adequate approach would be to make available steerable interfaces to users in their natural environment and observe how the usage of such systems evolves over time. Unfortunately, our steerable interface technology is not yet mature enough for real-world deployment. Nevertheless, it is important to investigate application fields for steerable interfaces and strategies for their usage.

Butz and Krüger [14] propose the use of steerable projection to create peephole interfaces in instrumented environments. The peephole paradigm [111] emerged from combining the concepts of spatially situated information [31], and see-through interfaces [10]. It is based on the idea that a spatially situated 2D virtual layer can be viewed and modified through a window, i.e. the peephole. The virtual information layer is spatially continuous, and is bound to some reference frame in the physical world. Typically, peephole interfaces are implemented using tracked portable devices such as PDAs.

Butz and Krüger extend the application of the peephole metaphor to steerable interfaces by using a steerable camera projector pair. While our projection-based steerable interfaces can be described as a peephole, the use of this metaphor makes only sense for applications exploiting spatially situated information content.

Pingali et al. [66] present a vision of Augmented Collaborative Spaces (ACS), in which both local and remote collaborative work is supported by seamlessly merging electronic information with physical space. Similarly to Butz and Krüger, Pingali et al. note, that associating temporary information with "objects and 3D, makes it easier to remember what is where". Pingali et al. consider steerable interfaces as enabling technology to use the physical space as a scratch-pad. Users could distribute electronic data, such as documents or images, in the environment to allow both easy access to data, and to provide a natural way of sharing data with other users.

While Pingali et al. [66] only vaguely sketch the requirements for the future ACS, they highlight a potentially important application domain for steerable interfaces, i.e. Computer-Supported Collaborative Work (CSCW).

Computer-Supported Collaborative Work The CSCW acronym is usually referred to Computer-Supported Cooperative Work. The terms *cooperation* and *collaboration* are often considered as synonyms. However, Dillenbourg et al. [26] argue: "in cooperation the task is split (hierarchically) into independent subtasks; in collaboration cognitive processes may be (heterarchically) divided into intertwined layers." While cooperation requires coordination only when assembling partial results, collaboration is a process of constructing and maintaining a shared conception of a problem

through coordinated activity.

In a design process, collaboration happens more often in early stages, when a shared understanding of project goals is established. Once the goals are defined and tasks are attributed, cooperation takes over. However, Wu et al. [107] note, that designers frequently change working styles and the ways in which they communicate. As a result, cooperation and collaboration are present throughout the whole design process. Though steerable interfaces can be used for both cooperation and collaboration, we consider them most useful in colocated collaborative work.

A common practice for enabling colocated collaboration is to mirror the output of a conventional workstation on a large display. Though, such setup allows establishing a common view of work progress, users cannot easily integrate their ideas concerning the task. Only one person has the control over the application. Thus, letting others contribute is cumbersome. Because during collaboration contributors change frequently, it is important to provide users with simple and effective means for passing the control over the interactive system.

Single Display Groupware (SDG) [81], addresses the problem of sharing the input interface by multiplying input devices. However, the SDG is limited to either special applications supporting concurrent input [58, 85], or applications introducing access protocols that users must follow to share application control [42]. Multiplying interaction resources can clutter the work space, and make the interaction confusing.

In colocated collaboration, participants share the physical space and can use their natural skills for solving conflicts when interacting with each other, and with the interactive system. People are used to work on a common paper-copy of a document, and to exchange paper notes. Mobile interfaces can be used to exploit these practices to facilitate computer-supported collaborative work.

In this chapter, we illustrate how steerable and mobile interfaces can be used to support collaborative work. We present a presentation editing application to show how our technology for creating mobile interfaces can be exploited in the context of colocated collaborative work. We also evaluate, interaction techniques for controlling the interface location in a user study involving a projection-based drawing application. The results of our study provide important lessons about both CSCW applications, and mobile projected interfaces in general.

6.2.1 ContAct presentation authoring application

With the advent of new display technologies, our working habits are likely to change. To anticipate this trend, it is necessary to explore new interaction scenarios based on mobile interfaces. In this section, we describe a presentation editing application that, by making the interface portable, allows users to author a presentation collaboratively.

The ContAct application was developed as a demo for the Franco-Finish research project RNTL ContAct. It allows a group of people to collaboratively edit a presentation. The application interface is projected on a portable display surface using the

steerable camera-projector pair. Users can interact with this system by touching interactive widgets with their fingers. The interface is composed of vision-based interactive buttons (see section 5.1.2). Though the PDS allows interface motion in 3D, for the sake of ease of use it is posed on a table. Users sit around the table and are free to pass the PDS along, and to interact with the projected interface directly with their fingers. The application setup is presented in figure 6.9.

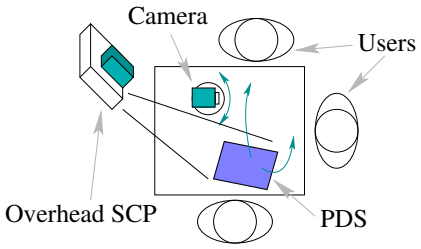

Figure 6.9: Overhead view of the ContAct application setup

The ContAct application allows groups of people to compose a presentation by combining on-line acquisition of images, text and video with pre-recorded material[1]. Users can browse a presentation, edit the content of the slide and capture data to add to the presentation. In each of these modes, the interface offers a different set of functions and a slightly different set of interactive buttons.

While browsing the presentation, users can delete slides and add empty slides. Each slide is automatically decomposed to content buffers. Typically, a slide having a title, an image and some text, is decomposed into a buffer containing the image, a text buffer with the title and another containing the body text. These buffers can be edited in the editing mode.

When editing the presentation, buffers can be dissociated from a slide and then deleted or coupled (moved) with a different slide. Text buffers can be edited using a wireless keyboard. Images can be passed to automatic character recognition software provided by Xerox[2] and transformed into text. Videos buffers can be visualized and played-back.

The ContAct application also allows the user to add content to the presentation. The camera mounted on the SCP can be used to digitize printed documents or physical objects. When the user requests to capture an image, the system displays a light colored polygon delineating the digitization area. Any object within this zone is photographed and displayed on the interface as an image buffer. The Xerox' Portable Document Camera (PDC) software is able to correct the orientation of a captured text and to enhance contrast of black and white image.

[1]A video presenting the ContAct application is available at `http://www-prima.inrialpes.fr/IHM06/ContAct.mpg`

[2]Xerox Research Center Europe (XRCE)

The application is also able to record user testimonies that may be integrated into the presentation. Based on the position of the PDS on the table (i.e. direction to which points the SCP), the system directs the steerable camera placed on the table towards the user demanding to record a video sequence. A color histogram based tracking algorithm is used to maintain the face of the user in the center of the camera image. The user can speak freely without being forced to remain in a fixed position in front of the camera. The recording can be stopped at any time using the ContAct application interface. Once the user has finished editing content buffers for a slide, the slide is automatically generated by the application.

The ContAct application allowed us to test our technology for mobile projected interfaces. It revealed some robustness issues related to the PDS tracking and interactive buttons implementation, which lead to the second implementation of both software (see sections 4.3 and 5.1).

The use of a lightweight portable display permits natural regulation mechanisms for the collaboration between users. Conflicts in accessing the limited interaction resources (i.e. application interface) are solved verbally in an intuitive manner, without any intervention by the computer system.

The PDS offers a portable interface. Alternatively, users may collaborate using a steerable interface. In the next section, we compare how people perceive both steerable and portable interfaces in the context of a collaborative task.

6.2.2 Comparing modalities for passing the control of an interface

In previous sections, we presented the concept of steerable interfaces, our implementation of projection-based mobile interfaces, interaction techniques for moving an interface in an environment, and an application exploiting our technology to support collaborative presentation editing. We know that, interface steerability can be exploited in HCI, and how to implement such feature. However, we still know very little about the use of steerable interfaces. In particular, we do not know whether the flexibility offered by steerable interfaces compensates the added interaction complexity related to spatial control.

In this section, we present a user study that aims at evaluating user preferences with respect to spatial control of interfaces in the context of collaborative work. The study was prepared together with Jérôme Maisonnasse and Julien Letessier.

We compare four different methods for passing the control of an interface. We evaluate both user acceptance and user performance for each of the following experimental conditions:

- *Fixed:* the application interface is projected on the tabletop and is fixed, i.e. cannot move. Users have to organize the space around the interface to allow everyone to contribute.

- *Button:* the interface can be passed between users by explicitly choosing the next contributor from a list of users projected at the side of the display.
- *PDS:* the PDS as an embodiment for the interface allows users to physically pass it between each other. In this condition the interface is portable.
- *Dialog:* a conversational-rules based dialog model is used to determine which user is currently contributing to the work. In this case the SUI control is implicit, and the interface is steered by the system.

We test the system by asking groups of people to collaboratively draw graph-like sketches. The experiment is designed for groups of three participants, but the interface can be operated only by a single user at a time. Users sit around a table, like in the setup for the ContAct application. The application interface is projected with the SCP. The camera of the SCP observes the scene through a IR filter. A lamp mounted above the table illuminates the table with both visible and infrared light.

The application interface The application allows users to draw colored strokes on the interfaces with their fingers. A palette of four colors is placed next to the left border of the interface. The drawing zone is rendered in black as the background color. One of the colors in the palette is dark-grey, which can be used to erase strokes. To ease erasing, the grey-colored line is four times thicker than other lines.

We use a vision-based finger tracker[3] to track individual or multiple fingertips. While multiple fingertips allow more advanced interaction techniques [54, 63], we prefer to limit the interaction style to the "point and click" paradigm. This limits the learning curve of our drawing application, and thus minimizes the impact of personal learning skills on the experiment.

The "point and click" paradigm is typical for conventional user interfaces. A mouse provides the ability to point at interactive widgets with the cursor and to click on them with the mouse button. This "point and click" function corresponds to pointing and touching objects with a finger. Unfortunately, the monocular finger tracking cannot detect when a finger touches the display surface. As a result, color selection in our drawing application is an issue.

Typically, discrete events such as mouse button clicks are simulated based on finger trajectory [47] or finger dwelling [11]. However, both of these methods constrain user gestures and can occasionally result in false detections. We have addressed this issue using two-handed input. To draw a stroke, the user has to choose a color with one hand and use the other hand for drawing. Nevertheless, a short dwell of both fingertips is required to start drawing.

To provide visual feedback of the fingertip tracking, white spots are projected on the detected fingertip positions. We limit the number of "active" fingers. There can be only one fingertip pointer in the drawing zone, and one in one of projected buttons.

[3]The finger tracker (see section 5.1) was developed by Julien Letessier

In both *Button* and *Dialog* conditions, the interface moves between predefined locations on the tabletop. In the *Button* condition, buttons with names of the users are displayed next to the right border of the interface. To move the interface to a person, the user has to put a finger and dwell shortly in the button with the person's name.

The buttons with user names are only visible in the *Button* experiment. To provide equal application interface throughout the experiment, in remaining conditions, the name-button area is delineated with a frame, and cannot be used for drawing.

During the experiment, users wear microphone headsets. We use a speech detection software[4] to determine when users speak. Speech events are used by a dialog modeling service[5] to determine which user is contributing to the task. The output of the dialog model software is used to control the position of the interface in the *Dialog* condition.

In both *Button* and *Dialog* conditions, the interface movements are blocked as long as an active fingertip is present in any of the color buttons or in the drawing zone. This limits unwanted interface movements that could result from false fingertip or speech detections.

The dialog model In the *Dialog* condition, the system detects which user wants to contribute, and moves the interface to that person. The detection is based on temporal analysis of speech events. The analysis is driven by conversational rules established in the field of psychology.

The dialog model is grounded on two assumptions: *(1)* speech refers to the task, and *(2)* a good collaboration is a succession of interleaving contributions from all participants.

The first assumption suggests that, when a user speaks, he or she wants to contribute, and that the interface should be moved to that person. However, not all speech events are considered equally important. In conversation, people show their attention by inserting sounds, words or short phrases in the dialog. These dialog reinforcements are detected by the system so that the interface remains with the user leading the conversation.

The second assumptions suggests that, people speaking at the same time are either arguing, or are not focused on the task. In such situation, the system is unable to attribute the interface to any of the users, and after a short time moves the interface away to a neutral position. In this sense, the system acts as a moderator for the collaboration.

Task We designed the task to enforce collaboration between users. We also wanted to limit the ability of participants to direct other users what to do, instead of doing it by themselves. The goal is to recall and draw elements of a graph-like map. Maps are

[4]The speech detector was developed by Dominique Vaufreydaz

[5]The dialog modeling software was developed by Jerome Maisonnasse

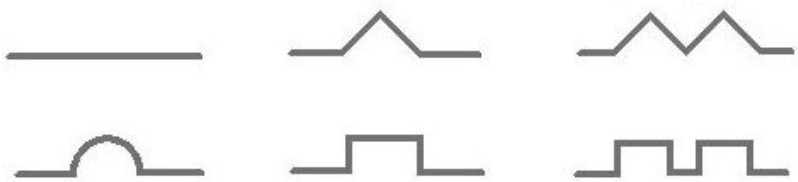

Figure 6.10: Segment forms present in maps

constructed of fifteen segments, each having one of six basic forms shown in figure 6.10.

Participants are asked to memorize six segments on a map. In particular, we ask them to note the form of a segment, its position with respect to other elements of the map, and the side of the segment on which it has its characteristic pattern. For example, for a vertical arc-segment a user has to remember that the segment has an arc pattern, that its upper-end is connected to some other segments, and that the arc is on the right side of the segment. We do not require the exact orientation of segments, and users are free to modify the shape of the reconstructed map.

