

University of Konstanz  
Department of Computer and Information Science

Master Thesis for the degree  
Master of Science (M.Sc.) in Information Engineering

**Bimanual Pointing Techniques for  
Cross-Display Interaction**

by

**Simon Föh**

(600006)

1<sup>st</sup> Referee: Prof. Dr. Harald Reiterer

2<sup>nd</sup> Referee: Prof. Dr. Marc H. Scholl

Date: November 2011



## Abstract

Multi-display environments can nowadays be found in meeting rooms, control rooms and other locations. Common multi-display environments provide limited input capabilities. Often, some important requirements such as mobility, privacy, awareness or similar cannot be granted. Based on a set of specific tasks for collaborative work, this thesis analyzes several aspects of cross-display interaction. In order to provide the most natural and efficient way of interacting with such an environment we propose bimanual gesture interaction. Using the two-handed approach does not only support cross-display object movement, but also manipulation of objects and more.

We introduce *HyPoba Pointing*, a novel pointing technique for bimanual cross-display interaction. This technique is designed for the special requirements of collaborative multi-display environments. Pointing gestures allow the control of one cursor for each hand and static gestures trigger various actions. *MultiDragger*, a tool which is designed to support bimanual pointing gestures in multi-display environments allows us to test *HyPoba Pointing*.

During an experiment with 14 participants, *MultiDragger* was tested using *HyPoba Pointing* and *Absolute Pointing*. The experiment contained two tasks. A single-handed drag & drop task and a bimanual manipulation task. The results show significant faster task completion time for *Absolute Pointing* and significant better accuracy for *HyPoba Pointing*. Whatever technique is preferred by the user, the development of *MultiDragger* has shown the possibility of bimanual gesture interaction for multi-display environments.

## Zusammenfassung

Multi-Display Systeme finden Anwendung in den verschiedensten Bereichen. In Konferenz- oder Kontrollräumen werden solche Systeme bereits erfolgreich eingesetzt. Oft bieten aktuelle Systeme nur eingeschränkte Interaktionsmöglichkeiten. Dies hat zur Folge, dass wichtige Anforderungen wie Mobilität, Datenschutz, die Wahrnehmung anderer Mitarbeiter oder ähnliches nicht vollständig unterstützt werden. Diese Masterarbeit untersucht die unterschiedlichen Aspekte und Interaktionsmöglichkeiten welche aktuelle Systeme bieten. Speziell unter der Berücksichtigung der Anforderungen für kollaboratives Arbeiten. Nachforschungen ergaben, dass bimanuelle Interaktion dank ihrer Natürlichkeit und Effizienz gut für solche Aufgaben geeignet sein könnte. Dieser Ansatz ermöglicht nicht nur das Verschieben von Objekten zwischen mehreren Bildschirmen, er bietet auch weitere Interaktionsmöglichkeiten wie zum Beispiel das Vergrössern von Objekten.

*HyPoba Pointing*, eine neue Zeigetechnik für die bimanuelle Interaktion mit Multi-Display Systemen wird in dieser Masterarbeit vorgestellt. Diese Technik wurde mit spezieller Berücksichtigung auf kollaboratives Arbeiten mit Multi-Display Systemen entwickelt. Die Zeigetechnik ermöglicht es dem Benutzer mit seinen Händen gleichzeitig zwei Zeiger zu steuern, einen für jede Hand. Zusätzlich lösen Handgesten unterschiedliche Aktionen aus. *MultiDragger*, ein neues Tool welches eigens dafür entwickelt wurde um mittels Gestensteuerung Objekte zwischen Displays verschieben zu können, wurde zum Testen eingesetzt.

In einem Experiment wurde *MultiDragger* mit *HyPoba Pointing* und absolutem Pointing getestet. Dabei wurden zwei unterschiedliche Tasks ausgeführt. Ein einhändiger drag & drop Task zum verschieben von Objekten zwischen mehreren Displays und ein zweihändiger Task zur Manipulation von Objekten. Die Ergebnisse zeigen eine signifikant schnellere Interaktion beim absoluten Pointing. *HyPoba Pointing* ermöglicht hingegen signifikant genauere Interaktion. Abgesehen davon, welche Zeigetechnik ein Benutzer bevorzugt, konnte das Experiment mit *MultiDragger* zeigen, dass bimanuelle Gestensteuerung für Multi-Display Systeme eingesetzt werden kann.



## Conventions

Throughout this thesis the following conventions are used:

- Unidentified third persons are always described in male form. This is only done for the purpose of readability.
- The plural “we” will be used instead of the singular “I”, even when referring to work that was primarily or solely done by the author. However many decisions have been discussed with members of the Human-Computer Interaction group at University of Konstanz.
- Links to websites or homepages of mentioned products, applications or documents are shown in a footnote at the bottom of the corresponding page. All links have been accessible on 1. November 2011, if no other date is supplied.
- The definition of human-computer interaction is used as a synonym for human-machine interaction as well as man-machine interaction.
- References in the label of images or tables refer to the origin of the image or table data.
- A set of videos of the experiment are added to the attached DVD
- The seminar- and project-paper which are referenced in this thesis can be found on the attached DVD



# Contents

<b>List of Abbreviations</b>	<b>11</b>
<b>List of Figures</b>	<b>13</b>
<b>List of Tables</b>	<b>17</b>
<b>List of Listings</b>	<b>18</b>
<b>1. Introduction</b>	<b>1</b>
1.1. What is this thesis about? . . . . .	2
1.2. Outline . . . . .	2
<b>2. Requirement Analysis</b>	<b>5</b>
2.1. Collaborative Multi-Display Environment . . . . .	5
2.1.1. Tasks for Collaborative Work . . . . .	6
2.1.2. Requirements for interacting with collaborative MDEs . . . . .	8
2.1.3. Related Work in Cross-Display Interaction . . . . .	10
2.2. Hand Gesture Interaction . . . . .	15
2.2.1. The Human Hand . . . . .	16
2.2.2. Gesture Types . . . . .	17
2.2.3. Bimanual Gestures . . . . .	18
2.3. Pointing Interaction . . . . .	21
2.3.1. Pointing Devices . . . . .	21
2.3.2. Distant Pointing Devices . . . . .	24
2.3.3. Distant Pointing Techniques . . . . .	25
2.4. Summary . . . . .	32
<b>3. <sup>2</sup>Hands<sub>4</sub>Displays</b>	<b>35</b>
3.1. Project Outline . . . . .	35
3.2. Gesture Tracking . . . . .	37
3.3. Optitrack . . . . .	40
3.3.1. Visual Tracking . . . . .	40
3.3.2. Installation . . . . .	41
3.3.3. Software . . . . .	42
3.3.4. TrackingTools Issues . . . . .	44

---

3.4. Data handling . . . . .	45
3.4.1. Data Fetching . . . . .	46
3.4.2. Virtual Objects . . . . .	47
3.4.3. Filters . . . . .	48
3.5. Multidragger . . . . .	53
3.5.1. Data Interface . . . . .	55
3.5.2. Landscape Arrangement . . . . .	56
3.5.3. Cross-Display Object Movement . . . . .	57
3.5.4. Datarecording . . . . .	58
3.6. Position-based Hybrid Pointing . . . . .	60
3.6.1. Velocity Functions . . . . .	62
3.6.2. Calculation Algorithms . . . . .	65
3.7. Summary . . . . .	70
<b>4. Experiment</b>	<b>73</b>
4.1. Goal of the Evaluation . . . . .	73
4.2. Evaluation Setting . . . . .	74
4.2.1. Participants . . . . .	74
4.2.2. Tasks . . . . .	75
4.2.3. Pre-Test . . . . .	78
4.2.4. Experimental Design . . . . .	79
4.3. Results . . . . .	81
4.3.1. Overall Analysis . . . . .	81
4.3.2. Per User Analysis . . . . .	86
4.3.3. Object Size Analysis . . . . .	89
4.3.4. Gender Analysis . . . . .	94
4.3.5. Interpretation of the Results . . . . .	96
<b>5. Two Hands and Beyond</b>	<b>99</b>
5.1. Future Ideas . . . . .	99
5.1.1. Remote View . . . . .	99
5.1.2. Non-Display Objects . . . . .	99
5.1.3. Mobile Display Integration . . . . .	100
5.2. Conclusion . . . . .	101
<b>Bibliography</b>	<b>105</b>
<b>A. Appendix</b>	<b>i</b>
A.1. Experiment: Welcome Text . . . . .	i
A.2. Experiment: Agreement . . . . .	iv
A.3. Experiment: Pre-Test Questionnaire . . . . .	v

# BIMANUAL POINTING TECHNIQUES FOR CROSS-DISPLAY INTERACTION

## *Contents*

---

A.4. Experiment: In-Test Questionnaire for Drag and Drop . . . . .	vi
A.5. Experiment: In-Test Questionnaire for Manipulation . . . . .	vii
A.6. Experiment: Post-Test Questionnaire . . . . .	viii



## List of Abbreviations

API	:	Application programmers interface
CDI	:	Cross-Display Interaction
CDOM	:	Cross-Display Object Movement
DB4O	:	Database for objects
DOF	:	Degree of freedom
D6D	:	DataPosition6D (Squidy Datatype)
D3D	:	DataPosition3D (Squidy Datatype)
D2D	:	DataPosition2D (Squidy Datatype)
EBS	:	Electronic Brainstorming System
EHCI	:	Enhanced Host Controller Interface
FOV	:	Field of View
GUI	:	Graphical User-Interface
HCI	:	Human-Computer Interaction
JNI	:	Java Native Interface
LHRD	:	Large High-Resolution Display
LP	:	Laser Pointer
MDE	:	Multi-Display Environment
OT	:	Optitrack
ZOIL	:	Zoomable Object-Oriented Information Landscape





## List of Figures

2.1. Creative collaborative work: (a) A common analog setting for designer to generate new ideas. (b) A digital environment to support designers including the use of traditional media. [Geyer et al., 2011]	6
2.2. Process illustration of (a) analog and (b) digitally enhanced creative collaborative work. . . . .	7
2.3. Cross-display Interaction: (a) Slingshot: The hand moves (red) in opposite direction aiming at the desired position [Bader et al., 2010]. (b) Lift-and-drop provides continuous feedback during the interaction [Bader et al., 2010]. (c) Touch Projector allows user to select and manipulate content on a remote screen by using a mobile-phone [Boring et al., 2010] (d) WeSpace allows users to share content from private notebooks on a large display [Wigdor et al., 2009]. . . . .	11
2.4. A selection of gestures provided for G-Stalt [Zigelbaum et al., 2010]: While (a) and (b) provide in intuitive link to the performed task, (c) and (d) can hardly be related to the corresponding task. In addition, disambiguation of gesture (c) and (d) is difficult and is likely to impose a higher cognitive load. . . . .	12
2.5. LightSpace cross-dispaly object movement [Wilson and Benko, 2010]: (a) Through-Body Transition: Simultaneously touching two surface transfers on touched object to an empty surface. (b) & (c) An object is swiped from a normal office table onto the users hand (d) The red spot indicates the object in the users hand. . . . .	14
2.6. Bimanual Interaction [Ulinski et al., 2009]: (a) The user triggers action with two equal devices each equipped with three buttons. (b)The symmetric-synchronous task performs best. . . . .	20
2.7. Distant Pointing interaction: (a) & (b) The user moves the pointer through hand movements in 3D [Vogel and Balakrishnan, 2005]. (c) The Pointing-Wand supports several pointing techniques, but triggering options can be applied [Jota et al., 2010]. (d) A laserpointer supports intuitive pointing on large displays [König et al., 2008]. . .	26

---

2.8. Hybrid RayToRelative Pointing [Vogel and Balakrishnan, 2005]: (a) In the absolute mode, the cursor transforms to a circle indicating the area of the cursor. This mode can be used to set the initial point for the relative mode. (b) From the initial point set in (a), the cursor is moved in relative mode. . . . .	30
2.9. Adaptive pointing: Adaptive pointing smooths the transition between relative and absolute CD-gain [König, 2010]. . . . .	31
2.10. Velocity Oriented approaches: (a) Adaptive Pointing [König et al., 2009] (b) PRISM [Frees et al., 2007] . . . . .	32
2.11. Comparison of distant pointing techniques: (a) Raycast Pointing (b) Arrow-Pointing (c) Image-Plane pointing (d) Fixed-Origin Pointing (e) Virtual Plane Pointing (f) Relative Pointing . . . . .	34
3.1. Project Outline . . . . .	36
3.2. Concept of a new <i>Whitey</i> : (a) Specialty formed clamps containing spherical markers are positioned on the ideal position of each finger. (b) Different perspectives of the semi-elastic marker-clamps. (c) Side-View of the base-target which remains still when the wrist is bending.	39
3.3. Filter-issue in TrackingTools. (a) The relation between the blue hand and the white finger is correct for unfiltered trackables. (b) The lag of the filtered blue hand does affect the relation between the hand and the white fingers. . . . .	40
3.4. Camera setup in the media room . . . . .	41
3.5. Filter-issue in TrackingTools. (a) The relation between the blue hand and the white finger is correct for unfiltered trackables. (b) The lag of the filtered blue hand does affect the relation between the hand and the white fingers. . . . .	45
3.6. Streaming Pipeline with sub-pipelines. The sub-pipelines are illustrated in figure 3.7. . . . .	46
3.7. Subpipelines for gesture recognition and object definition. (a) Defines 2 fixed room objects and one mobile. (b) A <i>Static-Gesture Recognizer</i> node which is attached to a <i>Rigid-Body</i> node. (c) One <i>Rigid-Body</i> node is used for the unfiltered position of the fingers and the other is used for pointing. (d) Two <i>Rigid-Body</i> nodes defining two pre-defined gestures are attached to the <i>Rigid-Body</i> node which is used for the pointing. . . . .	50
3.8. (a) A halo indicates the direction of the users off-screen pointing. (b) On a single ZOIL landscape multiple displays containing several objects can be arranged. . . . .	57

# BIMANUAL POINTING TECHNIQUES FOR CROSS-DISPLAY INTERACTION

## List of Figures

---

3.9. (a) Cursor-crossing effect: The virtual representations of the cursor for absolute pointing do not agree with the users intention. (b) Definition of an intersection point on a plane. . . . .	61
3.10. Pointing Modes: (a) Absolute pointing (b) Relative pointing (c) <i>HyPob</i> apointing (d) Acceleration problem with <i>HyPob</i> apointing. . . . .	65
4.1. Evaluation Setting . . . . .	74
4.2. Drag & Drop: (a) The user initializes the task by clicking on the homing position (green button). (b) The objects appear on both displays. (c) The orange icon in the dragged object indicates the accuracy is almost good enough. (d) The green icon indicates a good accuracy and the user can release the object. . . . .	77
4.3. Manipulation: (a) Like with the drag & drop the participant starts from a central position of the display. (b) The participant needs to adjust the colored object to the gray scale object. (c) The participant needs both hands to scale and rotate the object. (d) Finally the objects are aligned accurate enough. . . . .	78
4.4. Preferences of the pointing techniques based on the post-test questionnaire. . . . .	82
4.5. Results of the in-test questionnaire . . . . .	84
4.6. Overall Results: (a) The success rates for each task is very similar for both techniques. (b) The analysis of the task completion time shows a significant difference between the two task and minor differences between the two techniques. (c) The transition accuracy is almost equal for all tasks. (d) The number of level changes shows significant better results for <i>HyPoba Pointing</i> in both tasks. . . . .	85
4.7. Mean task completion for each user including the standard deviation. . . . .	87
4.8. Boxplot diagrams comparing a selection of 8 users and the values of all users. The percentage number at the bottom line indicates the success rate of each task. . . . .	88
4.9. Mean transition accuracy of each user for the drag & drop task . . . . .	89
4.10. Mean task completion time of each individual user for the manipulation task. . . . .	89
4.11. Boxplot diagrams comparing a selection of 8 users performing the manipulation task. The values of all users are displayed in the center. The percentage number at the bottom line indicates the success rate of each task. . . . .	90
4.12. Mean transition accuracy values for each user showing varying results. . . . .	91

4.13. Analysis of target size for drag & drop: (a) The success rate for drag & drop indicates a lower success rate for small objects using <i>Absolute Pointing</i> . (b) The difference of the task completion time is getting smaller for larger objects. (c) The transition accuracy does not indicate significant results. (d) <i>HyPoba Pointing</i> performs better according to the number of level changes for all target sizes. . . . .	92
4.14. Special boxplot visualization for time shift results. The left side of the box indicates the first half of the samples and the right side of the box the last half of samples. This allows the visualization of the shift of mean values and deviation for samples which have been performed at the beginning of the task and from those at the end of the task . . . . .	93
4.15. Analysis of the object size for the manipulation task. . . . .	94
4.16. Successrates for comparison between male and female users . . . . .	95
4.17. Mean task completion time for male and female users indicating faster interaction for the male users. . . . .	96
4.18. Mean transition accuracy for male and female users show more accurate interaction for male users. . . . .	96
5.1. Remote View: (a) The content of display 1 can be dragged to display 2. (b) The content of display 1 inside the Remote View . . . . .	100

## List of Tables

2.1. A summary of MDE requirements regarding several interaction techniques [Fäh, 2010]. . . . .	15
2.2. Average deviation caused by natural hand tremor [Myers et al., 2002].	17
3.1. Optitrack compared with Vicon and AR-Tracking, the most common tracking system used for motion capturing [Fäh, 2011]. . . . .	42
3.2. Important values for 3-Marker-Calibration in TrackingTools [Fäh, 2011]	43
3.3. Important values for camera settings [Fäh, 2011] . . . . .	43
3.4. Important values for trackable settings [Fäh, 2011] . . . . .	44
3.5. Important properties for the <i>Rigid-Body</i> node . . . . .	47
3.6. Important properties for the <i>Room-Object</i> node . . . . .	49
3.7. Minimal set of semaphoric gestures defined for the use in <i>MultiDragger</i>	52
3.8. Important properties for the <i>Static-Gesture Recognizer</i> node . .	53
3.9. Important properties for the <i>Multi-Intersection</i> node . . . . .	54
3.10. Data stored for each hand-controlled cursor . . . . .	59
3.11. Data stored for each object on the screen . . . . .	60
4.1. Computer background of all participants. . . . .	75
4.2. Different Datasets for the analysis of the experiment . . . . .	76
4.3. Accuracy and time parameters retrieved from the pre-test. . . . .	79
4.4. Evaluation schedule for each participant . . . . .	80
4.5. 846 Valid samples of all participants during the experiment. . . . .	83
4.6. T-statistic values showing the significance between the two techniques for both tasks. . . . .	86
4.7. T-statistic for a selection of users performing the drag & drop task. .	87
4.8. T-statistic for a selection of users performing the manipulation task.	91
4.9. T-statistic of the drag & drop task comparing the task completion time of the first half and the last half of samples for each user. . . .	93
4.10. T-statistic comparing the results of the drag & drop task for all male and all female users . . . . .	95
4.11. Comparison of the mean task completion times of an in official experienced user. . . . .	98

## List of Listings

3.1. Gesture-Definition for a single gesture input . . . . .	51
3.2. Bridge-Plugin for gesture input . . . . .	56
3.3. Mean direction history . . . . .	70
3.4. Crossover prevention . . . . .	71

## 1. Introduction

Collaborative multi-display environments can nowadays support users in many different situations. In a large control-room, for example in a traffic control center, several operators work in an environments with multiple displays. Often large screens provide an overview, while several small screens show details like images of traffic cameras. On the operators' desk additional information is displayed. Several operators work together to keep the traffic as fluent as possible. Therefore information needs to be shared among the operators.

Another typical place where multiple displays are used is a modern meeting room. Such meeting rooms are nowadays equipped with several displays of different size and functionality. Personal devices are often brought to meetings for personal use as well as for presentations. While some persons prefer taking notes using pen and paper, others use digital devices.

As soon as it comes to creative activities like brainstorming, affinity diagramming, strategic planning or decision making, traditional media are often preferred. Such collaborative tasks, especially in the context of industrial design heavily rely on the use of media like white-boards, post-it notes or similar media [Prante et al., 2004]. Various factors can affect the outcome of a designing process. For example, if a user stands behind a group of other collaborators, his participation is limited. The user may not see what other collaborators are doing, nor can he participate on an equal level. Common digital tools hardly integrate these factors and therefore provide only limited benefits for designers. More recent digital tools make use of multiple displays in order to re-build the common work-flow of designers. They also include digital paper which allows to synchronize the notes on paper with the digital system [Geyer et al., 2011].

Whether for meeting rooms or in creative studios, the combination of multiple displays with non-display devices or digital paper becomes more common since it allows to imitate the preferred working-method. This combination requires new tools and new interaction techniques.

## 1.1. What is this thesis about?

In this thesis we propose bimanual gesture interaction for the use in environments as described above. This kind of interaction allows all users to interact equally from any location in the environment with any device in the environment. We see a high potential in the use of such pointing gestures as they support the interaction of non-display devices and non-digital devices as well. In such an environment a user can grab a digital document from the display print it, by simply pointing at the printer or delete it, by throwing it into the real, physical garbage bin. All without any device except his bare hands.

With this idea in mind, we analyze three research fields of the HCI which are all required for this kind of interaction. Hand gesture interaction is required for the interaction with no other device than the human hand. Pointing interaction is required for distant access of any device in the environment. And finally, cross-display interaction for multi-display environments is necessary to define proper requirements for the entire setting.

## 1.2. Outline

This thesis contains four main parts. In chapter 2 we discuss three individual topics which are required to complete our goal. Collaborative Multi-Display Environments (MDEs) (section 2.1) build the basic hardware for our scenario. Existing MDEs are analyzed based on a task-related set of requirements. In order to provide natural interaction hand gesture interaction is introduced in section 2.2. The special focus is set as bimanual gestures required for distant pointing interaction. The third topic of chapter 2 focuses on pointing interaction, the connection between the MDE and bimanual gestures. Several pointing devices and techniques are discussed and analyzed. While chapter 2 primarily relies on the analysis of related work the third chapter is based on the master project.