We show to each user the same map, but with a different set of highlighted segments (figure 6.11). Two of the six highlighted segments are common with the two other experiment participants, i.e. each pair of users has one segment in common. The highlighted elements are distributed on the graph such that segments highlighted for one person interleave with elements memorized by others.

Each of the participants memorizes a part of a map. As a result, the map can be reconstructed only by combining contributions of all participants. Because the fifteen segments of each map are at different angles, the graph has little structure which makes it difficult to verbally instruct a subject to draw an element that she or he did not memorize.

We evaluate the recall scores individually for each subject. Each segment characteristics (i.e. position, form and orientation) have equal weight. A user can obtain a maximum of 18 points for all six correctly recalled segments. If a group draws more than 15 segments, one point is subtracted from the score of each subject. If a group draws less than 15 segments, the user (or users) who was supposed to memorize the missing segment looses 3 points and others loose 1 point each.

Protocol For each group the experiment is carried out in one block of all four experimental conditions. At the beginning, we present the context of this study and explain the goal of the experiment. We briefly present the elements of the interactive system. To explain the task, we show both an image of the six basic segment forms, and a sample map with highlighted elements.

To familiarize users with the interface, we ask each of them to draw a number of

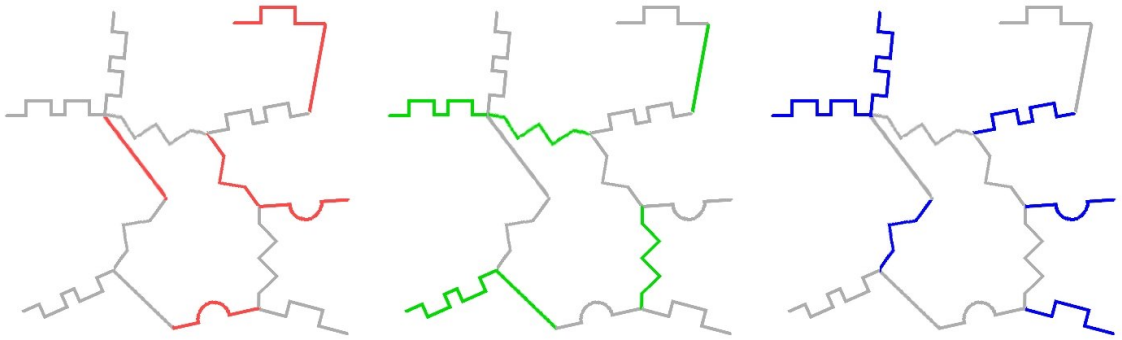

Figure 6.11: A sample set of maps shown to participants of the experiment

segments assembled in different figures. Depending on users' skills, this training can take between five and ten minutes for a group. We also explain recovery techniques that can be applied if finger tracking errors occur during the experiment.

Once participants are familiar with using fingers of both hands to draw, the actual experiment starts. For each experimental condition we follow a four step procedure:

1. We distribute a set of maps, each showing the same map with a different set of highlighted segments. For each condition the group memorizes a different map. Participants have no more than 4 minutes to memorize the graph. Once the time is up, we remove the maps.

2. We explain which interaction technique for moving the interface is to be used in this condition. The explanatory speech is a distraction introduced to eliminate short-term memory effects and retinal image memory.

3. The group has no more than fifteen minutes to draw the memorized map.

4. We briefly compare the image drawn by the group with the map they had to memorize. There is about 4 minutes between two experimental conditions.

At the end of the experiment, we ask participants to complete a questionnaire. In the questionnaire, we ask subjects to rank the experimental conditions in order of preference, and to evaluate on a scale from 1 to 5 a number of factors related to how they perceived the experimental conditions. We also ask them informally about their impressions of the system and the experiment.

The complete experiment takes approximately one and a half hour. We prepared four different maps, and we take care to vary separately the order of maps and conditions for each group.

Subjects We performed the experiment with 12 groups of 3 people. In fact, we had 13 groups, but due to a software failure, the first group interrupted the experiment after two conditions. As a result, we exclude the results of the first group. All numbers and results are calculated for groups from 2 to 13.

In the twelve valid groups, there were 13 women and 23 men. Age of the participants varies between 18 and 53 with an average of 27.76 and a variance of 35.63. Nineteen subjects study or work in fields related to computer science, others (17 subjects) have human-sciences background.

In all groups subjects knew each other beforehand. Most of the groups were composed of friends, and only few were assembled from people knowing each other only from work.

Hypotheses Before we ran the experiment, we had a number of hypotheses about the subjects' preferences:

H1: Subjects would prefer conditions with a mobile interface over the *Fixed* condition for a collaborative task.

H2: From the conditions where interface is mobile, subjects would like the best the *PDS* condition, where the interface is portable.

H3: Subjects would prefer explicit control of the interface position over implicit control of the *Dialog* condition.

We also had some hypotheses about the performance:

H4: Subjects would have the highest recall scores in the *Fixed* condition. Because participants are free to move and gather close to the interface, they can all see the interface from the same side. Thus the task is simpler than in other conditions where subjects see the interface rotating.

H5: Out of the conditions where the interface is mobile, the *Dialog* condition would yield the best recall scores. In *Dialog* condition, users do not spend their resources on controlling the interface; the interface moves "by itself".

H6: The *Dialog* condition would be the fastest accomplished. In this condition, the system works as a moderator and forces the group to focus on the task. Other conditions are more permissive, and allow users to work more chaotically.

Finally, we had two hypotheses concerning the collaboration within groups:

H7: A high number of changes in who contributes is a symptom of a good collaboration and thus yields higher recall scores.

H8: A high number of speech overlaps would result in lower recall scores. Overlaps are violations of conversation conventions, and thus indicate bad collaboration.

	Preference (best = 1, last = 4)											
	Fixed			Button			PDS			Dialog		
Group No	A	B	C	A	B	C	A	B	C	A	B	C
1	1	1	1	2	2	2						
2	2	1	4	3	3	3	4	4	2	1	2	1
3	3	3	1	1	2	3	2	1	2	4	4	4
4	3	4	4	1	2	1	2	1	3	4	3	2
5	1	3	2	3	1	1	2	2	3	4	4	4
6	4	1	4	2	2	1	3	3	2	1	4	3
7	2	2	1	3	3	4	1	1	3	4	4	2
8	4	4	4	2	2	2	3	3	3	1	1	1
9	4	2	4	3	3	3	1	1	1	2	4	2
10	3	3	4	1	2	2	2	1	1	4	4	3
11	3	3	4	2	1	3	4	2	2	1	4	1
12	4	4	3	2	3	2	3	2	4	1	1	1
13	4	4	4	3	2	2	2	1	1	1	3	3
Mean	3.06			2.19			2.17			2.58		
Variance	1.16			0.66			0.97			1.69		

	Recall score (max =18)											
	Fixed			Button			PDS			Dialog		
Group No	A	B	C	A	B	C	A	B	C	A	B	C
1												
2	13	11	12	16	18	17	14	12	17	15	16	16
3	18	17	18	17	16	14	16	14	18	18	14	13
4	16	18	13	18	13	16	7	12	12	11	17	13
5	15	13	14	17	14	15	18	14	17	15	17	17
6	15	17	16	11	8	17	18	18	18	18	17	18
7	14	18	17	17	16	13	16	12	15	18	17	17
8	11	11	12	15	16	17	15	11	10	15	16	15
9	18	18	18	18	16	18	17	18	18	18	18	18
10	17	18	18	18	18	18	18	18	18	17	17	18
11	17	14	14	17	16	18	11	16	17	18	18	18
12	16	17	12	14	17	17	18	17	11	13	12	9
13	17	18	16	18	18	18	18	16	18	17	18	18
Mean	15.47			16.11			15.36			16.11		
Variance	5.69			4.82			8.68			5.15		

Table 6.1: *Left:* User preferences, i.e. condition ordering. *Right:* User recall scores (18 is the maximum score). Colored cells correspond to female subjects. First group is discarded.

Results We consider each experimental condition as a separate factor, and use the ANOVA[6] test to verify whether any mean value of a measured parameter is significantly different between conditions. If ANOVA rejects the hypothesis that all means are equal, we proceed with 2-tailed t-tests between condition pairs. We use "f", "b", "p" and "d" subscripts to denote data referring to experimental conditions *Fixed*, *Button*, *PDS* and *Dialog* respectively.

Tables 6.1 summarize subjects' preferences and recall scores. Characters "A", "B" and "C" correspond to the three subjects composing each group. In the questionnaire, subjects ranked experimental conditions, 1 correspond to the most liked condition and 4 to the least liked one. The maximum recall score a subject could obtain is 18.

There are no significant differences of recall scores between experimental conditions, and the average scores are very similar. Therefore, the hypotheses H4 and H5 are not supported. However, when scores are arranged chronologically, there is a significant difference between the recall scores obtained in the first and both third and fourth test. This suggests that during the experiment users learned how to solve the task efficiently. The respective average scores are 14.72, 16.44 and 16.19.

In average, users preferred the *PDS* condition, followed by *Button* and *Dialog* conditions. Subjects disliked the *Fixed* condition the most. The probability of all four mean scores being equal is $p = 0.0015$. The differences between means of the *Fixed* condition and both *Button* and *PDS* conditions are significant ($p = 0.0003$ and $p = 0.001$ respectively). However, there is no significant difference between *Fixed* and *Dialog* conditions, which does not support hypothesis H1. In fact, in terms of user preference, there is no significant difference between the *Dialog* condition and any other condition,

[6]Analysis of Variance (single factor) available in Microsoft Excel

	Non-expert computer users						
Group	User ID	Sex	Age	Fixed	Button	PDS	Dialog
2	1	M	21	4	3	2	1
	2	M	20	1	3	4	2
	3	M	18	2	3	4	1
4	7	M	25	4	1	3	2
	9	F	26	3	1	2	4
5	10	F	26	3	1	2	4
	12	M	29	2	1	3	4
6	14	M	32	1	2	3	4
	15	M	30	4	1	2	3
	13	F	28	4	2	3	1
7	18	F	25	1	4	3	2
8	19	F	25	4	2	3	1
	20	F	24	4	2	3	1
	21	F	24	4	2	3	1
12	31	F	35	4	2	3	1
	32	F	24	3	2	4	1
	33	F	53	4	3	2	1
Mean			27.35	3.06	2.06	2.88	2
Variance			57.88	1.35	0.76	0.46	1.53

	Expert-level computer users						
Group	User ID	Sex	Age	Fixed	Button	PDS	Dialog
3	4	M	23	1	3	2	4
	5	F	23	3	2	1	4
	6	F	24	3	1	2	4
4	8	M	25	4	2	1	3
5	11	M	32	1	3	2	4
7	16	M	27	2	3	1	4
	17	M	29	2	3	1	4
9	22	M	25	4	3	1	2
	23	M	34	2	3	1	4
	24	M	27	4	3	1	2
10	25	M	32	3	1	2	4
	26	M	33	3	2	1	4
	27	M	33	4	2	1	3
11	28	M	25	3	2	4	1
	29	M	29	3	1	2	4
	30	F	33	4	3	2	1
13	34	M	25	4	3	2	1
	35	M	25	4	2	1	3
	36	M	29	4	2	1	3
Mean			28.05	3.05	2.32	1.53	3.11
Variance			13.63	1	0.53	0.57	1.25

Table 6.2: User preferences: users with medium experience with conventional computing *(left)*, and experienced desktop computer users *(right)*.

which would suggest that we can support neither of hypotheses H1, H2 nor H3.

During the experiments, we noticed that the *Dialog* condition is often rated either as the most liked or the most disliked. As a result the distribution of subject preferences for this condition violates the assumption of normal distribution used by both the t-test and the ANOVA test. The distribution has two maxima which correspond to the two most often ranks chosen by users for the *Dialog* condition. The presence of two peaks can be explained through the existence of two distinct groups of subjects.

Tables 6.2 shows subjects' preferences. The left table contains the votes of users having little or average experience with conventional computers. The right table shows results for subjects that are experts in using conventional computers. In the following, we refer to the first group as to "non-experts group" and the second, more experienced group, as to "experts group". All subjects in the experts group have an engineering degree in information technology, electronics or related field. All of them know at least one programming language. We use an "n" superscript to denote data referring to the non-experts group, and "e" superscript for the experts group.

The two groups rated the *Dialog* and the *PDS* condition differently. The non-experts liked *Dialog* condition the best, but didn't like to use the PDS. On the contrary, the experts group in average rated the *PDS* condition the best, and the *Dialog* condition as one of the two least liked. The between-groups differences for the *Dialog* and the *PDS* condition are significant with $p = 0.01$ and $p = 3 \cdot 10^{-6}$ respectively.

The *Fixed* condition and the *Button* condition was rated by both groups similarly, the average ranks are $\mu_f^n = 3.06$ and $\mu_f^e = 3.05$ for the *Fixed* condition, and $\mu_b^n = 2.06$ and $\mu_b^e = 2.32$ for the *Button* condition.