The project <sup>2</sup>*Hands<sub>4</sub>Displays* also consists of the three basic topics mentioned previously. Gesture Tracking (section 3.2) and Optitrack (section 3.1) introduce the required hard and software specifications for the bimanual gesture recognition. Section 3.4 introduces the handling of the tracking data and defines algorithms for pointing interaction. The required functions for the MDE are provided by MultiDragger (section 3.5), a software tool which supports bimanual cross-display object movement. The final section of chapter 3 introduces *HyPoba Pointing*, a novel pointing technique for distant pointing interaction. This hybrid approach makes use of a weighted combination of absolute and relative pointing.



The experiment in chapter 4 tested *MuldiDragger* and *HyPoba Pointing* on fourteen unexperienced users. Two different tasks, one for cross-display interaction and one for bimanual gesture interaction were tested. The experiment focused on efficiency and accuracy of *HyPoba Pointing* and on the usability of the two task.

Chapter 5 describes dropped ideas, emerging problems and lessons learned throughout the development of  ${}^2\text{Hands}_4\text{Displays}$ .



## 2. Requirement Analysis

Our approach on Cross-Display interaction (CDI) includes three different aspects which are discussed in this chapter. Section 2.1 discusses different settings and requirements for Multi-Display Environments with a special regard to collaborative work. Therefore, we define a scenario with typical tasks for collaborative work. In order to complete those tasks, we propose a set of requirements with a special focus on MDEs. The second important factor for our approach on CDI is hand gesture interaction (section 2.2). Based on our research we found bimanual gesture interaction can increase the naturalness and effectiveness of interacting in MDEs. Distant gesture interaction requires an analysis on pointing techniques, the third aspect for CDI. In section 2.3 we therefore analyze several pointing techniques which can be used for CDI.

### 2.1. Collaborative Multi-Display Environment

Previously we introduced several examples where MDEs can nowadays be found. In this section we focus on creative collaborative work. Typical tasks which can also be used for other scenarios can be found there. However, the context of creative collaborative work generally refers to the combination of digital and non-digital devices.

Especially in the creative context, designers like studios with a high material character [Vyas et al., 2009]. The distribution of sketches, notes, magazine clips, physical models or prototypes in a physical environment are important to the design activity and serve as organizational memory. Externalization, the use of physical space and the use of body have been identified as major themes of collaborative practices [Vyas et al., 2009]. Physically moving notes from a private desktop to a design studio or to a collaborative meeting can destroy geographical relations and the corresponding personal associations. A digital environment can help to increase the reuse of created physical relations. To digitally rebuild a typical setting of a designer studio or a collaborative brainstorming session, multiple displays are required. Figure 2.1 compares a traditional (a) and digital (b) creative session.

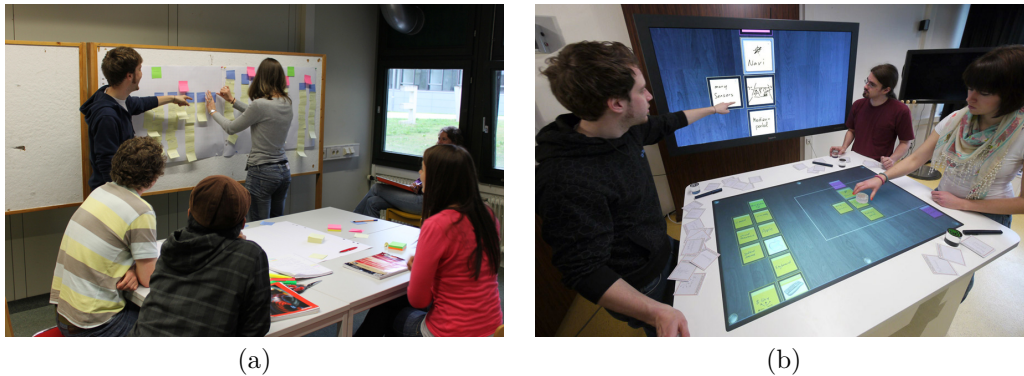


Figure 2.1.: Creative collaborative work: (a) A common analog setting for designer to generate new ideas. (b) A digital environment to support designers including the use of traditional media. [Geyer et al., 2011]

### 2.1.1. Tasks for Collaborative Work

In order to specify requirements for digital tools, supporting creative work in a collaborative way, we define a set of tasks which are typical for generating new ideas. Therefore we define three different phases which describe a typical workflow of idea and content generation. Each phase requires different in and output capabilities.

**Phase 1: Generation of new ideas and content for later discussion.** Each designer works individually (possibly at home) to generate a first private set of ideas. The goal for each designer is to create as much ideas on a specified topic as possible. Text notes, bullet lists, sketches or diagrams on different media are the outcome of this first phase. Each designer utilizes his favorite tools and media for this phase of work.

**Phase 2: Presentation of ideas and content.** In the second phase, each designer presents his ideas to the other team-members. To avoid forgetting presented ideas, each one requires a visual representation. Usually the prepared notes and sketches are pinned onto a white-board visible to everyone.

**Phase 3: Generation of new ideas.** The group discusses each individual idea. During this process, new ideas can be created, existing ideas can be changed or removed.

Especially during phases 2 and 3, cooperation can lead to more creative ideas and better solutions [Warr and O'Neill, 2005].

One of the big disadvantages of the common use of traditional media is the re-use of the outcome. A typical design process includes multiple cycles. Using digital tools the process can benefit from advantages like remote operability, storing data, replay functions or similar.

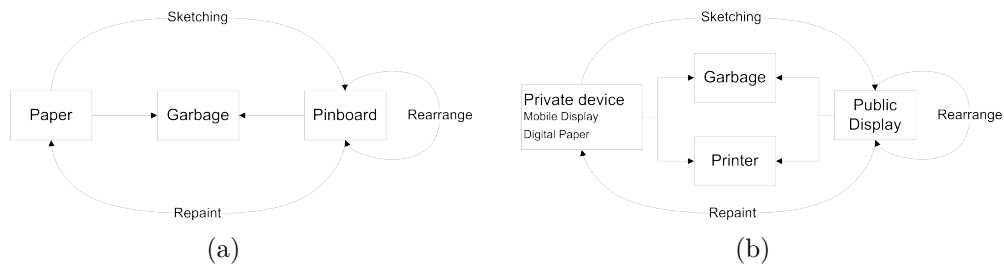


Figure 2.2.: Process illustration of (a) analog and (b) digitally enhanced creative collaborative work.

In *Phase 1* a designer traditionally uses paper-based media. Using digital tools, additional media like notebooks or tablet PCs have to be supported. The question whether designers would like to use such devices is not in the focus of this work, but with the advancing designing process more detailed sketches or models can be part of the discussion. For example, a designer has his preferred tool to generate a 3D mock-up. Most likely this tool is running on this personal device. Other designers prefer the use of pen and paper. Digital Paper<sup>1</sup> allows the combination of traditional media and digital environments.

In phase 2, each collaborator presents his ideas, sketches or even a 3D model to other users. The content generated on the private device needs to be transferred to the MDE. Therefore, the MDE needs to be able to get data from personal devices with respect to the privacy of any data on the device. We define a first task for collaborative work in a MDE as the following: (T1) *Move content from any private device to a public display*. Since the number of pixels is limited on displays, the user can rearrange and resize illustrations or sketches during the presentation. The second MDE-Task (T2) is the *manipulation of any content on any display*. The designer who created a 3D mock-up probably needs a special viewer which is installed on his personal device. This requires the definition of the third task (T3) *Access data and software on a private device*.

In the third phase, user rearrange, add or remove ideas and content. In a traditional environment, dropped ideas can be removed by unpinning the post-it from the white-board. The fourth task (T4) is *deleting objects from any display*. Designers with great sketching skills possibly prefer to use a pen to enhance existing sketches. Traditionally the designer unpins a sketch, adds the desired content and pins it back to the wall. Therefore the last task (T5) is *printing any object from any display*.

The simplified illustration in figure 2.2 compares (a) the traditional analog and (b) the digital circle of iterating through the defined tasks. Those tasks are typical in the creative context, however also in meetings and other collaborative environments

<sup>1</sup>Anoto-Technology allows users to sketch with normal paper and pen, but still it can be digitalized (<http://www.anoto.com>)

such or similar task can be found. The digital environment required for the tasks requires several displays and devices. In the following we focus on the requirements which are required to perform such tasks in a digital MDE.

### 2.1.2. Requirements for interacting with collaborative MDEs

For the tasks defined previously we define the collaborative multi-display environment as a heterogeneous configuration of any kind and shape of display and devices. In order to facilitate the terminology we do not differ whether a device has its on screen (mobile-phone or notebook) or the display is connected to another computer or network. We therefore define a display according to [Jota et al. \[2010\]](#) as an “array of light elements [...] showing dynamic output from a computer”. Mobile devices and similar devices are also defined as displays. To support the collaborative tasks defined above, different kind of devices are required.

**Stationary Displays** can basically be divided into vertical and horizontal displays.

*Horizontal displays*, commonly known as *table-tops* are mostly equipped with multi-touch technology. Those displays are great to support awareness of the group since all users can interact equally and the result of interaction can be recognized by any user of the group [[Jenabi and Reiterer, 2010](#)]. *Vertical Displays* range from common desktop displays (17” to 30”) up to wall-sized displays (30” to 250”). While desktop displays are better for individual work, large displays are more suitable for presenting content or to provide overview (e.g. phase 2 and 3). For more detailed information about the relation between task and display size please refer to [Terrenghi et al. \[2009\]](#).

**Mobile Displays** are getting smaller and more powerful every year. The smallest members of this group are *mobile-phones* equipped with gigahertz processors and up to 4” multi-touch screen. Recently, *Tablet PCs* or so called *Pads* became very popular. Providing screen-sizes up to 14” Tablet PCs are categorized somewhere between mobile-phones and notebooks. *Notebooks* are high-performance mobile personal computers with display sizes between 9” and 17”. Despite of the computational power, also graphical power is good enough to provide digital output for several external displays or rendering 3D mock-ups as by the scenario above. The main affordance of a mobile display is to keep the user mobile. It can be categorized as a private device, which is mostly accessed by its owner.

In order to provide good support for the collaborative tasks the following requirements (presented in our seminar report [[Fäh, 2010](#)]) have to be taken into account. They do not only focus on interacting in MDEs but also also for distant interaction

in general. Distant interaction is important in MDEs since displays tend to be more than an arm length away from a user. The following set of requirements mainly focuses on the collaborative tasks specified above.

- *Privacy*: During phase 2 each designer can decide whether he keeps a note private or presents it to the group. Privacy is difficult to be granted using large displays. During collaborative work users should be able to access private data and information. As long as data is not explicitly shared by the owner, only the owner should be granted access to such data [Wigdor et al., 2009].
- *Awareness*: For each user it is important to be aware of other users actions. This helps to prevent multiple users of doing the same work twice.
- *Shareability*: Designers need to share ideas in order to create new ones. The lack of a shared high-resolution display limits the sharable data and therefore imposes an overhead in data sharing. According to Wigdor et al. [2009] large high-resolution displays support collaboration since data from several collaborating users can be visualized and shared simultaneously.
- *Mobility*: In collaborative environments people should be free to change their position or perspective while performing joint tasks [Jenabi and Reiterer, 2010]. In group scenarios participants may need to change group or swap positions in the group.
- *Reachability*: If the user is mobile, it is also important to interact from any position in the environment [Jenabi and Reiterer, 2010]. Especially if displays are more than an arm length away, users should still be able to interact with the display.
- *Accuracy*: Regardless to the distance between user and display the user needs to be able to select small objects efficiently [Vogel and Balakrishnan, 2005].
- *Pointing and Selection Speed*: The selection of any object displayed should be possible without much effort. In complex MDEs objects can be located on different displays, maybe several meters away from each other. The selection of any object on any display needs to be performed quickly.
- *Comfortable Use*: Interacting with the system should be comfortable, simple and not physically demanding [Vogel and Balakrishnan, 2005].

This selection of requirements is focused on the previously defined tasks and does not claim to be complete. In addition common design principles such as effectiveness, efficiency, user satisfaction or ease of use can be seen as standard requirements in the field of HCI and are therefore not mentioned [ISO-9241-16:1999, 1999].

### 2.1.3. Related Work in Cross-Display Interaction

Many computer systems nowadays support multiple displays. Even for short presentations teachers and students tend to use a projector connected to a notebook. While the notebook provides a private space for notes, the projector is used to present content to the audience. But even in such simple settings moving objects from the notebook to the projector can cause problems. Not seeing the mouse cursor or not knowing what is currently displayed can be embarrassing and interrupt the flow of a presentation.

The following selection of tools and techniques support CDI based on varying approaches. With respect to the five tasks defined previously a strong focus is on cross-display object movement (CDOM).

**Slingshot** is an interaction technique based on the metaphor of a physical slingshot [Hascoet, 2003]. The digital *Slingshot* uses this metaphor to throw an object from a tablet PC onto a table. The user can select an object using a pen and drag it in the opposite of the desired throw direction. Releasing the object throws the object. The strength of the throw is set by the vector defined by the hand movement (red arrow in figure 2.3a).

**Lift-and-Drop** is an interaction technique proposed by Bader et al. [2010]. Lift-and-Drop is based on Airlift, a video-based input device which captures hands and fingertips independent from any display. It is a direct technique where object positions are calculated as “the orthogonal projection of the 3D position of the finger onto the respective display surface” [Bader et al., 2010]. The user simply lifts up an object from a tablet PC and drops it onto a table-top (see figure 2.3b).

**Touch Projector** uses an iPhone to select and manipulate objects on a remote screen [Boring et al., 2010]. A live video on the mobile device shows a visual representation of the objects the user is pointing at. Through touch input the user can select and also manipulate the visual representation on the mobile device. This representation is directly connected to the original on the large screen. Figure 2.3c shows how a user can select an item from a remote screen. The user could even access objects which are hidden behind another person. PrIME [Jenabi and Reiterer, 2010], a similar approach uses different visual representations of the content of the large display on the mobile display.

**WeSpace** is a collaborative workspace which allows users to bring along their personal devices and share their data on a large public display [Wigdor et al., 2009]. A multi-touch table is used as a multi-user control-center. Through the touch-table each user can directly access his personal device (Figure 2.3d).



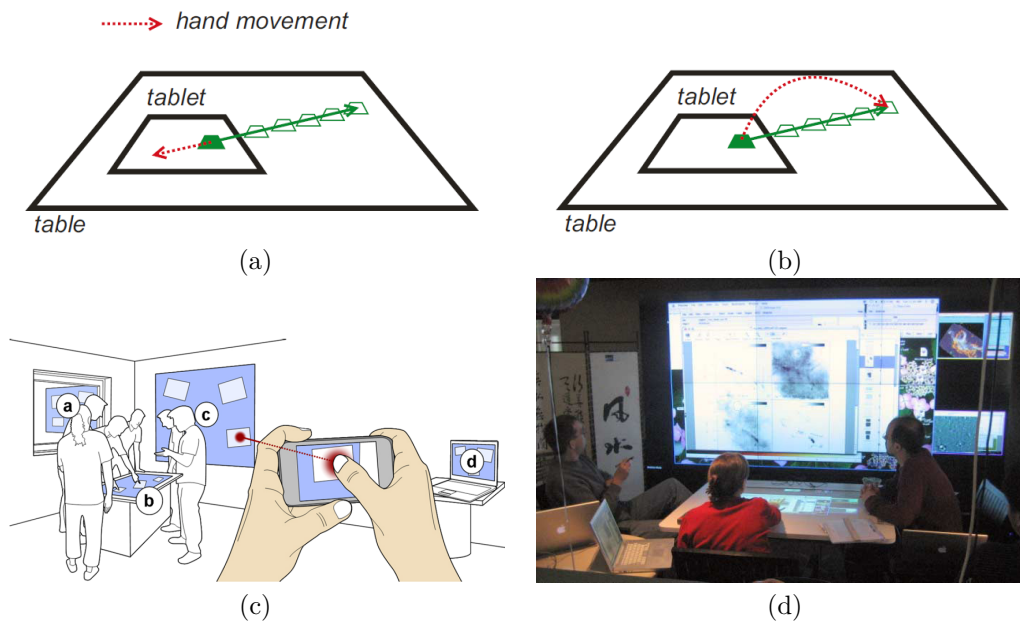


Figure 2.3.: Cross-display Interaction: (a) Slingshot: The hand moves (red) in opposite direction aiming at the desired position [Bader et al., 2010]. (b) Lift-and-drop provides continuous feedback during the interaction [Bader et al., 2010]. (c) Touch Projector allows user to select and manipulate content on a remote screen by using a mobile-phone [Boring et al., 2010] (d) WeSpace allows users to share content from private notebooks on a large display [Wigdor et al., 2009].

Using the table-top each virtual representation can be arranged and manipulated. The mobile devices can be controlled by double-tapping on its virtual representation on the table-top. This enables the user to control the devices mouse cursor on the table-top.

**G-stalt** tries to make use of the expressive power of the human body [Zigelbaum et al., 2010]. In G-stalt users can navigate through a 3D-Environment using only their hands. For this bimanual approach Zigelbaum et al. [2010] introduced the term *chriocentric* which means that the gestural interaction is based on both entire hands, not only on fingertips. This chirocentric approach is used to navigate through a collection of videos, seeking, playing and reordering videos. In order to support all required functionalities a set of 20 gestures is used. During a public demonstration at MIT<sup>2</sup> it was found the mentioned gesture set could be too complicated. The disambiguation of pinching gestures seems to be intuitive, since one hands performs a translation task and two hand perform translation and rotation (Figure 2.4 a and b). The gestures to arrange movies on a virtual (Telekinetic) line or plane seem to be hard to

<sup>2</sup>During MIT Media Lab's open hose events more than 250 visitors were able to tryout G-Stalt

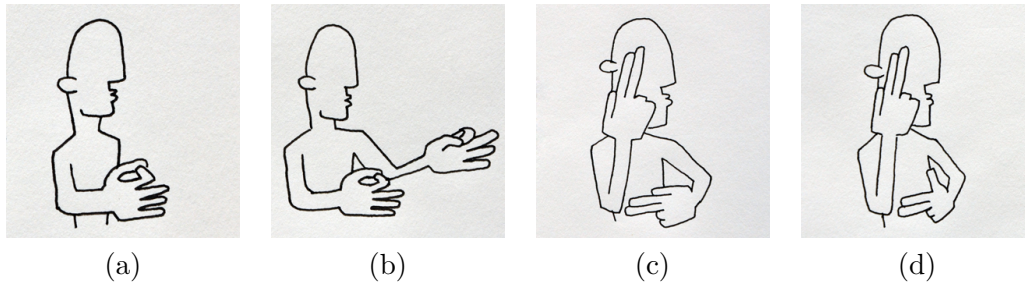


Figure 2.4.: A selection of gestures provided for G-Stalt [Zigelbaum et al., 2010]: While (a) and (b) provide an intuitive link to the performed task, (c) and (d) can hardly be related to the corresponding task. In addition, disambiguation of gesture (c) and (d) is difficult and is likely to impose a higher cognitive load.

memorize. Those two gestures (Figure 2.4 c and d) are very similar and hardly provide a natural relation to the executed task.

**LightSpace** uses multiple depth cameras and projectors to support un-instrumented cross-display object movement. No interaction devices, neither touch-sensitive displays are required [Wilson and Benko, 2010]. The user interacts with standard office tables and his bare hands. In LightSpace novel interaction techniques are presented such as mid-air interaction or Through-Body Transition. Figure 2.5a shows the simplicity of the *Through-Body Transition*. The user simply selects an object by touching and specifies the new location using the other hand. The interacting user is highlighted by a red projection from a roof-mounted projector. The shape of the user is detected by cameras and transferred to the projector.

Picking up objects is an intuitive interaction concept where objects can be moved across displays or passed to other users. By simply swiping a virtual object from the table into one's hand, the user can move the object to another user's hand or another interactive surface.

LightSpace is a very young project and has not yet been analyzed according to the requirements we defined previously. Therefore we provide a more detailed analysis for this project like it can be found in our seminar report [Fäh, 2010] for the other projects mentioned.

- *Privacy* is not considered in the current project. The techniques provided in LightSpace could easily be extended to be combined using mobile phones or even small private projections on the user's hand using the mid-air interaction technique.

- *Awareness* is granted by highlighting the entire person which is currently interacting. Through the red spot marking objects being carried by any user, other users are aware of any action.
- *Shareability* is very high since each user can individually access each item on any interactive surface.
- *Mobility* is well considered because of the possibility to carry any object personally to any place. Users can walk through the entire system and still interact in LightSpace.
- *Reachability* is not very high, since the user has to be in arm-length distance to the interactive surfaces. Interacting from a longer distance is not supported.
- *Flexibility* is limited due to the lack of input capabilities. Similar to the privacy aspect mentioned previously, additional input capabilities could be added to LightSpace. For example; virtual keyboard projections, similar to the one used for *Omnitouch* [Harrison et al., 2011] could provide possibilities for text input on any surface. Since the user is interacting only with his hands, additional input devices could simply be added, if required.
- *Accuracy* is likely to be high since only direct touch-interaction is applied. No statements about the accuracy of LightSpace were made by Wilson and Benko [2010]. Therefore this aspect is difficult to rate.
- *Pointing and Selection speed* depends on the location of the user. If the desired item is reachable the pointing and selection speed is considered high. For items further away the user first has to walk towards the interactive surface, which reduces the speed significantly.
- *Comfortable use* seems to be adequate. Holding an object in the hand looks uncomfortable due to the required hand position. All the other interaction techniques appear comfortable.