G	Id	score				ordered score			
		fix	butt	pds	dial	1	2	3	4
2	1	13	16	14	15	13	14	16	15
	2	11	18	12	16	11	12	18	16
	3	12	17	17	16	12	17	17	16
4	7	16	18	7	11	7	11	16	18
	9	13	16	12	13	12	13	13	16
5	10	15	17	18	15	15	17	15	18
	12	14	15	17	17	17	15	14	17
6	14	17	8	18	17	8	17	18	17
	15	16	17	18	18	17	18	18	16
	13	15	11	18	18	11	18	18	15
7	18	17	13	15	17	17	15	13	17
8	19	11	15	15	15	15	11	15	15
	20	11	16	11	16	11	11	16	16
	21	12	17	10	15	10	12	15	17
12	31	16	14	18	13	18	16	14	13
	32	17	17	17	12	17	17	17	12
	33	12	17	11	9	11	12	17	9
mean		14	15.41	14.59	14.88	13.06	14.47	15.88	15.47

G	Id	score				ordered score			
		fix	butt	pds	dial	1	2	3	4
3	4	18	17	16	18	18	17	18	16
	5	17	16	14	14	14	16	17	14
	6	18	14	18	13	13	14	18	18
4	8	18	13	12	17	12	17	18	13
5	11	13	14	14	17	17	14	13	14
7	16	14	17	16	18	14	16	17	18
	17	18	16	12	17	18	12	16	17
9	22	18	18	17	18	18	17	18	18
	23	18	16	18	18	16	18	18	18
	24	18	18	18	18	18	18	18	18
10	25	17	18	18	17	17	18	17	18
	26	18	18	18	17	17	18	18	18
	27	18	18	18	18	18	18	18	18
11	28	17	17	11	18	17	18	11	17
	29	14	16	16	18	14	18	16	16
	30	14	18	17	18	14	18	17	18
13	34	17	18	18	17	17	18	18	17
	35	18	18	16	18	18	16	18	18
	36	16	18	18	18	18	18	18	16
mean		16.79	16.74	16.05	17.21	16.21	16.79	16.95	16.84

Table 6.3: Recall scores: users with medium experience with conventional computing *(left)*, and experienced desktop computer users *(right)*.

In the non-experts group, differences between the two most liked conditions (*Button* and *Dialog*) and the less liked conditions (*Fixed* and *PDS*) are significant. On the other hand, the differences between the *Button* and *Dialog* and the *Fixed* and *PDS* conditions are insignificant. For this group, the results do not support any of the three hypotheses H1, H2 and H3.

In the computer experts group, only the *Fixed* and *Dialog* average ratings are similar ($\mu_f^e = 3.05$ and $\mu_d^e = 3.11$), others are significantly different. These results support hypotheses H2 and H3, but do not support hypothesis H1.

Table 6.3 shows recall scores for both groups. The experts group, has in average higher recall scores than the non-experts group, 16.7 and 14.72 respectively ($p = 2 \cdot 10^{-6}$). However, when comparing conditions separately, the differences between average scores in *Button* and *PDS* conditions are not statistically significant. Scores within each group are similar between conditions. Thus we cannot support hypotheses H4 and H5.

The learning effect is visible in both groups, as the worst mean recall score was obtained in the first test; $\mu_1^n = 13.06$ and $\mu_1^e = 16.21$. The differences between scores ordered chronologically are significant only in the non-experts group between the first and both the third ($p = 0.01$) and the forth ($p = 0.02$) passed condition.

Considering all subjects, the average times for accomplishing a condition vary from 8.5 minutes for the *Button* to 12 minutes for the *PDS*. Although the difference between mean times is large, it is not statistically significant. Therefore, hypothesis H6 is not supported.

The control over the interface was passed the most number of times in the *Dialog* condition, in average 33.83 times. The interface was moved the least in the *PDS*

condition with 6.75 turn-takings per test. Though the differences between average number of turn-takings are significant, we cannot support the hypothesis H7, because there were no differences in average recall scores between conditions. For the same reason we do not support hypothesis H8.

Results discussion Of our hypotheses, only hypotheses H2 and H3 were supported by the study, and only in the experts group. Hypothesis H1 was not supported in both the experts and non-experts groups. However, both groups rated the *Fixed* condition as one of two least liked. While each of the two groups disliked one of the mobile interfaces, they liked the two other mobile interfaces the most. As a result, we can state that in general, subjects enjoyed more to work with a mobile interface than with the fixed interface.

The non-experts group disliked the *PDS* condition as much as the fixed interface. Note that, this is the opposite of what we hypothesized in H2. On the other hand, the *PDS* was the preferred condition in the experts group. With 58% of subjects choosing it as the best condition, it is significantly more liked that the *Button* condition ($p = 0.003$) and the *Dialog* condition ($p = 2 \cdot 10^{-5}$).

Compared to the experts group, the non-experts group felt that the *PDS* condition is less intuitive, with average scores on a 5 points scale of $\mu_p^n = 3.88$ and $\mu_p^e = 4.53$ and $p = 0.02$. They also judged the PDS as less reactive ($\mu_p^n = 2.65$, $\mu_p^e = 4.05$, $p = 0.0003$) and less suited for the task ($\mu_p^n = 3.18$, $\mu_p^e = 4.42$, $p = 0.0001$).

We can look for an explanation of the differences in evaluating the *PDS* condition in the vision-based implementation of the PDS. While subjects moved the cardboard screen on the table they often occluded the camera view of the PDS, which sometimes made the PDS tracking fail.

Before the *PDS* condition, we explain that the system might loose track of the portable screen, but that it can find the PDS as soon as the borders of the cardboard are visible to the camera. We asked users to ignore tracking errors and focus on the task.

It seems that subjects from the non-experts group were more disturbed by tracking errors. Non-experts are likely to perceive the image projected on the PDS as an extension of the physical cardboard, thus they have difficulties to accept when the illusion is broken. On the contrary, subjects with good understanding of information technology, are more aware of the "mechanics" behind the PDS application, and can more easily cope with tracking errors. They are also more likely to take into account technical limitations of the PDS implementation and adapt their movements to the camera-PDS configuration. Finally, experts group ranks the interaction with the PDS as the most fun ($\mu_p^e = 4.53$, $\mu_f^e = 2.32$, $\mu_b^e = 3.21$, $\mu_d^e = 3.89$). In the experts group, the PDS is considered as technological gadget, a curious application of computer vision. As a result, subjects from the experts group are more forgiving towards the PDS tracking.

The results in the non-experts group are biased because of the occasional failures of the PDS tracking. The experts group seems to be influenced by the innovative aspect of the interface projected on a tracked cardboard. As a result, the study does not support the hypothesis that subjects would like more a portable interface than a steerable interface (H2).

The hypothesis H3 suggests that subjects would prefer the explicit control of the *PDS* and *Button* conditions over the implicit control offered in the *Dialog* condition. While the hypothesis H3 is supported in the experts group, it is not in the non-experts group, where both *Button* and *Dialog* conditions are preferred equally.

In the *Dialog* condition, all of the groups modified their verbal communication to adapt to the dialog model of the system. However, the appreciation of the necessity to adapt to the system was different in each of the two groups. As opposed to the experts group, non-experts consider the *Dialog* condition the most fun to use ($\mu_d^n = 4.47$, $\mu_f^n = 3.06$, $\mu_b^n = 3.35$, $\mu_p^n = 3.06$). However, both groups evaluated the interaction in the *Dialog* condition the least predictable.

The two groups are likely to experience differently the interface controlled by the system, because of how they apprehend information technology in general. The experts group is composed of programmers for whom an unexpected behavior of an interactive system is synonymous with an error. They have stronger a-priori opinion about the functioning of computers. On the other hand, the non-expert users have less practice with computers and thus are less limited by learned usage patterns. As a result, non-experienced users are more likely to enjoy non-conventional interaction.

A number of subjects in the non-experts group enjoyed the system surveilling the collaboration of the group and influencing the interaction. They pointed out, that the *Dialog* condition forces them to respect more strictly conversation rules, which results in a more organized collaboration. As subjects tend to control their behavior, they are more focused, and in consequence their actions are more elaborate and better coordinated with other users. Subjects from the experts group, perceived the same features as limiting.

Subjects from the group number 8 (non-experts) noticed that, the *Dialog* condition helps reserved people to contribute equally with others. Since the interface is attributed by the system, it is more difficult for a person to monopolize the interaction. If a person says something the system might decide to move the interface to that person. While the non-expert subjects considered that a shy person can be encouraged by the system to contribute, subjects from the experts group found this behavior intimidating.

Depending on the background of the subjects, the hypothesis H3 may be supported. We can expect that users having strong interaction habits learned with conventional computers would prefer explicitly controlled mobile interfaces.

Hypotheses H4 and H5 are not supported as there are no significant differences between recall scores obtained in different conditions. We suspect that the differences were hidden by the observed learning effect.

Though in the *Fixed* condition the recall task is easier than in other conditions,

subjects performed similarly in all conditions. In fact, some of the subjects complained that having an interface with fixed orientation is a drawback, as it is convenient only for the person in front of it. The interaction is "monopolized" by the user sitting in front of the interface. At the same time, subjects enjoyed the ability to see the interface simultaneously with approximately same orientation for all. Nevertheless, there is no clear evidence to support either H4 or H5.

Similarly our study does not support hypothesis H6, conjecturing that *Dialog* condition would be the fastest accomplished. Note, that some of the groups where composed of both experts and non-experts. Therefore, we only compared average times of all groups together.

The hypothesis H7 suggests that a higher number of interface control turn-taking would coincide with higher recall score. We are not able to support this thesis, because there are no significant differences in recall scores between conditions. Nevertheless, we note that there are significant differences in the number of turn-taking during interaction: $\mu_f = 17.73$, $\mu_b = 9.09$, $\mu_p = 6.75$, $\mu_d = 33.83$. The differences between average numbers are significant except for the *PDS*, *Button* pair. In the *Fixed* condition, the turn-taking was counted manually by a human operator.

Note that, the number of turn-taking in the *Dialog* condition is 5 times higher than in the *PDS* condition. However, approximately 33% of the number for the *Dialog* condition were contributions shorter than 5 seconds. The average number of turn-taking, where each turn was longer than 5 seconds are: $\mu_f = 14.64$, $\mu_b = 8.64$, $\mu_p = 5.91$, $\mu_d = 22.58$. The correlation coefficients between the number of turn-taking (longer than 5 seconds) and the average scores are: $c_f = 0.03$, $c_b = -0.42$, $c_p = -0.13$, $c_d = 0.11$.

The high number of turn-taking in the *Dialog* condition was perceived by a number of subjects as disturbing. They complained that the interface moved too much, and that sometimes it moved away while they were looking at it trying to remember missing segments.

The small number of turn-taking in the *PDS* condition can be partly explained by the way it was used by some groups. There were two groups that put the PDS between two users to establish a common view for at least a part of the group. As a result, they only had to move the interface when the third user would like to contribute.

General observations and usability issues Subjects in general had little difficulties to learn how to operate the interface. However, there was one left-handed subject, who had some troubles in the training phase because the placement of color buttons favorites right-handed users. Though she had to spend more time for training before the tests, she got used to draw with the right hand pretty quickly, and her performance was comparable with that of other subjects.

During the experiments, several users had sometimes trouble to initiate the drawing function. To draw a stroke, a user has to enter a color button with one finger and

use a second finger to draw. However, a fingertip starts drawing only after it dwells for a short time in the drawing zone. The dwell allows first selecting a color, and then positioning the drawing finger at the beginning of the stroke. To stop drawing the user has to move the fingertip away from the color button. This mechanism was found to be not very intuitive. The fact that we offered the ability to use different modalities to start and stop the drawing function was confusing for users. We should have rather restrict the interaction such that when a fingertip is in a color button the other fingertip starts drawing immediately, or allow both initialization and canceling of the drawing action with a finger dwell. Fortunately, this issue occurred rarely.

In the informal discussion following the experiment, several subjects noted that they found it easier to keep track of the interaction in the *PDS* condition than in *Button* and *Dialog* conditions. They suggested that the continuous movement of the PDS helps them to cope with changing orientation of the interface. Fast movements of the interface in *Button* and *Dialog* conditions resulted in an additional cognitive effort to re-calculate the view of the interface. This was particularly disturbing in the *Dialog* condition where the interface movements were less predictable to users.

Our observation supports the findings of Pinhanez and Podlaseck [68], who note that users can have difficulties "to follow the path of interfaces that jump discontinuously from one surface to another." While their observation refers to large interface movements, i.e. movements between surfaces scattered in the environments, it applies also to smaller movements within the same surface. This problem can occur when the steerable interface is either rapidly moved and rotated like in our experiment, or "teleported" between surfaces in the environment.

The *Button* condition was appreciated by most groups. It was considered simple, fast and efficient. When we designed the menu based interface, we expected users to use buttons to send the interface from current user to the next one. We expected that subjects would agree who is the next contributor and that the current user would press the corresponding button to pass the interface. In practice, most of the time the interface was "pulled" by the user who wanted to contribute. This observation, suggests that a good control interface for a SUI, should support both "pushing" and "pulling" interaction style.

During the experiment, we noticed that most subjects entered the color buttons from the side. They tried to avoid occluding other colors with their hand. Even when instructed to place their finger along the side of the interface, after a while they continued to enter the interface from the side. Some of the subjects were afraid that occluding more than one button with their hand would break the drawing function. One subject explained that he felt that the system is sensitive to his skin, and thus it would be ambiguous to enter more than one color button at a time. Note that, the finger tracking does not use color information to detect fingers (see section 5.1). The subject had the impression that any widget projected on his skin would be triggered. This observation supports the findings of Podlaseck et al. [69] and Kjeldsen et al. [47].