In our seminar report [Fäh, 2010] we analyzed those techniques regarding the requirements defined above<sup>3</sup>. Many projects make use of mobile devices, which generally fulfill many of those requirements. Mobility and privacy can automatically be granted, since private mobile-devices are used. On the opposite, depending on the situation, users are likely to be distracted by other things using a personal device. For example; writing an email instead of listening to other group members. In addition, not having a mobile device which is supported by the system, forgetting the

---

<sup>3</sup>For more details information about the analysis please refer to our seminar report [Fäh, 2010]

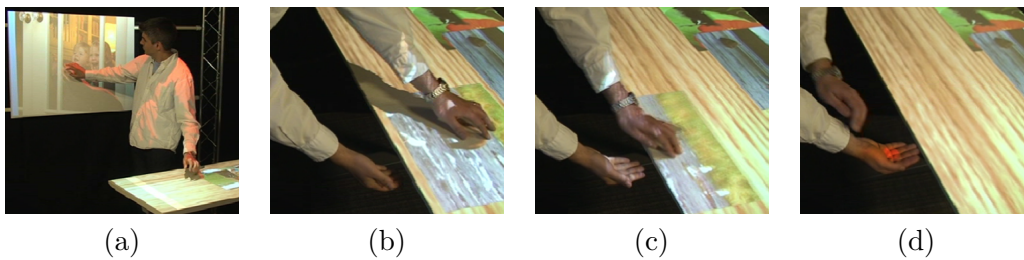


Figure 2.5.: LightSpace cross-display object movement [Wilson and Benko, 2010]:  
 (a) Through-Body Transition: Simultaneously touching two surface transfers on touched object to an empty surface. (b) & (c) An object is swiped from a normal office table onto the user's hand (d) The red spot indicates the object in the user's hand.

device at home or running out of battery are other negative factors of using mobile devices. An overview of the presented techniques above with the corresponding requirements can be found in table 2.1. The rating uses the following values:

- ++ Technique is very valuable for the according requirement
- + Technique applies in general, but not very good
- +/- Technique could not be rated
- Technique can apply under certain conditions
- Technique does not apply at all

We found gesture interaction as proposed in *G-stalt* and *LightSpace* apply best to our requirements. In *G-stalt*, *Mobility* and *Reachability* are not limited at all. Since the user can use his bare hands, (despite of the markers in *G-stalt* which most of the current system still require) the *flexibility* is very high since the user can easily use touch-surfaces, keyboards or mobile devices simultaneously. Only privacy is not considered at all using the approach from *G-stalt*.

Summarizing all the aspects, we found gesture base interaction has a high potential for the tasks defined previously. We assume that technical improvements will soon make the marked gloves from *G-stalt* needless. Even though *LightSpace* requires less accuracy regarding gesture recognition, it already provides a good example of markerless tracking. In the near future users could be able to just walk in and use the system without the need of any additional interaction device, except for two bare hands.

“The most profound technologies are those that disappear” Weiser [1999]

Using gesture based freehand pointing, the need of a physical input devices can be reduced or even be eliminated. The way gesture interaction is used in *G-stalt* requires to consider two main aspects. First, least on pointing technique has to be

Table 2.1.: A summary of MDE requirements regarding several interaction techniques [Fäh, 2010].

	Privacy	Awareness	Shareability	Mobility	Reachability	Flexibility	Accuracy	Pointing and Selection speed	Comfortable use
Slingshot	++	-	-	-	-	-	-	+	-
Lift-and-Drop	++	++	+	-	-	-	++	+/-	+
Touch Projector	++	-	-	++	++	+	++	-	+
WeSpace	+	+	++	-	+/-	+/-	++	+	+
G-Stalt	-	++	++	++	++	+/	+	+	+
LightSpace	+	++	++	+	-	++	+	+/-	+

applied in order to provide freehand pointing from a distance. And second, static gestures can be used for triggering actions.

The following section (2.2) focuses on gesture interaction especially on pointing gestures as proposed by *G-stalt*.

## 2.2. Hand Gesture Interaction

Human gesture provide an intuitive and natural way of interacting with the computer. In current research, there is a wide diversity through the field of human gestures. Even the term *gesture* itself is often used in different ways. According to Fikkert [2010] gestures can be defined as “a motion of hands, facial expressions, gaze tracking, head movements, hand postures and whole body postures”. Hand gestures will be defined analogous to the definition by Foehrenbach [2009].

“A hand gesture is a movement of the hand and fingers, performed by the user with the intention to interact with the computer.” The shape of the hand is referred to as “hand posture” or “static gesture”.

Kendon [2004] describes a variety of everyday gestures which are used in combination with speech. These kinds of gestures are interesting for human-computer interaction, as they are already known by the potential users and could therefore lead to a decreased learning effort and a better recall when used for interaction.

**Naturalness:** “The hand is uses every day for a variety of tasks, using skills which require little thought.” [Sturman, 1992]

Everyday gestures are interesting for human-computer interaction, “as they are already known by the potential users and could therefore lead to a decreased learning effort and a better recall when used for interaction” [Foehrenbach et al., 2009]. From the day when humans are born they learn the necessary skills to manipulate physical objects. With little or no cognitive load a healthy person can grab, move or turn objects. The person is task-orientated and does not have to focus on the input device.

**Expressiveness:** “The position and movements of the hand and fingers provide the potential for higher power of expression” [Baudel and Beaudouin-Lafon, 1993]. In human communication gestures do not only specify a command, but also its parameters. For example, waving a hand to someone may indicate the other person to come over. The velocity and amplitude of waving may additionally indicate whether to come quickly or not.

**Mobility:** Hands are always close to the user’s position and therefore the mobility of the hands also depends on the users position. Even if the user is sitting on a table the mobility of each hand has at least the range of an arm’s length. This aspect has to be taken into account when using hands as an input device.

### 2.2.1. The Human Hand

With all the advantages the human hand can have for human-computer interaction, it also imposes several problems. Especially for distant midair interaction the main problem is pointing accuracy. König [2010] specifies two main problems according to distant pointing interaction. *Natural Hand Tremor* which is based on muscular contractions and *Human Motor Precision* based on the hand-eye coordination.

**Natural Hand Tremor** causes serious noise using absolute pointing devices in midair.

With increasing distance between the pointing devices and the display simple tasks like pointing or selection become more difficult or even impossible. Physiological tremor, caused by involuntary muscular contraction, affects the hand when holding a constant position. Myers et al. [2002] evaluated different pointing devices according to deviation caused by hand tremor. Table 2.2 shows a selection of the results: a laserpointer held close to the body (*laser close*) and a laserpointer held at arm’s length (*laser extend*). The average deviations range from 4.31mm up to 11.68mm which cause serious inaccuracies. To overcome this problem various filter methods such as low-pass filtering, moving windows or kalman filter can be applied.

**Human Motor Precision** describes the pointing precision a human can achieve. Increasing distance between user and display reduces the pointing precision. The



Table 2.2.: Average deviation caused by natural hand tremor [Myers et al., 2002].

<i>pointing device</i> \ <i>distance</i>	5 feet (152cm)	10 feet (304cm)	15 feet (457cm)
laser close	0.17" (4.31mm)	0.29" (7.36mm)	0.36" (9.14mm)
laser extend	0.18" (4.57mm)	0.35" (8.89mm)	0.46" (11.68mm)

human muscles cannot perform movements fine enough for the selection of individual pixels in direct selection tasks. Not only the limited movement of muscles, also limited hand-eye coordination and the limited perception capability of the human eye are factors which increase the difficulty of interacting from a distance [König, 2010]. Applying transfer functions with adequate CD-Gain can help to overcome those limitations (Section 2.3.1).

Even though the control of the hand is limited, it remains the most powerful tool in our every day lives. Naturalness, expressiveness and mobility motivates further investigation in this area.

### 2.2.2. Gesture Types

The diversity throughout the research field of human gestures is wide. Karam and Schraefel [2005] discuss different aspects on gesture interaction in their taxonomy of gestures in human computer interaction. The five different gesture types evade from their taxonomy:

- *Deictic Gestures*: The first application using deictic gestures is Bolt’s “Put that there” [Bolt, 1980]. Generally, “pointing in narrative is known as a deixis” [Fikkert, 2010]. Selecting and pointing at virtual objects is the most common use of deictic gestures.
- *Manipulative Gestures*: Quek et al. [2002] defined manipulative gestures as tight mapping of the movements of the hand and arms to the movements of some virtual object in the interface. Since Karam and Schraefel did not differentiate between direct and indirect mappings of gestures, the traditional mouse interface also belongs to this category. Manipulation of a virtual object with the mouse is the indirect mapping of a manipulative gesture from the hand via mouse to the cursor. Such gestures are mapping gesturing in two degrees of freedom to two-Dimensional interactions. In Rekimoto’s Pick-and-Drop [Rekimoto, 1997] the user physically picks up an object and moves it to another place without touching the surface. According to Karam and Schraefel this kind of manipulative gesture of multiple degrees of freedom also maps to 2D interactions. Looking at G-Stalt [Zigelbaum et al., 2010] a logical extension

to this taxonomy has to be made, since G-Stalt is using multiple degrees of freedom for 3D interaction.

- *Semaphores*: Gang-signs like “hang-loose” or “east-side” are commonly seen as a well known example for semaphoric gestures. Scuba divers use a well specified set of signs to communicate with their buddies. Such gestures are not considered intuitive gestures, since such signs have to be learned. While deictic and manipulative gestures are generally made with hands and arms, semaphoric gestures can be performed with fingers, hands, head, arms, feet, body or even hand-held objects [Fikkert, 2010]. Semaphoric gestures are divided into static- and dynamic gestures. Forming the “ok” sign using thumb and forefinger is considered to be a static sign while waving a hand to say good-bye is a dynamic gesture.
- *Gesticulation*: This last category is regarded to be the one of the “most natural form of gesturing and is commonly used in combination with conversational speech” [Karam and Schraefel, 2005]. Gesticulations are idiosyncratic spontaneous movements of the hands and arms during speech [Fikkert, 2010].

Common approaches combine deictic gestures for the indication of the point of interest and semaphoric gestures to trigger actions. One aspect that makes the human hand a very powerful tool is the fact that most humans have two hands. In addition to the previously named gestures, gestures can also be performed with two hands.

### 2.2.3. Bimanual Gestures

Most tasks in our everyday life are done by two hands. Especially complex tasks require the use of both hands. Researches on bimanual interaction are concerned with how such highly skilled actions can be used to improve human-computer interaction. With the emerge of accurate multi-touch displays bimanual interaction is now widely used. Research in this field started much earlier with different devices. More than twenty years ago, Buxton and Myers found that “the success of bimanual interaction techniques relies heavily on the degrees of parallelism and symmetry between the tasks assigned to each hand” [Buxton and Myers, 1986]. For comparison Buxton and Myers tested a scrolling and selection task in a desktop application. The same tasks had to be completed either using one hand or using both hands. While the dominant hand was using the mouse for the selection task, the non-dominant hand interacted with a dedicated scrolling device. In the bimanual condition, expert users were 15% faster than in one-handed and novices were 25% faster in the bimanual condition.



More recently [Balakrishnan and Hinckley \[2000\]](#) investigated symmetric bimanual interaction and how attention, speed and visual integration of the tasks affects performance. Therefore the user was controlling two individual cursors on a graphic tablet. The task was to follow two red squares on the screen. These targets were moving on the screen with changing velocities. Alternatively the targets were moving with the same speed in the same direction. With increasing difficulty of the tasks participants generally first moved one cursor than the other. They found that “symmetric tasks where the two hands are not operating nearby in the focal visual field should be avoided” [[Balakrishnan and Hinckley, 2000](#)]. If the two cursors are widely separated, some secondary feedback can increase the performance. In a navigation task such feedback could be provided in overview visualization.

[Guiard \[1987\]](#) proposes a framework of bimanual manipulation with three different classes of bimanual actions. According to Guiard, asymmetric interaction are those in which the non-dominant hand sets the frame of reference in which the dominant hand works. Therefore the action of the non-dominant hand precedes that of the dominant hand. For most users the dominant hand can be controlled more precisely. However some people write with the left hand and play a ball with the right hand. In our every-day life, there are many examples of asymmetric two-handed activities. For example, the dominant hand is writing a text while the non-dominant hand holds and positions the paper.

[Ulinski et al. \[2009\]](#) extended Guiard’s framework by adding an additional class. In their research Ulinski et al. compares four different selection techniques (in 3D), one for each class of bimanual interaction. All selection techniques use a volumetric box for selection. All objects inside the box are selected. The way the box is placed, rotated and scaled varies according to the respective bimanual class. The system provides an equal interaction device for each hand (Figure 2.6a). The device is magnetically tracked providing 6 degrees of freedom (6DOF) to locate the 3D positions and orientations of each hand in front of the display. It has three buttons for triggering actions.

- *Symmetric-Synchronous (SS)*: Each hand is performing identical actions at the same time. For the selection, each hand is virtually attached to opposite corners. The box can be moved scaled and rotated with both hands at the same time. To scale the box, a button on the interaction device has to be pressed. Position and orientation could still be controlled while modifying the scale.
- *Symmetric-Asynchronous (SA)*: Each hand is performing identical actions at different times. In this setting, the position of the box is changed by changing the position of both hands one at a time. Pressing the scale button with the

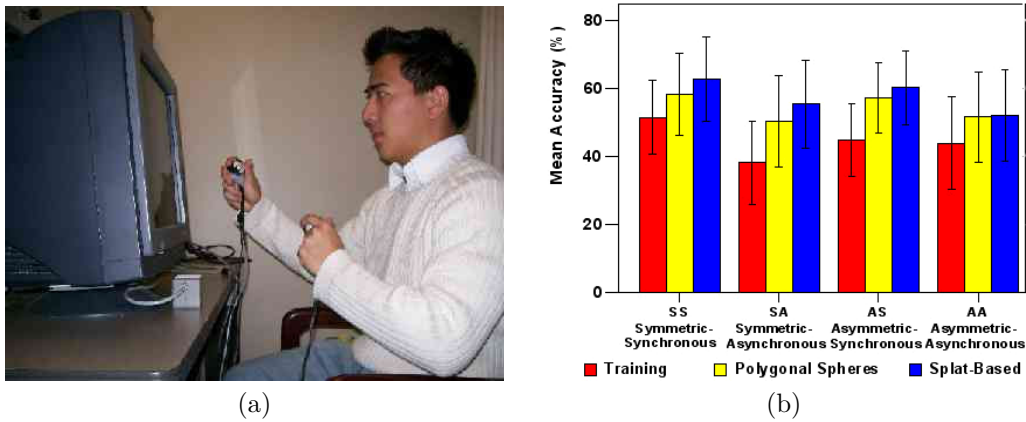


Figure 2.6.: Bimanual Interaction [Ulinski et al., 2009]: (a) The user triggers action with two equal devices each equipped with three buttons. (b) The symmetric-synchronous task performs best.

dominant hand enables the scaling mode of the box. Position and orientation can still be controlled while modifying scale.

- *Asymmetric-Synchronous (AS)*: Both hands are performing different, but coordinated actions to perform the same task synchronously. For both asymmetric techniques the movement of only one hand is permitted. Therefore an additional button on each device is used to toggle the active hand. The manipulation of the box is implemented like in the SS technique. With the difference that for each action only one hand at a time can be used. Position and orientation can still be controlled while modifying the scale.
- *Asymmetric-Asynchronous (AA)*: Both hands are performing different, but coordinated actions to perform the same task asynchronously. Again, the box is positioned by moving the non-dominant hand and rotated by rotating the non-dominant hand. Therefore position and rotation can be performed at the same time, while modifying the scale needs to be done separately by the dominant hand.

The results suggested “that symmetric and synchronous selection strategies both contribute to faster task completion” [Ulinski et al., 2009]. Regarding physical demand, none of the classes showed significantly different results. Whereas asynchronous action significantly increased the cognitive demand in asymmetric techniques and decreased the ease of use in symmetric techniques. The results in figure 2.6b show the best accuracy mean for the symmetric-synchronous task.

The use of symmetric and synchronous selection strategies in combination with static semaphoric gestures to trigger selection seems to be a combination with a low cognitive load. Those findings agree with those from G-stalt, where users were

confused by the large set of gesture commands. With respect to the limiting factors of the hand (hand tremor and limited precision) a good pointing technique is required in order to apply bimanual pointing gestures.

## 2.3. Pointing Interaction

In the previous sections we found that bimanual gesture interaction can be a adequate base for cross-display interaction in a collaborative context. The choice of a good pointing technique is the last missing piece to complete the entire interaction cycle.

Pointing interaction is commonly used in combination with large high-resolution displays (LHRD). More recently, pointing techniques from LHRDs have been adapted for the use in MDEs (Section 2.1). This section focuses on pointing techniques with a special regard to the use in MDEs. Therefore we first provide an overview of different pointing devices which have been used for pointing experiments in related work.

### 2.3.1. Pointing Devices

Pointing devices are used to control a cursor which indicates the point of interest. Mouse, trackballs or graphic tablets are well known examples of pointing devices. Recently, additional devices such as touch screens, data gloves or laserpointers have been added to the group of pointing devices. Pointing devices can be defined as a subclass of input devices in general and play an important role in today's graphical user interfaces (GUI) (Bieg [2008] according to [ISO-9241-16:1999, 1999]). One of the main GUI tasks for pointing devices is the *selection-through-pointing-task*. In such tasks the pointing device typically sets the position of the pointer, while a trigger performs the selection. The selection can be triggered for example by buttons, gestures, tapping or dwelling. Not only selection but also dragging items across the screen, tracing scrollbars or browsing on websites are tasks designed for pointing devices.

Nowadays various pointing devices are commercially available [Buxton, 2010]. Most pointing devices share the same set of properties proposed in MacKenzie [1995] and Hinckley [2008]. In the following we define the most important input-device properties and parameters. For a more detailed analysis please refer to König [2010].

### 2.3.1.1. Input-Device Properties

**Physical property sensed:** Position, motion or force are the most common physical properties being sensed. While a mouse senses the motion performed by the hand a joystick senses forces to certain directions. Despite those common physical properties also other properties such as body temperature, light intensity, sound level or electric capacity can be used for human-computer interaction [König, 2010].

**Degree of freedom:** The total number of dimensions an input device senses are commonly defined as *degree of freedom* (DOF).

1 DOF	Button: Pressed or Released
2 DOF	Mouse: Movement of X- and Y-Axis. Each Button defines an individual 1 DOF (e.g. Mouse with 3 Buttons = 5 DOF)
3 DOF	3D-Point: A single Point (e.g. a tracked finger, see section 3.4)
6 DOF	Hand: A hand in a 3D environment (see the Tracking section) has 3 DOF (X,Y,Z) for the definition of position and another 3 DOF (heading, Attitude, Bank) for the orientation.
$x$ DOF	Hand with Finger: Depending on definition, each finger has its own DOF or each gesture defined for a hand specifies another DOF.

**Indirect vs. Direct:** The difference between direct and indirect can be easily explained comparing mouse and touch-input. A mouse is generally moved on a table and controls a virtual cursor on a screen. This increases the cognitive effort [König, 2010]. The user is forced to map the forward direction of the mouse to the upward direction of the mouse cursor. Whereas touch-input provides a unified input and display surface [Hinckley, 2008]. This reduces the cognitive load for moving objects since the cursor can be directly replaced by the touch-point of the finger. On the opposite, occlusion is likely to occur.

**Absolute vs. Relative:** Touchscreens are a very common example for absolute devices. Such devices have a reference frame with an absolute origin [Bieg, 2008]. For instance, touching the center point of a touch screen directly moves the cursor on the same position. Recent improvements in technology provide touch-surface which does not need to calibrate the origin. Current mobile phones are a good example for absolute devices which can be used by any person without calibration. Typically absolute devices directly define a coordinate tuple X and Y.

The most common example for a relative device is the mouse. It does not have an absolute origin. A mouse does not sense its location on the mouse-pad but the relative movement between time  $i$  and time  $i - 1$ . Relative devices often transmit displacement or velocity information. A common effect of relative devices is called “clutching”. If the mouse moves out over the table or the mouse-pad, the user “clutches” by levering the mouse and replacing it to continue the action. Relative movements are defined by a transfer function.

### 2.3.1.2. Input-Device Parameters

In addition to the previously introduced properties, input device parameters such as *Transfer function*, *device acquisition time* and so on have a deep impact on the pointing behavior.

**Transfer function:** The transfer function describes the mapping from input to output. Position sensing devices perform a 1:1 mapping between the device movement and the movement of its virtual representation (e.g., hand position defines cursor position on a touch screen) [König, 2010]. Motion or force sensing input devices perform a relative mapping of the user's input and the virtual representation. A simple linear example for a relative mapping transfers 1 inch movement of the mouse into 10 pixel cursor movement. Non-linear mapping functions provide fast cursor movement for fast device movement combined with precise cursor movement for slow device movement. This control-display gain (CD gain) can be manipulated to enhance pointing speed and accuracy.

**Gain** is often defined as the ratio between magnitude of changes in the position of the control (e.g. mouse, joystick) and the magnitude of the pointer on the display. The *control-display gain* (CD-gain) is 1 if the “display pointer moves at exactly the same distance and speed as the control device” [Casiez et al., 2008]. Hinckley [2008] defined the CD gain as “the distance moved by an input device divided by the distance moved on the display”.

**Device acquisition time** is defined as the time a user needs to move a hand to a device. A common example is moving a hand from the keyboard to the mouse in order to position the cursor instead of writing text.

**Filtering** reduces jittery motions of indirect input devices. Inaccuracies in the sensors or natural hand tremor can be smoothed using filtering mechanisms such as Kalman filtering or similar [Oh and Stuerzlinger, 2002].

**Lag** is defined as the responding time of the system between input and response [MacKenzie, 1995]. The first source for *Lag* is the sensor itself. We define

$Lag_{Sensor}$  as the time between the users interaction and the time the sensor publishes the output.  $Lag_{Filter}$  is the time filtering algorithms require to calculate the smoothed data based on the unfiltered incoming data from the sensor.  $Lag_{Output}$  is the time the system needs to handle the action performed by the user. Lag is a serious issue regarding fluent interaction.

### 2.3.2. Distant Pointing Devices

Most pointing techniques have been introduced and evaluated for LHRDs. Distant pointing, direct pointing at the display and laserpointers are commonly used for such displays [Jota et al., 2010]. Regarding pointing techniques for MDEs no studies comparing diverse pointing techniques could be found. We assume that regarding the requirements defined in section 2.1 most pointing techniques can be applied for MDEs as well.

Before we move on with those pointing methods, we quickly introduce the pointing devices used for the analysis. Each of those devices supports at least two pointing techniques described in the following section.

**Laserpointer-Interaction** is a flexible and position-independent input technique especially for large displays. It allows direct pointing at any point on the display, like a intuitive extension of the hand [König et al., 2008]. An invisible infrared laserbeam emerges from the laserpointer. Infrared cameras detect any intersection with the display and calculate the pointer position on the displays. This technique requires rear-projection displays since the cameras need to be behind the projection-surface. This limits the portability to other displays, since LCD or plasma displays do not support laserpointer tracking. For displays supporting the laserpointer it allows interaction with almost no lag. Three buttons provide full control of a mouse cursor as with a standard mouse (see Figure 2.7d). The calibration of the device is user-independent. This allows passing the device from user to user and is just ready to use.