Podlaseck et al. [69] report that the conventional function of objects can interfere

with the function added by the projection. In their study, Podlaseck et al. asked a number of subjects to pick a colored rectangle from a set of five displayed on different objects. Objects varied from a monitor screen, through a sheet of paper to the surface of milk. People tend to ignore interactive widgets projected on objects that have a clearly defined function. Podlaseck et al. report that rectangles projected on the surface of milk was not perceived at all by 10% of participant of the experiment. Similarly, in [47] Kjeldsen et al. note that people more often ignored a button projected on a bucket of sweets than a similar button projected on a wall or a table. Our observations lead to consider that virtual elements having a clearly defined function can influence user perception of both the physical object on which they are projected, and that of other virtual objects projected in a close proximity.

Pinhanez and Podlaseck [68] highlight another cognitive issue related to projected interfaces. They argue whether projected interfaces should be delineated with a frame. Though in our study we used exclusively images projected in a frame, we find important to mention this issue to provide a more complete discussion of usability of projected mobile interfaces.

Pinhanez and Podlaseck [68] define a *frameless display* as "a display with no perceptible boundaries", which "appears to be embodied in the physical world." They analyze several applications in which interfaces are projected on mundane surfaces, and identify a set of guidelines for the design of projected frameless graphics. They note that frameless displays have the ability to "embed characters into reality" or to "transform positively the perception of an object". On the other hand, they suggest the use of frames for images showing information "disconnected from the environment". Most conventional interactive applications have no connection with the environment, and thus should remain in a frame even when displayed with a projector.

Pinhanez and Podlaseck [68] note that "a frame creates and indicates spatial disruption". A frame can "provide a metaphorical window into another world which is disconnected from the viewer in space and/or time". In other words, a frame can help to separate the projection from the display surface. Therefore, it can be expected that the interference between the original function of an object and the function added through projection should be less important when displaying widgets delineated with a frame.

6.3 Summary of the chapter

In this chapter, we addressed the issues of designing interaction techniques for controlling the position of steerable interfaces, and of evaluating their usability in the context of collaborative work.

We proposed a number of interaction techniques and control interfaces for moving a steerable interface. The interfaces are designed to offer maximum control with a minimal graphical footprint. We propose both interaction techniques that can be

realized with bare fingers and techniques requiring the use of dedicated instruments such as a laser pointer or a PDS.

We analyze the proposed interaction techniques using both the framework of Barralon et al. [60] for coupling interaction resources, and the IFIP usability properties. While the concept of coupling interaction resources offers a convenient reasoning framework for multi-surface interaction, it had to be adapted for steerable interface. In particular, we consider locus of the interface as a resource at the same level as physical resources such as display screens of conventional work stations.

The application of the coupling framework allowed us to identify some shortcomings of our interaction techniques, such us the preemptiveness of the menu-based interface, or the long interaction path when moving the interface with the PDS. It also revealed some elements of the coupling framework that had to be clarified. For instance, we noticed that the coupling finite state automaton cannot be transitively closed, i.e. some transitions between states are irreversible.

In the second section of this chapter, we discussed which application fields can take the most advantage from interface steerability. Besides future working environments augmented with localized information [66], where steerable interfaces can offer a mobile peephole to the information [14], we identify computer supported collaborative work as a potential field for steerable interface applications.

To illustrate how mobile interfaces can support collaborative work, we present a prototype application for authoring a presentation. The ContAct application interface is projected on the PDS, and allows groups of people to compose a presentation by combining on-line acquisition of images, text and video with pre-recorded material. We designed the ContAct application as a proof of concept prototype, which allowed us to improve our projection-based technology, and to prepare the ground for a user study aiming at comparing fixed, portable and steerable interfaces.

In the study, we asked 36 subjects divided into 12 groups to recall and collaboratively reconstruct graph-like maps. Users can draw on the interface directly with their fingers. The interface is designed for a single user, but in three out of four experimental conditions the interface is mobile.

The four experimental conditions were: *Fixed*, where the interface did not move, *Button*, where users touched projected buttons to send the interface between each other, *PDS*, where collaborating users moved the handheld cardboard display, and *Dialog* where the system directed the interface to the person who leads the collaboration, based on the temporal analysis of user's dialog. We compared the recall results, performance, and the number of turn-taking between users. We also asked subjects to reveal which condition they liked the best.

The results of our study show that, users prefer to collaborate using the mobile interface rather than the fixed one. Users, such as programmers, having strong habits in using conventional computers do not like the *Dialog* condition, in which the computer acts like a moderator, but they prefer the portable interface projected on the PDS, and the menu-based interface of *Button* condition. On the contrary, users having less

experience with information technology, were more disturbed by failures of the PDS tracking and prefer the *Dialog* and *Button* conditions.

Apart from the comparison of different modalities for controlling the location of an interface, we observed a number of usability issues related to the interaction with our system and with projected interfaces in general. We note that care should be taken to provide continuous transitions during interface movements, even when the displacement is small. We also observed that some users are cautious when using front-projected interactive widgets because they are afraid to trigger them accidentally with a hand.

The experiment proves the robustness of our projection-based implementation of both portable and steerable interfaces, and shows that mobile interfaces have the capability to facilitate collaboration between users.

Seventh Chapter

Conclusions

Motivated by the fast development of new interactive technologies leading the information technology towards Ubiquitous Computing, we define, develop, and test a new class of user interfaces, distinguished from conventional computing by spatial mobility.

In this chapter, we summarize our contributions. We outline our work and discuss its limitations. We also present directions for future work.

7.1 Contributions

This thesis covers a large spectrum of the design process of an interactive system. We started by designing and constructing the SCP, an actuator-sensor pair allowing to display interactive graphics on almost any surface in the environment, and finished by developing and evaluating applications exploiting interface mobility. As a result, the contributions are two-fold; on one hand they relate to the development of the technology for projection-based mobile interfaces, and on the other to the definition and analysis if steerable interfaces.

7.1.1 Technology related contributions

One way of making an environment interactive is to disperse the interaction resources. In future work environments, people will be free to interact with the computer system anywhere. To anticipate this trend, we need to create prototypes of interactive spaces which would allow to experiment with interface mobility. The design and implementation of such prototype is one of the major contributions of this work.

Our prototype was built from scratch, and involved both the choice of appropriate technology, and the integration of the technology with the computer system. Our objective was to develop a system allowing people to interact with a graphical interface anywhere in the environment. Out of the available technologies, we consider computer vision combined with steerable video projection the best adapted for that purpose.

We designed a Steerable Camera Projector pair, which is able to display interactive image in a hemisphere around the device.

While our SCP is offers similar functions as the Everywhere Display [67], it is an original design. In particular, we decided to mechanically couple the camera with the video-projector. The camera is mounted such that its field of view overlaps with the projection cone. As a result, it can be used to permanently observe user actions over the projection, and to detect interaction events. This mechanical configuration allowed also to create a projection-based Portable Display Surface.

While projection offers a number of advantages, such as the ability to augment mundane objects with digital data, it raises several technical issues. In particular, when displaying on planar surfaces at arbitrary angles the projected image is distorted. To cope with this issue, we adapt a pinhole projector model and calculate a pre-warp to rectify the projected image. Our approach for modeling the projector-screen configuration can be used for dynamic image correction, and to rectify images while the SCP is moved.

An other issue with steerable projectors is that by design they are not bound to the display surface, i.e. the projector is agnostic about the display surface. As a result, the computer system has to find adequate display surfaces in the environment. We propose an off-line automatic surface detection method exploiting the SCP's ability to display patterns. We also discuss our approach to environment modeling, in which we highlight the relation between the type of interactions supported by the system, and the necessary features of the environment model.

The SCP can be used not only to display mobile images on static surfaces, but also to project data on mobile surfaces. We use computer-vision to track a card-board made lightweight display surface, the PDS.

We present two versions of the PDS tracking algorithm. Our first implementation tracks edges of the PDS as points in a two dimensional Hough space. The second implementation tracks line segments in the image plane, where contours are filtered-out and approximated based on gradient directions. Both algorithms run at camera frame-rate, and are relatively robust to partial occlusions. However the second algorithm is more robust to clutter in the background, and can track simultaneously several quadrilateral objects.

The SCP together with the PDS provides a spatially flexible computer output, one of the two key elements of an interactive system. To render the images interactive we use computer vision to detect user actions over the projected image.

We identify interaction modalities that are suitable for use with mobile projected interfaces. The use of a particular modality depends on the context in which an application is deployed. In this study, we focus on direct interaction with projected images.

We present our implementation of vision-based interactive widgets. The widgets are composed of very simple occlusion detectors assembled to react only to objects having a particular form, e.g. fingertips. The widgets are robust to accidental full-

occlusions and, to some extent, to partial occlusion. Our method allows us to build interfaces based on basic interactive widgets such as buttons and sliders.

The software components of our interactive environment are implemented as services. Services are isolated, autonomous software that interoperate by exchanging messages using a communication protocol. Our definition of services for interactive systems is based on the W3C definition of web-services and service-oriented architecture.

Because our approach is user centered, we evaluate our hardware and software in terms of usability. We base the evaluation on the ISO 25000 standard and IFIP usability criteria. We derive non-functional requirements that apply to both software and interaction devices. Software is evaluated against qualitative requirements, like: autonomy or absence of tedious parameter setups and recoverability. Interaction devices should meet quantitative criteria such as: resolution, static stability and latency.

In our study, we define the set of services necessary to build interactive applications supporting interface mobility. We discuss certain implementation details, and provide a user-centric evaluation of their performance.

7.1.2 Contributions related to interaction with mobile interfaces

In this study, we identify a new class of mobile interfaces, that we call steerable interfaces. We consider steerable interfaces as a means for rendering spaces interactive. Interactive spaces are closely related to the notion of ubiquitous computing, which in turn draws on concepts such as virtual and augmented reality. To position our work on steerable interface with respect to VR and AR, we summarize the main ideas behind each of them, and present a simple classification space.

We note that, mobility is the common denominator for steerable and portable interfaces. Mobility applies to both physical entities such as people or interaction devices, and to virtual entities such as data or graphical objects. In this study, we are concerned with the mobility of an interface, an abstract figure build from physical entities, but not bound to them for its lifetime. In order to eliminate ambiguities, we provide definitions of: *(1)* an interface, *(2)* a steerable interface, and *(3)* a portable interface.

Though our definition of steerable interfaces is limited to neither projection-based displays nor to graphical interfaces, our steerable interfaces are based on graphical output.

We refer our definition of steerable interface to the definition of Pingali et al. [64]. We point-out the differences between the two approaches. In particular, our definition is not limited to interfaces implemented using a particular technology such as steerable projection.

We identify three properties that are fundamental for the description of steerable

interfaces: observability, controllability and non-preemptiveness. These properties form a classification space for steerable interfaces, and a conceptual framework helping to identify differences between steerable interfaces implementations. Though the properties are primary for the analysis of interaction with steerable interfaces, their presence is not a necessary condition to classify an interface as steerable.

We propose a reasoning framework for the design of interactions allowing to control the location of an interface. We consider locus of the interface as a resource similar to physical resources such as display screens of conventional work stations. This allows us to base our analysis of interactions on the framework of Barralon et al. [60] for coupling interaction resources.

We proposed a number of interaction techniques and control interfaces for moving a steerable interface. The interaction techniques that can be realized with bare fingers, or using dedicated instruments such a laser pointer or a PDS.

The application of the coupling framework to the proposed interaction techniques allowed us to both spot some shortcomings of our interaction techniques and to improve the coupling framework. Notably, we noticed that the coupling finite state automaton cannot be transitively closed, i.e. some transitions between states are irreversible.

In our study, we also discussed which application field can take the most advantage from interface steerability. Besides future working environments where steerable interfaces can offer a mobile peephole to localized information [14], we identify computer supported collaborative work as a potential field for steerable interface applications.

To illustrate how mobile interfaces can support collaborative work, we present a prototype application for authoring presentations. The ContAct application interface is projected on the PDS, and allows groups of people to compose a presentation by combining on-line acquisition of images, text and video with pre-recorded material. We designed the ContAct application as a proof of concept prototype, which allowed us to improve our projection-based technology and to prepare the ground for a user study comparing fixed, portable and steerable interfaces.

The results of our user study show that, people prefer to collaborate using a mobile interface over a fixed one. Users, such as programmers, having strong habits in using conventional computers do not like the implicit voice-based interface steering, in which the computer acts like a moderator. On the other hand, they like both the portable interface projected on the PDS, and the menu-based interface. Users having less experience with information technology were more disturbed by failures of the PDS tracking and prefer to control the location of the interface using voice or buttons.

Apart the comparison of different modalities for controlling the location of an interface, we observed a number of usability issues related to the interaction with our system and with projected interfaces in general. We note that care should be taken to provide continuous transitions during interface movements, even when the displacement is small. We also observed that some users are cautious when using front-projected interactive widgets because they are afraid to trigger them accidentally with a hand.

The experiment proved the robustness of our projection-based implementation of both portable and steerable interfaces, and shows that mobile interfaces have the capability to facilitate collaboration between users.

7.2 Limitations and directions

While our study shows that our system makes it possible to create usable applications with steerable and portable interfaces, the system works in the controlled conditions of our laboratory. There are a number of technical limitations that make it infeasible to deploy our system in the real-world. In this section, we present these limitations and suggest how they can be overcome. We also discuss future directions for steerable interface applications.

The SCP installed in our laboratory is our first prototype, and like all prototypes it has some weak points. Noise is a major issue when using the SCP. The mechanical structure is build from lightweight aluminum profiles, which resonate vibrations induced by the stepper motors. As a result movements of the SCP can be disturbing for users. Nevertheless, other implementations of steerable projectors, such as the Fluid Beam[1], remain a reasonable solution to create steerable interfaces.

The PDS application offers a projection-based portable display. It can be considered as an alternative for implementing electronic paper. However, to provide a believable illusion of object augmentation, the latency must be suppressed. Minimizing the latency can be achieved by using high frame rate cameras, and implementing the tracking on the GPU.

Another issue with the PDS is the presence of projection errors that are due to the simplifying assumption that the homography between the camera and the projector is constant. Alternatively, the tracking could take in account a geometrical model of the PDS, and instead of using homographies it could calculate a 3D transformation. However, this solution would require calibrating the camera and the projector optics.

Our second PDS tracking implementation can potentially track multiple PDSs. Similarly the SCP can display on several display surfaces simultaneously. However, currently our calibration services are not able to handle pre-warp matrices for multiple displays. The support for multi-surface interaction is a difficult software architecture problem, which was addressed in the IST GLOSS project.

Apart the "short term" improvements of our technology, there is a number of issues that influence the future of steerable projectors in general. Image rectification for mobile projectors is an issue that must be improved. Though the first steerable projector was introduced in 2001, steerable projectors remain laboratory prototypes. One of the reasons for such situation is the lack of graphical drivers capable to rectify the output of conventional operating systems. As result, projection-based steerable interfaces are limited to custom applications. Another major reason is that steerable

[1] http://www.fluidum.org/en/FluidBeam.shtml

interfaces are not able to detect surfaces suitable for projection "by them-self". The development of automatic environment modeling methods is a crucial step towards large scale deployment of steerable projectors.

In this study, we considered steerable interfaces isolated from other forms of interactive devices. However, mobile interfaces will cohabit with a myriad of interaction resources integrated with the environment or worn by users. Most probably, steerable interfaces will be considered as an extension for conventional and wearable computing. As a result, both the proposed interaction modeling and interaction techniques need to be adapted to more complex configurations of interaction resources.

Finally, there is a need to identify the "killer-application" for steerable interfaces. The colocated collaborative work and peephole interfaces to localized virtual space [14], are examples of application classes where mobile interface can be useful. However, it is difficult to foresee if there are other applications fields for steerable interfaces.

The current situation for steerable interfaces can be compared to the situation of mobile telephony 20 years ago. Everyone knew that mobile phones can be handy, but no-one anticipated the number and diversity of uses people have found for mobile phones. Similarly, until steerable interfaces become ubiquitous in the real world, we will not be able to assess their real value.

Our study is only the first step towards interactive spaces enabled with steerable interfaces. It provides the conceptual and technological founding for further studies on steerable interfaces in interactive environments.

Appendix A
IFIP usability properties

The International Federation of Information Processing (IFIP) is an "apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people" [34]. IFIP's Working Group 2.7(13.4) 'User Interface Engineering' (referred as 'WG 2.7') concentrates on the software problems of user interfaces. In 1997 IFIP published the 'Design Principles for Interactive Software', which is an attempt "to show developers of interactive systems how to make use of principles to ensure a high quality user interface." The scope of work involved is:

- to increase understanding of the development of interactive systems;
- to provide a framework for reasoning about interactive systems;
- to provide engineering models for their development.

To cope with these issues, software architecture models are referred to a set of external and internal properties of interactive software.

Because of our user-centric approach, in this study, we are mainly interested in the external properties. They are grouped into two categories concerning interaction flexibility (see table A.1) and interaction robustness (see table A.2).

Authors take care to keep the set of properties "as complete and mutually independent ('orthogonal') as possible." On the other hand, authors do not claim to achieve this endeavor and state that: "Rather than suggest that we could present a complete catalog of external properties to support usability, we consider the properties discussed to represent (some aspects of learning and user satisfaction apart) the current state-of-the-art." Though IFIP properties are not complete, we can use them as a basis for usability prediction and evaluation.

Flexiblility Property	Description	Related properties
Representation:		
Device multiplicity	More than one way to do something	Multi-media capability
Representation multiplicity	More than one way to present something	I/O multiplicity, equal opportunity, multi-modality
Input/Output re-use	History repeating itself	Use of defaults
Planning:		
Human role multiplicity	Several people doing several things	Access control
Multithreading	One person doing several things	Concurrency, interleaving
Non-preemtiveness	Doing what you want when you want	User-driven dialog, mixed-initiative dialog
Reachability	Getting anywhere from anywhere else	Commensurate effort
Adaptivity:		
Reconfigurability	The user changing the interaction	Programmability of the interface
Adaptivity	The system changing the interaction	Automatic macro construction
Migratability	Transferring control	

Table A.1: IFIP properties affecting interaction flexibility (after [34])

Robusteness Property	Description	Related properties
User dependent:		
Observability	The user may perceive	Immediacy, browsability, feedback, feedthrough
Insistence	The user will perceive	Salience, timeliness, persistence, awareness
Honesty	The user correctly comprehends	Affordance, familiarity, suggestiveness, guessability
Predictability	Understanding how the system will react	Observability, consistency, affordance, response time stability
User independent:		
Access control	Role-sensitive restriction of information availability	Human role multiplicity, feedthrough, awareness, visibility, privacy
Pace tolerance	Response times match user's expectations	Timeliness, adaptivity, migratability
Deviation tolerance	User's recovery intentions are supported	Forward/backward recoverability, commensurate effort, pre-emptiveness

Table A.2: IFIP properties affecting interaction robustness (after [34])

Appendix B
Steerable Camera-Projector

This document describes the design process and the assembly of the actual implementation of a Steerable Camera-Projector platform developed in PRIMA group at INRIA Rhône-Alpes. Rather than technical specification, we provide an overview of the design and "rule of thumb" advice that helped us to build the SCP. These can be considered as guidelines to those who are interested in building a similar device, but note that in section B.3 we present alternative approaches to the SCP design. These are based on "off the shelf" components, and therefore are much less time consuming.

B.1 The Steerable Camera-Projector pair - general description

Our system has been designed to enable experiments in which a room is used as an interactive display surface. In particular, our system must permit experiments in which the location of an interactive display surface is automatically adapted to the relative positions and orientations of users. To meet this challenge, we have constructed a Steerable Camera-Projector platform (SCP) that provides a video-projector with two mechanical degrees of freedom: pan and tilt.

The configuration of the main components of the SCP is shown in Figure B.1. The SCP's mechanical performance is presented in Table B.1. The SCP is composed of an XGA-resolution video-projector (Epson EMP-720), a computer controlled pan-tilt platform and a CCD, PAL video camera. In order to reuse the communication protocol and hardware interface from the pan-tilt cameras installed in our Augmented Meeting Environment laboratory, we constructed our system by cannibalizing a commercially available pan-tilt camera (Sony EVI-D31). The serial computer interface for the pan-tilt head was connected to high performance stepper motors, enabling rotation of loads up to 2.5kg approximately. The camera and lens assembly was detached from its pan-tilt base and rigidly fixed to the projector mounting. A mechanical assembly was

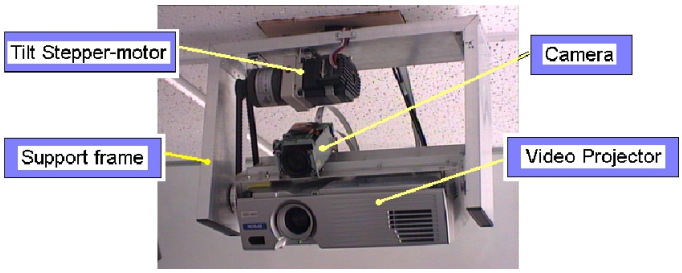

Figure B.1: Scheme of SCP

designed so that the SCP could be mounted on the ceiling in an office-like augmented environment.

Table B.1: Rotation platform mechanical performance

	Pan	Tilt
Rotation range	$\pm 177°$	$+90°$
Angular resolution	$0.205°$	$0.163°$
Angular velocity	$\sim 146 \frac{deg}{s}$	$\sim 80 \frac{deg}{s}$

B.2 Design Milestones

The design of the mechanical structure of an SCP is critical for a successful implementation. Any mistake would cost both money and time of the invested in assembling the SCP. The success of the design depends equally on the order of the design steps and the time taken to complete each of them. In the right order one should:

1. Specify the mechanical assembly layout, in particular design the articulations and mechanical-power transfer. Verify how does the SCP fit in the available space, how will it be fixed to the ceiling, how cables that go to the video projector should be passed to the rotating SCP head.

2. Chose a video projector - it should by lightweight, very bright (at least 1000 Lumens), with high contrast (at least 400:1) and (if possible) with auto-focus. To our knowledge there is no such video projector available on the market, so compromise is inevitable. We use the Epson EMP-720, but a reasonable alternative could be the PLUS V-1100Z. Both of them are small and lightweight but neither have auto-focus.

3. Estimate the total mass and mass moments of inertia of the moving parts.

4. Chose the gears and motors with adequate drivers and power supplies. Since one of the motors is mounted on the moving chassis, this and the previous step should be done iteratively.

5. Verify that all parts fit together - in other words repeat all steps of the design.

The rest of this paper presents in more detail each of the listed above milestones. The design of the SCP developed in PRIMA-group is used as a reference.

B.3 Alternative design approaches

Since our implementation of the SCP platform, both the projector and motor-gear assembly technologies have evolved. At the same time other research groups have been attracted to the idea of steerable projection. As a result, other SCP-like devices have been constructed using components that are more technologically advanced.

The SCP device developed at Technical University of Karlsruhe combines a Casio XJ-450 projector with a pan-tilt platform made by Motion Perception. The advantage of this approach is the use of "off the shelf" components. Additionally, the Casio projector offers 2500 lumen images, auto-focus and auto-keystoning correction. However, the auto-focus and keystoning correction rely on structured light techniques involving projection of visible patterns, and thus are not usable for dynamic image rectification and focusing.

Butz, Schneider and Spassova [15] present the FluidBeam. It is made by installing a video projector in a moving yoke designed for the steerable stage spot-light beamers. The commercially available yoke [1] allows movements of heavy projectors. The video projector of FluidBeam weighs 7.5 kg, which is almost three times the weight our SCP can handle. On the other hand FluidBeam is bigger than our SCP, and requires larger work volume.

The design presented by Butz et al. is very appealing, and unless there is a need for a highly customized device, we suggest implementing it rather than building an SCP from scratch.

B.4 Mechanical structure

Figure B.2a illustrates several different mechanisms with two rotational degrees of freedom. You have to remember, though, that the cost and complexity of your design will strongly depend on the power of the motors you will choose. The goal of the structure design is to minimize the necessary power of the motors. The design should also be simple and robust. To achieve these goals you should keep in mind that:

[1] The moving yoke is available at http://www.amptown-lichttechnik.de

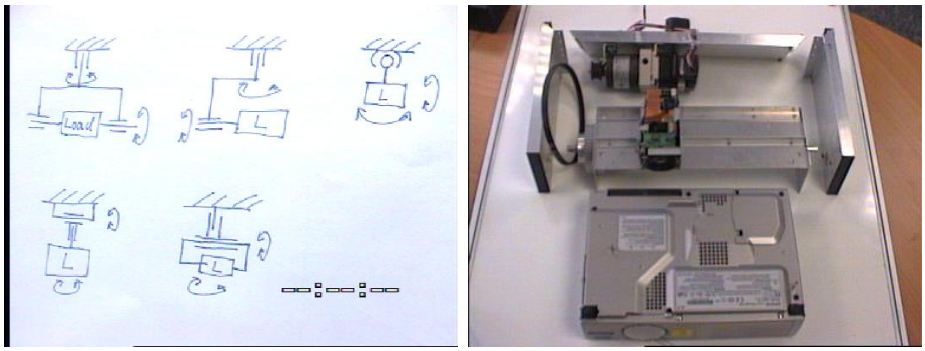

Figure B.2: All parts

- the structure has to be light and rigid,

- all heavy elements should be as close as possible to both rotation axes,

- parts should be compliant with some standard,

- use common materials like aluminum or epoxy-resins,

- the video projector bulb should be close to both rotation axes - accelerations reduce the lifetime of the bulb.

Unless you have experience in working with resins you should prefer aluminum as you principal material. It is relatively light and easy to process. Figure B.2b presents the structure of the moving aluminum-chassis of our SCP.

Video projector mount The beamer-mount is made from a 1.5mm aluminum plate (Figure B.3). It was straightened by riveting two equal-sided angle section aluminum profiles bought at a local "Castorama" (equivalent of an Obi supermarket). The mount was straightened in order to minimize elastic deformations which cause misalignment of the tilt axes, resulting in accelerated bearing degradation.

Another two plates attached perpendicularly to the beamer mount host holders of the axes. There are no twisting moments on the axes (except the negligible bearing friction moment). Therefore, the axe-holders can be very simple and lightweight. One axe is forced in the pulley of the belt gear (Figure B.4).

The other axe is mounted by interference in a holder made of two 2.5mm plates screwed from both sides to the 1.5 mm plate (Figure B.5).