**Freehand-Pointing** as introduced by Bolt [1980] allows the user to specify the point of interest by pointing with a finger or a hand. Back in 1980, Bolt used magnetic coils to measure the magnetic field of a ring worn by the user. The movements of the users hand were limited to a small invisible window where the coils could determine the position of the ring. Since the position of the user was a fixed seat, the pointing direction was calculated as a projection from the fixed origin and the variable point defined by the magnetic ring. More recent,

commercially available products (e.g. iPoint-Presenter<sup>4</sup> or Microsoft Kinect<sup>5</sup>), make use of planar interaction spaces where users can point by simply moving the hand on a virtual plane. Those techniques do not support the mobility of the user and will therefore not be considered in our research.

Freehand-Pointing as used in *G-stalt* is broadly discussed in Vogel and Balakrishnan [2005] and Foehrenbach et al. [2009]. Commonly visual tracking systems such as Vicon or ART are used for similar projects. Recent improvements in technology support markerless hand tracking as presented in Lightspace [Wilson and Benko, 2010]. Even though most researchers still make use of retro-reflective markers. Generally a pre-defined target (see figure 2.7b) on the users hand wrist determines the position and orientation of the pointing hand. Both, position and orientation have 3 degrees of freedom (3DOF). The combined data with 6DOF allows pointing from any position in the environment. A common problem with freehand pointing is the lack of buttons. While earlier systems added buttons to trackable data gloves [Hinckley et al., 1994], Vogel and Balakrishnan [2005] introduced two techniques to replace buttons by gestures.

**Pointing-Wand** is introduced by [Jota et al., 2010] to compare several ray pointing methods. Figure 2.7a shows this simple interaction device which is only used for pointing. No buttons for triggering functions were added. It is equipped with a set of retro-reflective markers and is tracked by a vicon tracking system, as mentioned above.

### 2.3.3. Distant Pointing Techniques

Distant pointing techniques have a long history. More than three decades ago Bolt [1980] created *Put that there*. A system which was only controlled by freehand pointing and speech. In this section we provide an overview of common pointing techniques. We analyze those techniques with special regard to their extendibility for MDEs. In the following analysis we also differ between *rotational* and *positional* interaction. Rotation-based pointing is affected by the rotation of the hand while position-based pointing is only affected by the position of the hand.

#### 2.3.3.1. Individual Techniques

We define the term “individual techniques” as a group of techniques which simply consider one aspect of the hand movement. Whereas “hybrid techniques” refer to

---

<sup>4</sup>iPoint from the Heinrich Hertz Institute (<http://www.hhi.fraunhofer.de/>)

<sup>5</sup>Kinect from Microsoft (<http://http://www.xbox.com/kinect>)



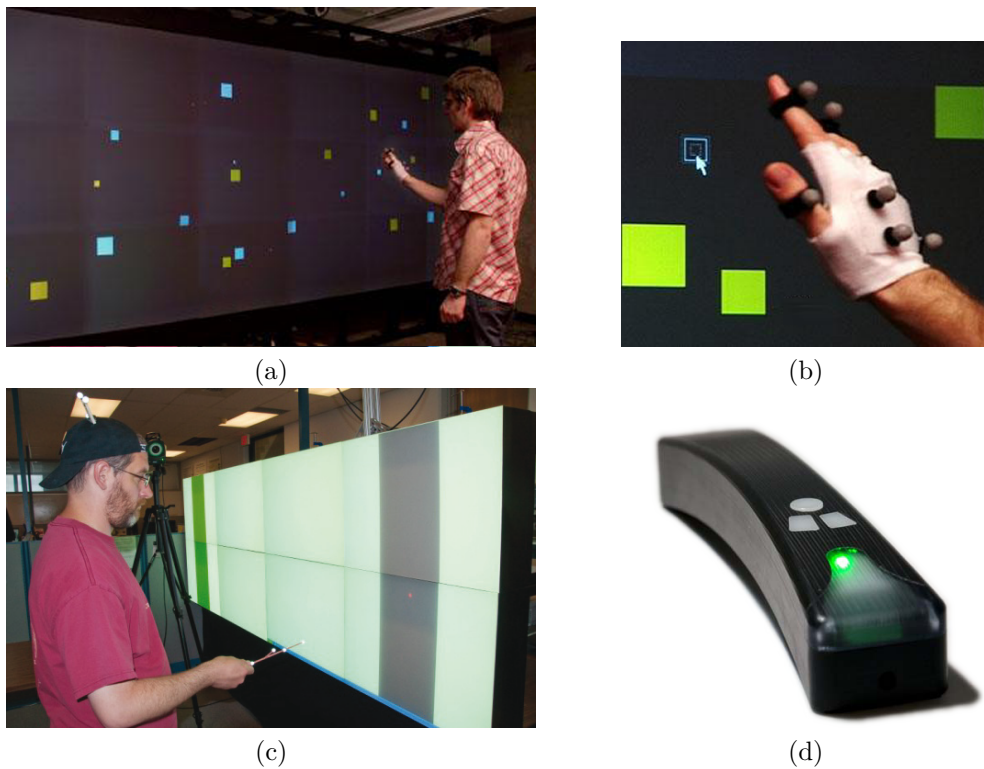


Figure 2.7.: Distant Pointing interaction: (a) & (b) The user moves the pointer through hand movements in 3D [Vogel and Balakrishnan, 2005]. (c) The Pointing-Wand supports several pointing techniques, but triggering options can be applied [Jota et al., 2010]. (d) A laserpointer supports intuitive pointing on large displays [König et al., 2008].

techniques which combine several pointing techniques or use complex acceleration functions.

**Ray Casting** is the most common variant of ray pointing. To disambiguate between the technique and device we chose the nomenclature from Vogel and Balakrishnan [2005]. As defined by Jota et al. [2010] “the ray is specified directly by the position and direction of a physical device”<sup>6</sup>. The *Absolute Pointing* for laserpointer interaction as mentioned by König et al. [2008] is also based on that definition. Vogel and Balakrishnan [2005] defined the cursor position as “a ray emanating from the index finger intersecting with the display”. Even though the interaction techniques are different, the mathematical calculations of the intersection point are identical. As shown in figure 2.11a the pointer position  $D$  is calculated as a projection from the position of the control  $C$  and its orientation  $\vec{c}$  intersecting with the display surface.

<sup>6</sup>In Jota et al. [2010] this technique is called *Laser Pointer*



2.3. Pointing Interaction

---

<i>Type of control</i>	Rotational, Absolute
<i>Display Selection</i>	Good
<i>Large Displays</i>	Good
<i>Small Displays</i>	Low accuracy

**Arrow Pointing** relies on the metaphor of using a bow and arrow. Jota et al. [2010] constraint “the use of the pointer to be somewhat aligned with the user’s eye”. This approach uses the same calculations as for *Ray Casting*, only that the user was told to use the wand “by looking down the shaft at the screen” (Figure 2.11d).

<i>Type of control</i>	Rotational, Absolute
<i>Display Selection</i>	Good
<i>Large Displays</i>	Good
<i>Small Displays</i>	Low accuracy

**Image-Plane Pointing** emerges from visual arts where painters estimate the size and position of an object by placing a thumb between one eye and a painting [Jota et al., 2010]. This technique was often found in virtual reality applications where it’s referred to as occlusion selection or crushing heads technique [Argelaguet and Andujar, 2009; Wingrave et al., 2005]. Figure 2.11b shows a ray defined by two points. The origin  $O$  is defined by the location of the users eye. The control  $C$  is defined by the users thumb. The intersection of the scaled vector  $\vec{OC} = \vec{c}$  with the display defines the intersection point. From the users point of view, the pointer is directly aligned with his thumb.

<i>Type of control</i>	Positional, Absolute
<i>Display Selection</i>	Good
<i>Large Displays</i>	Good
<i>Small Displays</i>	Low accuracy

**Fixed-Origin Pointing** is a reproduction of Shadow-reaching. *Shadow-reaching* [Shoemaker et al., 2007] allows the user to control pointers on a large display through the users shadow-cast. A light source from a point behind the person illuminates the user and produces a shadow. This idea was reproduced by Jota et al. [2010] as *fixed-origin pointing*. A fixed origin  $O$  near the bellybutton of the user simulates the light source. The vector  $\vec{OC} = \vec{c}$  in figure 2.11e represent the straight lines from the shadow-cast.

---

<i>Type of control</i>	Positional, Absolute
<i>Display Selection</i>	Good
<i>Large Displays</i>	Good
<i>Small Displays</i>	Inaccurate

**Virtual-Plane Pointing:** Another absolute pointing technique can be found in several commercial products such as Microsoft-Kinect or iPoint-Presenter. We categorize this method according to the fact that the user defines an absolute position through a virtual representation of the screen. According to [Vogel and Balakrishnan \[2005\]](#) this method performs fast and accurate, but in order to support the mobility of the user the virtual plane needs to be coordinated with the body movement.

<i>Type of control</i>	Positional, Absolute
<i>Display Selection</i>	not possible
<i>Large Displays</i>	depending on scalability
<i>Small Displays</i>	depending on scalability

**Relative-Pointing** is a technique which uses the translation of the controller's position to determine the pointer position. As shown in figure 2.11f the cursor on the display  $D$  is moved relative to the movement of the control  $C$ . This method requires an appropriate gain function for the mapping of the control movement to the display movement (CD-gain). Depending on the choice of the CD-gain a recalibration function according to Hinckley's [[Hinckley et al., 1994](#)] *ratcheting recalibration mechanism* should be provided. Such mechanisms, also known as *clutching mechanism* allow the user to move the control (e.g. Hand) without affecting the pointers position. [Vogel and Balakrishnan \[2005\]](#) proposed a clutching gesture, however a button on a laserpointer could also be applied for clutching.

<i>Type of control</i>	Positional, Relative
<i>Display Selection</i>	not possible
<i>Large Displays</i>	Depending on Gain
<i>Small Displays</i>	Depending on Gain

### 2.3.3.2. Hybrid Techniques

Different techniques for switching between absolute and relative input mode can be found in related work. [König \[2010\]](#) proposes a classification scheme which distinguishes between target-oriented, manual-switching and velocity-oriented approaches.

We define *hybrid techniques* as absolute pointing techniques which change CD-gain to adapt the pointing behavior.

### Target Oriented Techniques

Target-oriented techniques require knowledge of the environment in order to lower the CD-gain when the cursor is close to an interactive target. This is referred to as a “fisheye effect in motor space” [Baudisch et al., 2005]. This method can cause several problems. First, environments containing large numbers of objects are likely to have strange interaction behaviors since the cursor can permanently be attracted by an object. Secondly moving quickly out of one target or to drag a target can lead to unintentional effects.

### Manual Switching Techniques

Manual Switching techniques require a trigger to switch between absolute and relative. *Hybrid RayToRelative* Pointing is an approach by Vogel and Balakrishnan [2005] to eliminate the cognitive load of clutching. As the name implies it allows switching between absolute and relative pointing. Two different pointing gestures refer each to one pointing technique. For laserpointer interaction other triggers like buttons could be applied as well. According to Vogel and Balakrishnan [2005] we explain the method using the hand gestures.