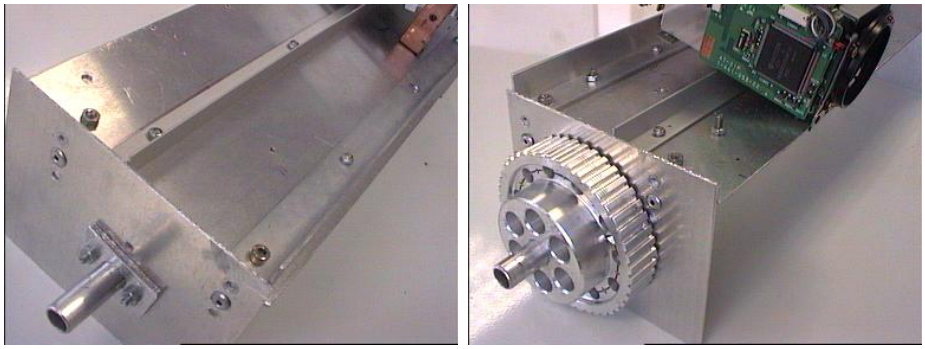

Figure B.3: Beamer mount

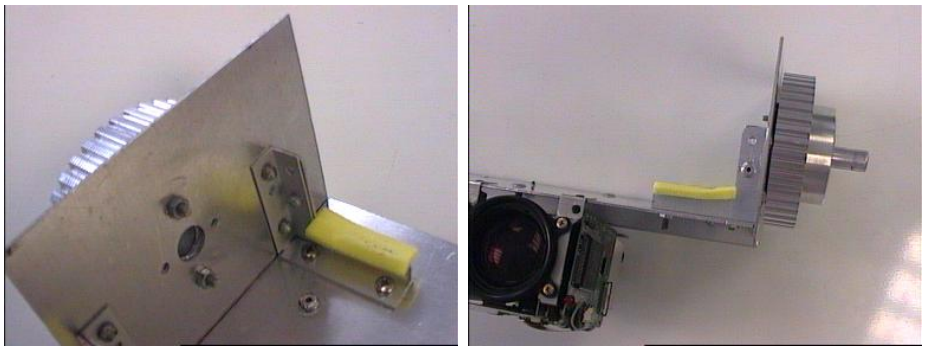

Figure B.4: Axe holder

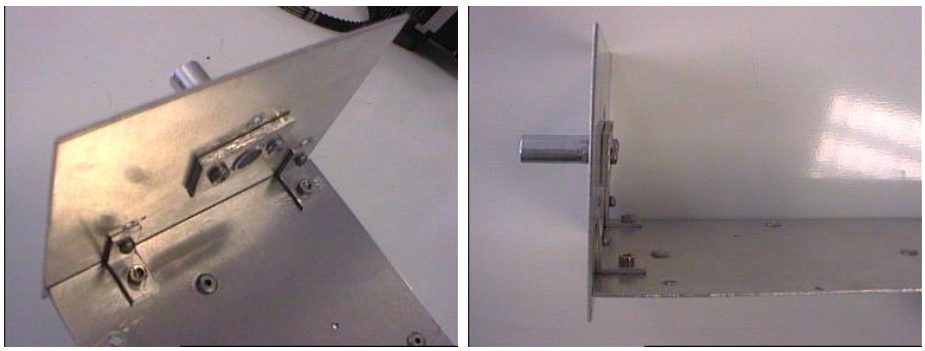

Figure B.5: Axe holder

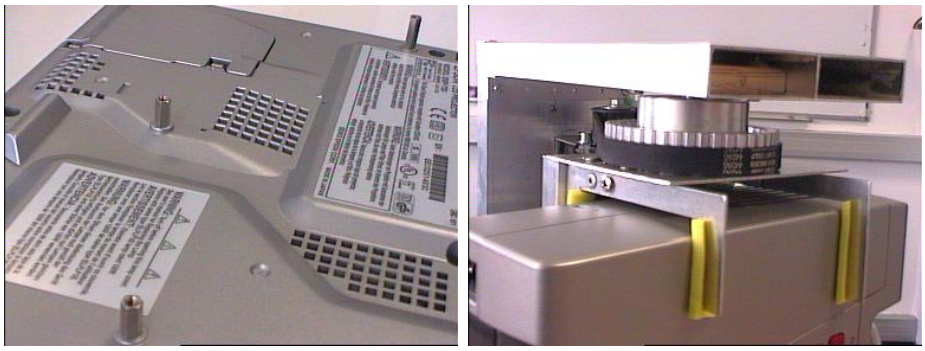

Figure B.6: Beamer installation

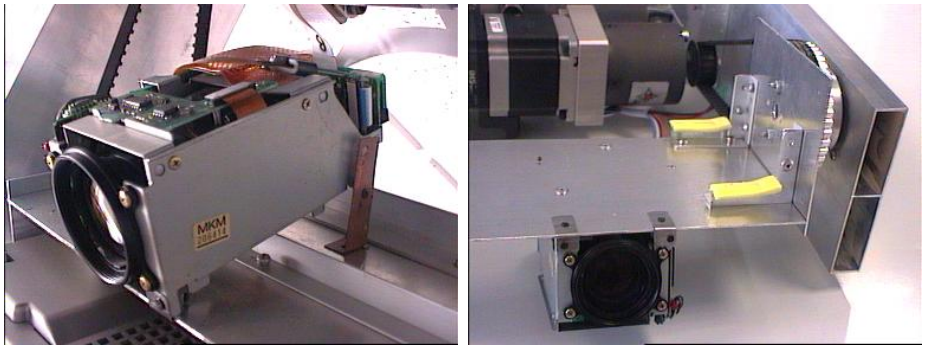

Figure B.7: Camera installation

The beamer is screwed to the mount in a standard way using the mount points provided by the beamer designer (Figure B.6a). Additionally, to reduce twisting of the beamer mount the beamer is fixed by two metal holders cut-out from a 1.5mm plate (Figure‘B.6b).

The camera-head is mounted on the SCP using thin metal stripes screwed with the original Sony-screws directly to the beamer mount (Figure B.7). Since the camera is relatively light and there are no important forces on its mounting the screw-threads where made in aluminum plate of the beamer mount.

The position of the camera on the SCP depends on your future application. It is wise to prepare two possible setups, one as close as possible to the objective of the beamer, the other one as far as possible.

Main frame The main frame of the chassis is made of elements cut out from a rectangular aluminum profile rule. This type of rule is used in masonry and should be available in any "Baumarkt" for less than 15 euros. The rule I bought was 1.5 meter long, 100mm wide and 18mm high. Its wall are about 1.5mm thick (Figure B.7b).

Figure B.8: Main frame elements connection

The elements of the SCP main frame can be separated (Figure B.8). The angle joints are riveted to the side segments of the main frame and screwed to the main beam of the SCP frame. The nuts of the fixing screws are glued to the frame using an epoxy raisin.

Figure B.8a shows also the front mounting of the tilt motor. Note the damping layer between the main frame structure and the motor bed. It serves to minimize the influence of the motor vibration on the rest of the SCP. The tilt motor together with its gear has two mounting points shown in Figure B.9.

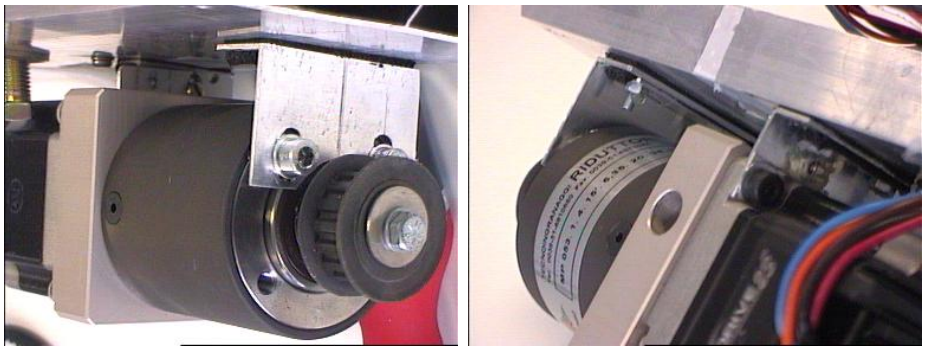

Figure B.9: Tilt motor installation

In case of our SCP the tilt motor drives a belt gear. Therefore the motor mounting is designed in a way that enables adjustment of the belt tension. Figure B.10 shows the belt gear and the motor front mounting with its oval screw-wholes.

Belt gears have several advantages; vibration damping, no backlash, silent work. On the other hand, belt gears require maintenance and in general are less precise than high quality wheel gears.

The main frame is connected to the pan motor via an element of a standard positive clutch (Figure B.11). The advantage of this solution is its simple assembly, but

Figure B.10: Belt gear adjustment

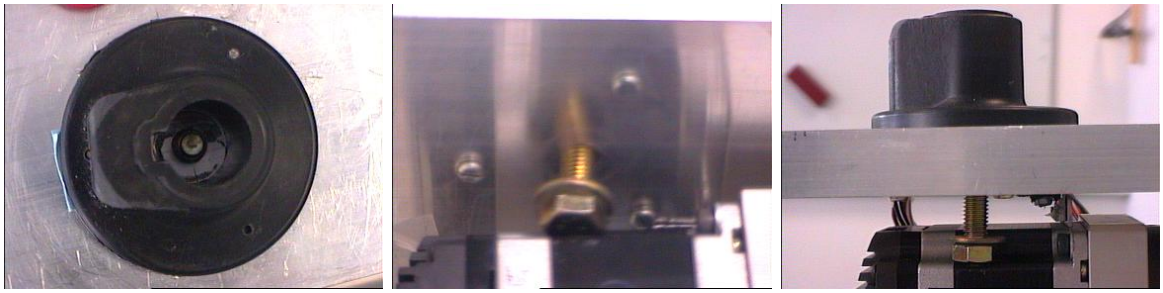

Figure B.11: Pan motor mount

mounting directly the main frame on the gear-head axis results in direct vibration transfer from the gear to the SCP structure. Note that for safety reasons the main frame is fixed to the pan motor axis using an M6 screw.

Mass estimation and articulated joints The articulated joints (bearing-axis couple) are elements with the highest failure risk in the SCP design. You have to estimate the maximal forces acting on the bearings. To do so, you have to estimate the mass and moments of inertia of the moving parts. The first step is to calculate the volume of all parts of your SCP. Then multiply the total volume by the density of the corresponding material. The density of aluminum can be approximated with $2.22g/cm^3$. Once you have calculated the mass of each part you can proceed with moments of inertia.

For each part estimate moment of inertia in respect to each rotation axis. Note that mass moment of inertia for a rigid body is defined as:

$$I = \int_m r^2 dm$$

where m is the mass of the body and r is the distance to the point of reference. For uniformly dense primitives you can find easy to use formulas. For example the

inertia moment of a cylinder in respect to its axis of symmetry can be calculates as: $I = 0.5Mr^2$, where M is the total mass of the cylinder and r is its radius.

Use Steiner's law to calculate inertia moments for a parallel rotation axis. It says that if the moment of inertia J_T of a body about a fixed axis passing through the mass center T is known, the moment of inertia J of the body about another fixed axis parallel to the former one can be determined as:

$$J = J_T + md^2,$$

where m is the mass of the body, and d is the separation of the axes.

Next step is to estimate the forces that act on the axes and bearings. If you do your design right, the bearing-load on the pan axis should be mostly due to the gravitation. However, take into account that if the pan rotation axis does not go through the center of gravity of the SCP, forces caused by the centripetal acceleration will occur. The tilt axis is also exposed to gravitation and eventual centrifugal forces. Moreover accelerations around pan axis create forces that act perpendicularly to the pan and tilt axis and are due to the inertia of the projector and its mount. The maximal dynamic force that is due to angular acceleration can be calculated as:

$$F = \frac{I \cdot \varepsilon_{max}}{s},$$

where I is the mass moment of inertia of the moving parts, ε_{max} is the maximum angular acceleration around pan axis and s is the span of the bearings on the tilt axis. Belt gears introduce an additional transverse force on the axes.

Noise, vibration and safety issues A big disadvantage of the aluminum structure is that it resonates easily. The stepping of the stepper-motors is the main source of vibration. Straight toothed wheel gears produce additional vibrations in a wide frequency range. Typically, when moving the SCP it will pass its resonant frequency which results in an unpleasant noise. Therefore, it is important to either isolate the aluminum structure from all vibration sources or modify the resonant frequency. Since it is difficult to evaluate the resonant frequency of the SCP before the prototype is actually build, one should rather focus on isolating the structure from vibrations and on minimizing the vibrations themselves. The best way of reducing vibrations is to use lightweight motors and gears.

Moving parts of a robotic device are potentially dangerous. Fortunately, this is less of an issue for the SCP as it is mounted on the ceiling and thus separated from users. As for the video projector, it is likely that during its exploitation the SCP will project light on people faces. Users are strongly discouraged from looking into the projector's lens.

B.5 Stepper motors and gears

Before choosing the motors, it is necessary to take a look at the gears and the control hardware of the Sony camera. Originally the EVI-D31 uses two PM35S-048 motors with 7.5° per full step, giving 48 steps per revolution. They are driven be a Toshiba TA8435H bipolar stepping motor driver, which is set to eight micro-steps per full step.

There are 1720 discreet positions in pan and 570 positions in tilt. To move to next position the CPU of the EVID-D31 sends to the stepper motors drivers eight rectangular wave pulses which corresponds to one full step of the motor.

Typically, industrial stepper motors have $1.8\frac{deg}{step}$. Assuming that the SCP has to rotate 360° in pan and 90° in tilt and that the motor driver can steer eight micro-steps per full step, the necessary gear ratios can be calculated as:

$$i_{tilt} = \frac{90^\circ}{570 \cdot 1.8^\circ} = \frac{1}{11.4} \qquad i_{pan} = \frac{360^\circ}{1720 \cdot 1.8^\circ} = \frac{1}{8.6}$$

The gear ratio is defined as the ratio of the output-axis (the load) revolutions to the corresponding input-axis (motor shaft) revolutions.

Naturally motors with different step angles and different driver settings are available and you can play with those parameters. The more micro-steps the motor does per camera position, the more fluent will be the movement of the SCP.

Torque estimation The performance of a stepper motor is characterized by torque vs. speed curve. Choosing a motor for an application consists in verifying if the required torque for all speeds does not exceed the torque vs. speed curve. In practice one should identify the worst working condition or conditions and verify if the motor is able to handle them.

Since the Sony EVI-D31 stepper motor control does not provide any acceleration ramping, the worst working condition occurs during the first phase of accelerating the motor from standstill to maximum camera speed. Knowing that the maximum response time of a motor cannot exceed 2 full steps, the required torque for a given rotation axis can be estimated as:

$$T = T_f + J_r \cdot \varepsilon = T_f + J_r \cdot \frac{V^2}{2} \frac{2\pi}{steps/rev}$$

where T_f is the frictional torque, J_r is the mass moment of inertia of the moving parts in respect to the motor shaft, V is the speed to which you accelerate and $steps/rev$ ratio describes the number of full steps per revolution of the motor shaft. If T_f is expressed in $Nm \cdot 10^{-3}$, J_r is in gm^2 and V is expressed in full steps per second the resulting torque is in $Nm \cdot 10^{-3}$. If the torque vs. speed curve is not defined for the given motor speed value V, the calculated torque corresponds roughly to the holding torque of the motor. The J_r moment equals $J_r = J_s + J \cdot i^2$, where the J_s is the mass inertia moment of the motor shaft, J is the sum of inertia moments of all moving parts (including the gear) in respect to the driven rotation axis and i is the corresponding

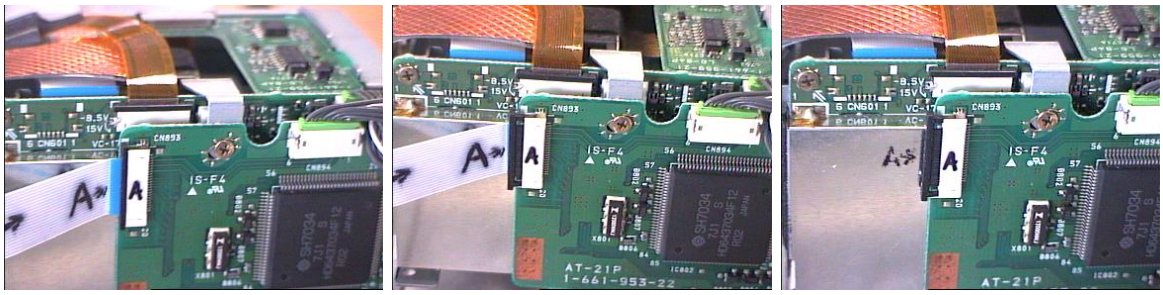

Figure B.12: PCB tape connector

gear ratio. A rule of thumb is to keep the J_r mass inertia moment reflected to the motor shaft no more than ten times greater than the mass moment of inertia of the motor shaft. The J_r can be modified by changing the gear ratio.

Once the motors start turning the torque necessary to keep the SCP at its maximal speed is only due to friction and therefore is relatively low. Note however, that some stepper motors have relatively low maximum speed and might not be able to satisfy your needs.

Some references for motors, gears and drivers A stepper motor is characterized by the step angle, driving method, physical dimensions and performance curves. Most of the high performance steppers have $1.8\frac{deg}{step}$. Many of them are compliant with industrial standards (like NEMA xx or ISO 900x), which enables the end-user to use any gear-head that is compliant with the same standard. Take a look at the offers of two companies that produce complete stepper motors solutions (motors, gear-heads, drivers, power supplies); Phytron (Phytron-Elektronik GmbH) and Intelligent Motion Systems (IMS Europe GmbH). Note that the control signals from the Sony hardware are the following: step pulses, enable and direction.

For belt gears solutions I have only two references; the Martin Sprocket & Gear, Inc. for pulleys and Gates Corporation for belts. I bought the Martin and Gates products from Radiospares, but I am sure you can buy them elsewhere.

B.6 Assembly

Taking apart the Sony camera does not cause any problem. Keep all screws since you might use them to assemble the SCP. Do not force the conductor tapes - the surface mounted connectors have a locking bar which you just have to release (Figure B.12). You will have to get the same type of tapes, but longer, in order to connect the camera head mounted on the SCP to the rest of the EVI-D31 hardware.

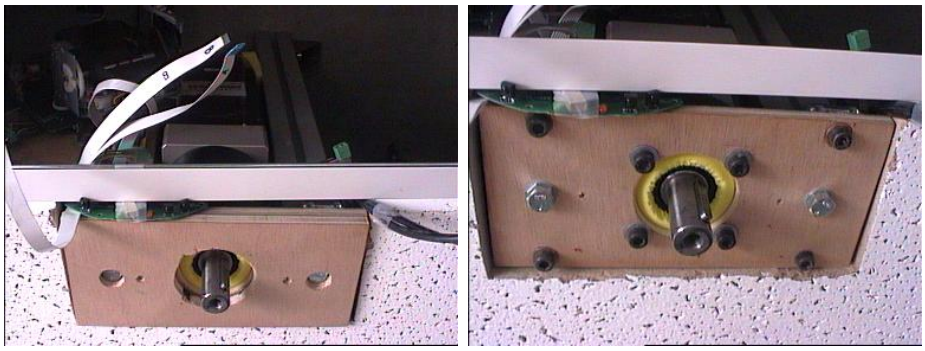

Figure B.13: Pan motor in the ceiling mounts

SCP initialization The Sony EVI-D31 has an initialization mechanism which enables it to find the reference position when powered-up. It is based on the IR position encoders which you find on the camera-head-mount. Without signals from those encoders the EVI-D31 hardware will freeze, therefore you have to either leave this mechanism as it is, or make a similar one on the SCP. Note that if you choose (like I did) not to make a replica of the position encoders, the SCP will work in an open-loop setup - without feedback about the current position.

Connecting the stepper motors to the EVI-D31 There are three signals that the Sony CPU sends to the TA8435H drivers to control the stepper motors: micro-step-pulse (pin 7 of the TA8435H), direction (pin 5) and enable (pin 3). Each of the micro-step-pulses corresponds to one micro step of the motor and is a rectangular waveform with high level signal width of $2 \div 4 \mu s$. The TA8435H drivers of the EVI-D31 are permanently set to eight pulses per full-step of the motor.

Direction signal has two logic levels each corresponding to one direction of rotation of the motor shaft. Enable signal is low ($0V$) while the motor should be enabled and high ($+5V$) at standstill.

The logic ground of the Sony hardware is on the first pin of the TA8435H. The $+5V$ power supply is on pin 13.

Ceiling mount Since in our augmented meeting office there is more than 40cm of space between the real ceiling and the visible ceiling-panels there was no problem to install our SCP (Figure B.13). Note that the ceiling mount has to be very solid not only to keep the SCP steady but also to ensure the safety of people standing below the SCP.

List of parts The list of parts (excluding rivets and screws) is listed in Table B.2.

Part name or provider reference	Part function and description	Producer	Provider
EVI-D31	Video camera and pan-tilt control hardware	Sony	Provideo Grenoble
FJH-20-D30-4	Connector tape for the control hardware and camera head	Samtec	Mateleco Alpes
EMP-720	Video projector	Epson	Provideo Grenoble
MD-2222-4	Tilt motor with integrated driver	IMS	Supradis Automatismes
MD-3447	Pan stepper motor+driver	IMS	Supradis
MD CC 100	Cable to interface the MD-series drivers	IMS	Supradis
IP 404-240	Power supply for the MD-2222	IMS	Supradis
IP 804-240	Power supply for the MD-3447	IMS	Supradis
MP 053 R1/4	Gear head for the MD-2222	Tecnoingranaggi Riduttori	Supradis
MP 080 R1/7	Gear head for the MD-3447	Tecnoingranaggi Riduttori	Supradis
109-845	Bearing for the tilt axis		Radiospares
350-8074	Pulley for the belt gear 16 teeth	Martin	Radiospares
350-8282	Pulley for the belt gear 44 teeth	Martin	Radiospares
338-7899	180 XL 037 belt	Gates	Radiospares
676-859	Clutch element to couple main frame with pan gear-head		Radiospares
40x20x1.5 plate	Aluminum plate for beamer mount		Radiospares
40x20x2.5 plate	Aluminum plate for tilt axis holder		Radiospares
1.5m Rule	1.5m aluminum, rectangular profile for main frame construction		Castorama
15x1.5 angle profile	equal-sided angle section aluminum profile used for joint elements		Castorama
10x1 angle profile	equal-sided angle section aluminum profile for beamer mount straightening		Castorama
circle profile	10mm diameter, 1mm thick aluminum pipe for tilt axis		Castorama

Table B.2: List of parts of the SCP

Bibliography

[1] ISO/IEC 25000. Software engineering – software product quality requirements and evaluation (square) – guide to square. In *ISO/IEC 25000:2005 standard*, 2005.

[2] G. Apitz and F. Guimbretière. Crossy: a crossing-based drawing application. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 3–12, 2004.

[3] M. Armstrong and A. Zisserman. Robust object tracking. In *Proceedings of the Asian Conference on Computer Vision*, volume I, pages 58–61, 1995.

[4] M. Baker. Human factors in virtual environments for the visual analysis of scientific data, 1995.

[5] M. Beigl, H.-W. Gellersen, and A. Schmidt. Mediacups: experience with design and use of computer-augmented everyday artefacts. *Comput. Networks*, 35(4):401–409, 2001.

[6] F. Bérard. *Vision par ordinateur pour linteraction homme-machine fortement couple*. PhD thesis, Universite Joseph Fourier Grenoble I, 1999.

[7] F. Bérard. The magic table: Computer-vision based augmentation of a whiteboard for creative meetings. In *Proceedings of the ICCV Workshop on Projector-Camera Systems*. IEEE Computer Society Press, 2003.

[8] S. Berger, R. Kjeldsen, Ch. Narayanaswami, C. Pinhanez, M. Podlaseck, and M. Raghunath. Using symbiotic displays to view sensitive information in public. In *Third IEEE International Conference on Pervasive Computing and Communications (PERCOM'05)*, pages 139–148, Kauai Island, Hawaii, USA, 2005. IEEE Computer Society.

[9] H. Beyer and K. Holtzblatt. *Contextual design: defining customer-centered systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[10] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, NY, USA, 1993. ACM Press.

[11] S. Borkowski, J. L. Crowley, J. Letessier, and F. Bérard. User-centric design of a vision system for interactive applications. In *ICVS*, page 9. IEEE Computer Society, 2006.

[12] S. Borkowski, J. Letessier, and J. L. Crowley. Spatial control of interactive surfaces in an augmented environment. In R. Bastide, P. A. Palanque, and J. Roth, editors, *EHCI/DS-VIS*, volume 3425 of *Lecture Notes in Computer Science*, pages 228–244. Springer, 2004.

[13] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. A. Shafer. Easyliving: Technologies for intelligent environments. In *HUC '00: Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, pages 12–29, London, UK, 2000. Springer-Verlag.

[14] A. Butz and A. Krüger. A mixed reality room following the generalized peephole metaphor. *IEEE Comput. Graph. Appl.*, 26(1):56–63, 2006.

[15] A. Butz, M. Schneider, and M. Spassova. Searchlight - a lightweight search function for pervasive environments. In A. Ferscha and F. Mattern, editors, *Pervasive*, volume 3001 of *Lecture Notes in Computer Science*, pages 351–356. Springer, 2004.

[16] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[17] T. Cham, R. Sukthankar, J. Rehg, and G. Sukthankar. Shadow elimination and occluder light suppression for multiprojector displays. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2002.

[18] O. Chapuis and N. Roussel. Metisse is not a 3d desktop! In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 13–22, New York, NY, USA, 2005. ACM Press.

[19] Lawerence Chung. Representation and utilization of non-functional requirements for information system design. In *CAiSE '91: Proceedings of the third international conference on Advanced information systems engineering*, pages 5–30, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[20] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, 2003.

[21] J. Coutaz, S. Borkowski, and N. Barralon. Coupling interaction resources: an analytical model. In *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 183–188, New York, NY, USA, 2005. ACM Press.

[22] J. L. Crowley, J. H. Piater, M. Vincze, and L. Paletta, editors. *Computer Vision Systems, Third International Conference, ICVS 2003, Graz, Austria, April 1-3, 2003, Proceedings*, volume 2626 of *Lecture Notes in Computer Science*. Springer, 2003.

[23] C. Cruz-Neira, A. Bierbaum, P. Hartling, C. Just, and K. Meinert. Vr juggler - an open source platform for virtual reality applications. In *AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, USA, 2002.

[24] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The cave: audio visual experience automatic virtual environment. *Commun. ACM*, 35(6):64–72, 1992.

[25] J. Davis and X. Chen. Lumipoint: Multi-user laser-based interaction on large tiled displays. *Displays*, 23(5), 2002.

[26] P. Dillenbourg, M. Baker, A. Blaye, and C. O'Malley. The evolution of research on collaborative learning. *Learning in Humans and Machine: Towards an interdisciplinary learning science*, pages 189–211, 1996.

[27] E. Dubois and L. Nigay. Augmented reality: which augmentation for which reality? In *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*, pages 165–166, New York, NY, USA, 2000. ACM Press.

[28] J. A. Fails and D. Olsen Jr. Light widgets: interacting in every-day spaces. In *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 63–69, New York, NY, USA, 2002. ACM Press.

[29] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[30] S. S. Fisher, M. McGreevy, J. Humphries, and W. Robinett. Virtual environment display system. In *SI3D '86: Proceedings of the 1986 workshop on Interactive 3D graphics*, pages 77–87, New York, NY, USA, 1987. ACM Press.

[31] G. W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Commun. ACM*, 36(7):39–49, 1993.

[32] M. Fjeld, F. Voorhorst, M. Bichsel, K. Lauche, M. Rauterberg, and H. Krueger. Exploring brick-based navigation and composition in an augmented reality. *Lecture Notes in Computer Science*, 1707:102–116, 1999.

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

[34] Ch. Gram and G. Cockton, editors. *Design principles for interactive software*. Chapman & Hall, Ltd., London, UK, UK, 1997.

[35] J. Greenbaum and M. Kyng, editors. *Design at work: cooperative design of computer systems*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1992.

[36] T. Grossman, D. Wigdor, and R. Balakrishnan. Multi-finger gestural interaction with 3d volumetric displays. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 61–70, New York, NY, USA, 2004. ACM Press.

[37] D. Holman, R. Vertegaal, M. Altosaar, N. Troje, and D. Johns. Paper windows: interaction techniques for digital paper. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 591–599, New York, NY, USA, 2005. ACM Press.

[38] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *UbiComp '01: Proceedings of the 3rd international conference on Ubiquitous Computing*, pages 116–122, London, UK, 2001. Springer-Verlag.

[39] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Trans. Graph.*, 21(3):693–702, 2002.

[40] Yi-P. Hungy, Y.-S. Yangz, Y.-Sh. Cheny, I.-B. Hsiehz, and Ch. Fuhz. Free-hand pointer by use of an active stereo vision system. In *Proceedings of the 14th International Conference on Pattern Recognition (ICPR'98)*, volume 2, pages 1244–1246, August 1998.

[41] J.Coutaz, C.Lachenal, and S. Dupuy-Chessa. Ontology for multi-surface interaction. In *Proceedings of the ninth International Conference on Human-Computer Interaction (Interact'2003)*, 2003.

[42] B. Johanson, G. Hutchins, T. Winograd, and M. Stone. Pointright: Experience with flexible input redirection in interactive workspaces. *Proceedings of UIST-2002*, 2002.

[43] D. R. Olsen Jr and T. Nielsen. Laser pointer interaction. In *ACM CHI'2001 Conference Proceedings: Human Factors in Computing Systems. Seattle, WA*, 2001.

[44] F. Jurie. Tracking objects with a recognition algorithm. *PRL*, 19(3-4):331–340, March 1998.

[45] R. Kjeldsen and J. Kender. Finding skin in color images. In *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, page 312, Washington, DC, USA, 1996. IEEE Computer Society.

[46] R. Kjeldsen, A. Levas, and C. S. Pinhanez. Dynamically reconfigurable vision-based user interfaces. In Crowley et al. [22], pages 323–332.

[47] R. Kjeldsen, C. Pinhanez, G. Pingali, J. Hartman, T. Levas, and M. Podlaseck. Interacting with steerable projected displays. In *FGR '02: Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, page 402, Washington, DC, USA, 2002. IEEE Computer Society.

[48] S. R. Klemmer, J. Li, J. Lin, and J. A. Landay. Papier-mâché: toolkit support for tangible input. In *Proceedings of the 2004 conference on Human factors in computing systems (CHI'04)*, pages 399–406. ACM Press, 2004.

[49] J. C. Lee, S. E. Hudson, J. W. Summet, and P. H. Dietz. Moveable interactive projected displays using projector based tracking. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 63–72, New York, NY, USA, 2005. ACM Press.

[50] J. Letessier and F. Bérard. Visual tracking of bare fingers for interactive surfaces. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 119–122. ACM Press, 2004.

[51] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[52] A. Lux. The imalab method for vision systems. *Machine Vision and Applications*, 2004.

[53] S. Malik and J. Laszlo. Visual touchpad: a two-handed gestural input device. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 289–296, New York, NY, USA, 2004. ACM Press.

[54] S. Malik, A. Ranjan, and R. Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 43–52, New York, NY, USA, 2005. ACM Press.

[55] R. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2d-3d model-based approach. In *Proceedings of International Conference on Computer Vision*, pages 262–268, September 1999.

[56] B. A. Meyers, R. Bhatnagar, J. Nichols, C.H. Peck, D. Kong, R. Miller, and A.C. Long. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*. ACM Press New York, NY, USA, April 2002.

[57] D. Molyneaux and G. Kortuem. Ubiquitous displays in dynamic environments: Issues and opportunities. In *Proceedings of Workshop on Ubiquitous Display Environments*, Nottingham, England, September 2004. review, display technology.

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

[59] N. Nakamura and R. Hiraike. Active projector: Image correction for moving image over uneven screens. In *Companion of the 15th Annual ACM Symposium on User Interface Software and Technology*, pages 1–2, October 2002.

[60] N.Barralon, C.Lachenal, and J.Coutaz. Couplage de ressources d'interaction. In *IHM'04*, pages 13–20, 2004.

[61] K. Nickel and R. Stiefelhagen. Pointing gesture recognition based on 3d-tracking of face, hands and head orientation. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 140–146, New York, NY, USA, 2003. ACM Press. 20-30 degree precision.

[62] Federal Ministry of the Interior. *Planning and Realization of IT-Projects. V-Model: Software Lifecycle Process Model*. KBSt, San Francisco, CA, USA, 1992.

[63] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Comput. Graph. Appl.*, 22(6):64–71, 2002.

[64] G. Pingali, C. Pinhanez, A. Levas, R. Kjeldsen, M. Podlaseck, H. Chen, and N. Sukaviriya. Steerable interfaces for pervasive computing spaces. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications - PerCom'03*, March 2003.

[65] G. Pingali, C. Pinhanez, T. Levas, R. Kjeldsen, and M. Podlaseck. User-following displays. In *Proceedings of the IEEE International Conference on Multimedia and Expo 2002 (ICME'02)*, 2002.

[66] G. Pingali and N. Sukaviriya. Augmented collaborative spaces. In *ETP '03: Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence*, pages 13–20, New York, NY, USA, 2003. ACM Press.

[67] C. Pinhanez. The everywhere displays projector: A device to create ubiquitous graphical interfaces. In *Proceedings of Ubiquitous Computing 2001 Conference*, September 2001.

[68] C. S. Pinhanez and M. Podlaseck. To frame or not to frame: The role and design of frameless displays in ubiquitous applications. In M. Beigl, S. S. Intille, J. Rekimoto, and H. Tokuda, editors, *Ubicomp*, volume 3660 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2005.

[69] M. Podlaseck, C. Pinhanez, N. Alvarado, M. Chan, and E. Dejesus. On interfaces projected onto real-world objects. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 802–803, New York, NY, USA, 2003. ACM Press.

[70] I. Poupyrev, T. Ichikawa, S. Weghorst, and M. Billinghurst. Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques. *Computer Graphics Forum, EUROGRAPHICS*, 17(3), 1998.

[71] R. Raskar, P. Beardsley, J. van Baar, Y. Wang, P. Dietz, J. Lee, D. Leigh, and T. Willwacher. Rfig lamps: interacting with a self-describing world via photosensing wireless tags and projectors. *ACM Trans. Graph.*, 23(3):406–415, 2004.

[72] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. iLamps: geometrically aware and self-configuring projectors. *ACM Trans. Graph.*, 22(3):809–818, 2003.

[73] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the ACM SIGGRAPH'98 Conference*, 1998.

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

[75] Philippe Renevier. *Systèmes Mixtes Collaboratifs sur Supports Mobiles: Conception et Réalisation*. PhD thesis, Universite Joseph Fourier Grenoble I, 2004.

[76] T. Richardson, F. Bennett, G. Mapp, and A. Hopper. Teleporting in an X window system environment. Technical report, IEEE Personal Communications, 1993.

[77] H. Roeber, J. Bacus, and C. Tomasi. Typing in thin air: the canesta projection keyboard - a new method of interaction with electronic devices. In *CHI '03 extended abstracts on Human factors in computing systems*, pages 712–713. ACM Press, 2003.

[78] N. Rosa, P. Cunha, and G. Justo. Processnfl: A language for describing non-functional properties. In *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9*, page 282.2, Washington, DC, USA, 2002. IEEE Computer Society.

[79] W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of IEEE WESCON*, pages 1–9. TRW, 1970.

[80] D. Schuler and A. Namioka, editors. *Participatory Design: Principles and Practices*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, 1993.

[81] J. Stewart, B. B. Bederson, and A. Druin. Single display groupware: a model for co-present collaboration. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 286–293, New York, NY, USA, 1999. ACM Press.

[82] R. Stoakley, M. J. Conway, and R. Pausch. Virtual reality on a WIM: Interactive worlds in miniature. In *Proceedings CHI'95*, 1995.

[83] N. A. Streitz, J. Geiler, T. Holmer, S. Konomi, C. Mller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-land: An interactive landscape for creativitiy and innovation. *ACM Conference on Human Factors in Computing Systems*, 1999.

[84] N. A. Streitz, C. Rcker, Th. Prante, R. Stenzel, and D. van Alphen. Situated interaction with ambient information: Facilitating awareness and communication in ubiquitous work environments. In *Tenth International Conference on Human-Computer Interaction*, June 2003.

[85] M. Sugimoto, K. Hosoi, and H. Hashizume. Caretta: a system for supporting face-to-face collaboration by integrating personal and shared spaces. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, New York, NY, USA, 2004. ACM Press.

[86] N. Sukaviriya, M. Podlaseck, R. Kjeldsen, A. Levas, G. Pingali, and C. Pinhanez. Augmenting a retail environment using steerable interactive displays. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 978–979, New York, NY, USA, 2003. ACM Press.

[87] P. N. Sukaviriya, M. Podlaseck, R. Kjeldsen, A. Levas, G. Pingali, and C. S. Pinhanez. Embedding interactions in a retail store environment: The design and lessons learned. In M. Rauterberg, M. Menozzi, and J. Wesson, editors, *INTERACT*. IOS Press, 2003.

[88] R. Sukthankar, R. Stockton, and M. Mullin. Smarter presentations: Exploiting homography in camera-projector systems. In *Proceedings of International Conference on Computer Vision*, 2001.

[89] Zs. Szalavri and M. Gervautz. The personal interaction panel - a two-handed interface for augmented reality. In *Proceedings of EUROGRAPHICS'97, Budapest, Hungary*, September 1997.

[90] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211–244, 2001.

[91] Y. Tokuda, S. Iwasaki, Y. Sato, Y. Nakanishi, and H. Koike. Ubiquitous display for dynamically changing environment. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 976–977, New York, NY, USA, 2003. ACM Press. model, environment model, laser range sensor.

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

[93] J. Underkofflerand B. Ullmer and H. Ishii. Emancipated pixels: Real-world graphics in the luminous room. In *Proceedings of ACM SIGGRAPH*, pages 385–392, 1999.

[94] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[95] V. Vlahakis, J. Karigiannis, M. Tsotros, M. Gounaris, L. Almeida, D. Stricker, T. Gleue, I. T. Christou, R. Carlucci, and N. Ioannidis. Archeoguide: first results of an augmented reality, mobile computing system in cultural heritage sites. In *VAST '01: Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage*, pages 131–140, New York, NY, USA, 2001. ACM Press.

[96] D. Vogel and R. Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *uist '05: proceedings of the 18th annual acm symposium on user interface software and technology*, pages 33–42, new york, ny, usa, 2005. acm press.

[97] W. Vogels. Web services are not distributed objects. *Internet computing*, 7(6):59–66, 2003.

[98] S. Voida, M. Podlaseck, R. Kjeldsen, and C. Pinhanez. A study on the manipulation of 2d objects in a projector/camera-based augmented reality environment. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 611–620, New York, NY, USA, 2005. ACM Press.

[99] W3C. Web services architecture. In *W3C Working Group Note*, February 2004.

[100] C. Ware and R. Balakrishnan. Reaching for objects in vr displays: lag and frame rate. *ACM Trans. Comput.-Hum. Interact.*, 1(4):331–356, 1994.

[101] Mark Weiser. The computer for the twenty-first century. *Scientific American*, 256(3):94 – 104, September 1991.

[102] P. Wellner. The digitaldesk calculator: Tactile manipulation on a desk top display. In *ACM Symposium on User Interface Software and Technology*, pages 27–33, 1991.

[103] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *Proceedings of Int'l Conf. Computer Vision*, pages 353–360, 2003.

[104] Andrew D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76, New York, NY, USA, 2004. ACM Press.

[105] Andrew D. Wilson. Playanywhere: a compact interactive tabletop projection-vision system. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 83–92, New York, NY, USA, 2005. ACM Press.

[106] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):780–785, 1997.

[107] J. Wu, T. C. N. Graham, and P. W. Smith. A study of collaboration in software design. In *ISESE '03: Proceedings of the 2003 International Symposium on Empirical Software Engineering*, page 304, Washington, DC, USA, 2003. IEEE Computer Society.

[108] Y. Wu and T. Huang. Color tracking by transductive learning. In *Proc. of IEEE Conf. on CVPR'2000, Vol.I, Hilton Head Island, SC*, pages 133–138, 2000.

[109] R. Yang and G. Welch. Automatic and continuous projector display surface calibration using every-day imagery. In *Proc. 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (CECG'01)*, 2001.

[110] G. Ye, J. J. Corso, D. Burschka, and G. D. Hager. Vics: A modular vision-based hci framework. In Crowley et al. [22], pages 257–267.

[111] 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.

[112] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer. Visual panel: virtual mouse, keyboard and 3d controller with an ordinary piece of paper. In *Proceedings of the 2001 workshop on Percetive user interfaces*, pages 1–8. ACM Press, 2001.

[113] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.