A ray emanating from the users index finger (Figure 2.8a) is used for raycast pointing. In the absolute mode the cursor is replaced by a large circle. The circle suggests the selection of an approximate area. The circle can be rapidly moved over large distances with only little movements of the hand. Switching from the pointing gesture (Figure 2.8a) to the relative gesture (neutral hand posture, Figure 2.8b) the circle is replaced by the cursor at the same position. Using an adequate CD-gain for the relative mode, the cursor can be placed with very high accuracy. This hybrid approach allows users to quickly select a wider area of interest by absolute pointing and precise pointing using the relative pointing mode.

<i>Type of control</i>	Hybrid (rotational & positional)
<i>Display Selection</i>	Good
<i>Large Displays</i>	Good
<i>Small Displays</i>	Good

An experiment comparing *RayCasting*, *Relative Pointing* and *RayToRelative Pointing* shows a significant high error rate for the RayCasting method for small targets. Three different target-sizes ( $W_L = 144\text{mm}$ ,  $W_M = 48\text{mm}$ ,  $W_S = 16\text{mm}$ ) have been compared for different selection tasks [Vogel and Balakrishnan, 2005]. Figure 2.8(c)

shows the mean error rate for the sequence task<sup>7</sup>. As the error rate for raycasting on small targets, also the selection time increases. The overall result showed faster interaction using *RayCasting* especially if clutching would have been required in the relative mode. The high error rate “prevents it from being a practical technique” [Vogel and Balakrishnan, 2005]. No major differences between the *Relative Pointing* and *RayToRelative Pointing* could be found.

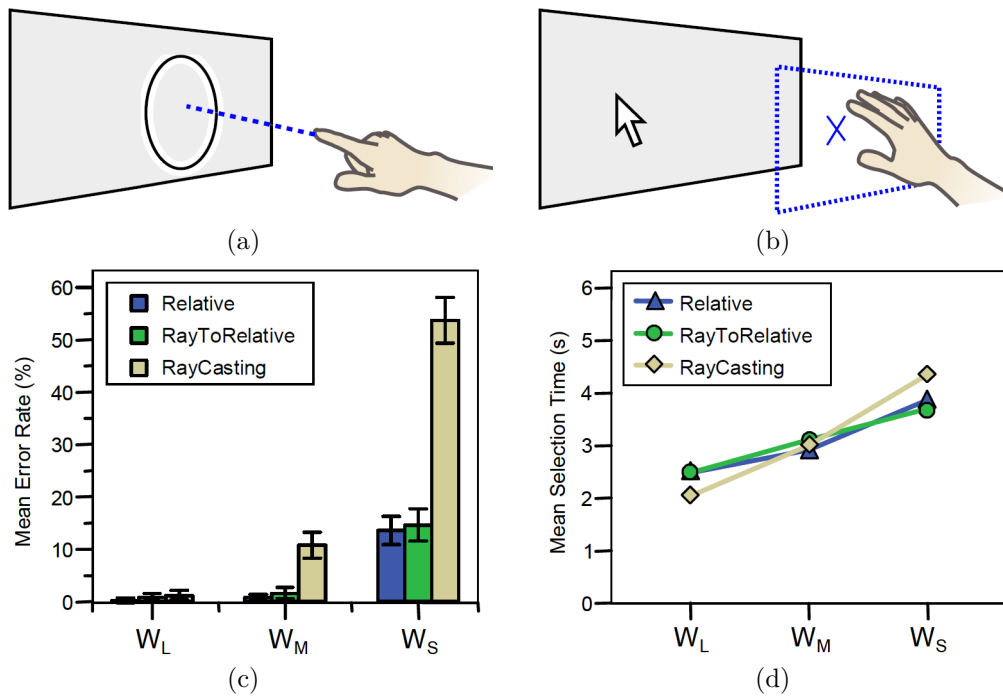


Figure 2.8.: Hybrid RayToRelative Pointing [Vogel and Balakrishnan, 2005]: (a) In the absolute mode, the cursor transforms to a circle indicating the area of the cursor. This mode can be used to set the initial point for the relative mode. (b) From the initial point set in (a), the cursor is moved in relative mode.

Forlines et al. [2006] described another approach for manual switching between absolute and relative pen interaction. They introduce a *trailing widget* which was used for switching modes. In absolute mode, the cursor was right on the pens position. The trailing widget floats beside of the cursor avoiding to be hit by the cursor or interactive objects. Tapping in the trailing widget switches the input device to relative mode. Lifting the pen from the display or clicking a button on the pen returns to absolute mode. Without significant difference of selection time, the hybrid approach turned out to have a higher error rate (Hybrid = 6.8%, Absolute = 4.3%, Relative = 3.9%). The trailing widget was rated as distracting.

<sup>7</sup>Three targets in a row appear after the previous is successfully selected. For more details on the task please refer to Vogel and Balakrishnan [2005]

### Velocity Oriented Techniques

Velocity oriented approaches provide an automatic switch between different pointing modes. *Adaptive Pointing* attempts to improve pointing performance for absolute pointing device by implicitly adapting the CD-gain [König et al., 2009]. *Adaptive Pointing* relies on the optimized submovement model by Meyer et al. [1988]. A very common example is the widely used mouse-cursor acceleration. It is based on the idea that fast movements are used to overcome large gaps between cursor and target, while slow movements tend to compensate for over- or undershooting ([Balakrishnan, 2004] cited in König et al. [2009]). Commonly static velocity or acceleration functions are used for mappings. Frees et al. [2007] introduce a dynamically adjusted CD-gain between the hand and target in a virtual 3D environment. Evaluation studies have shown a significant reduction of error rate and completion using Frees’s PRISM technique.

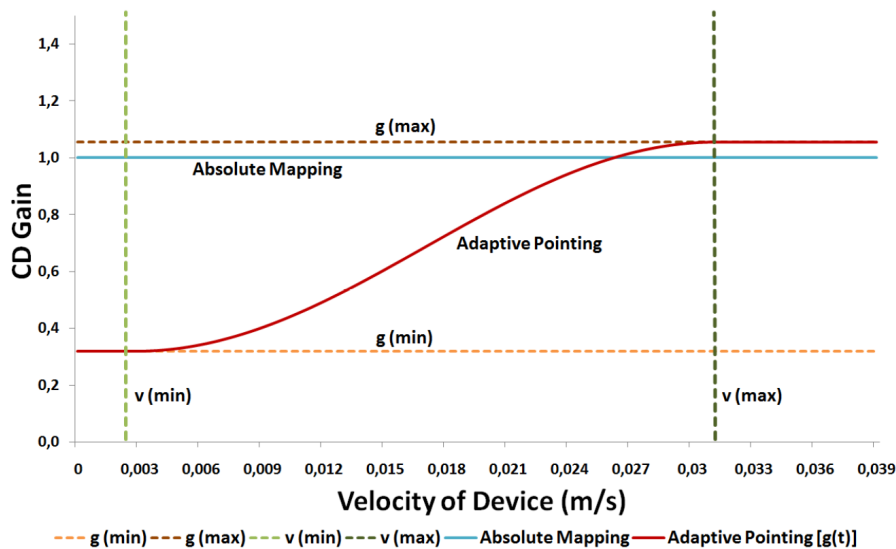


Figure 2.9.: Adaptive pointing: Adaptive pointing smooths the transition between relative and absolute CD-gain [König, 2010].

*Adaptive Pointing* attempts to simulate absolute pointing behavior. We therefore classify this technique as a hybrid approach. Compared to the switching methods mentioned previously, the user should not realize the gain variation using *Adaptive pointing* (Figure 2.9).

<i>Type of control</i>	Hybrid (rotational)
<i>Display Selection</i>	Good
<i>Large Displays</i>	Good
<i>Small Displays</i>	Good

In an experiment 24 participants produced 13'578 valid trials for analysis [König et al., 2009]. Bubbles appearing on random position in three different sizes ( $W_L = 80$  pixels,  $W_M = 40$  pixels,  $W_S = 20$  pixels) had to be clicked using a laserpointer. After each target the user had to dwell on a central homing position. Figure 2.10a shows a significant better result for adaptive pointing for small targets. Similar results have also been reported by Vogel and Balakrishnan [2005] for *RayToRelative* pointing (Figure 2.8b).

While absolute pointing techniques turned out to be error prone, relative pointing techniques require clutching mechanisms and can hardly be applied for display selection. Several hybrid approaches have shown better results for error rate and task completion time. With respect to the need of display selection, relative pointing does not fulfill the requirements for our scenario.

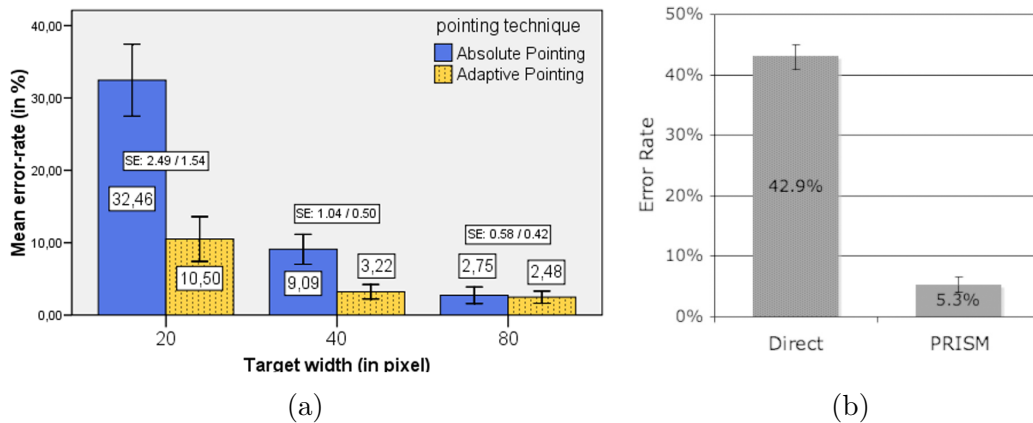


Figure 2.10.: Velocity Oriented approaches: (a) Adaptive Pointing [König et al., 2009] (b) PRISM [Frees et al., 2007]

## 2.4. Summary

In this chapter we defined several requirements relevant for our scenario. We defined five basic tasks to reproduce the three phases of the collaborative cycle. A digital environment needs to support various devices in order to support those tasks. Private mobile devices, large vertical devices, interactive touch-tables and more can help imitate the traditional workspaces. In order to support collaborative work in such environments we proposed a set of six MDE-requirements: *Privacy*, *Awareness*, *Shareability*, *Mobility*, *Reachability*, *Accuracy*, *Pointing and Selection Speed* and *Comfortable use*.

We found that gesture interaction has a high potential as it supports most of those requirements. The naturalness, expressiveness and mobility of gesture interaction

outbalance the negative factors like natural hand tremor and limited human motor precision. In order to reduce the cognitive load we found the number of static gestures (gestures which have to be memorized) should be reduced to a minimum. Symmetric-asynchronous bimanual interaction can replace static gestures with the use of more intuitive dynamic gestures (e.g. scaling and rotating objects). Using an ideal pointing technique those gestures can be performed more than an arm length away from a display. We assume that a hybrid approach for bimanual pointing show similar effect as for single-handed pointing.

The following chapter introduces our approach on bimanual cross-display interaction for multi-display environments.

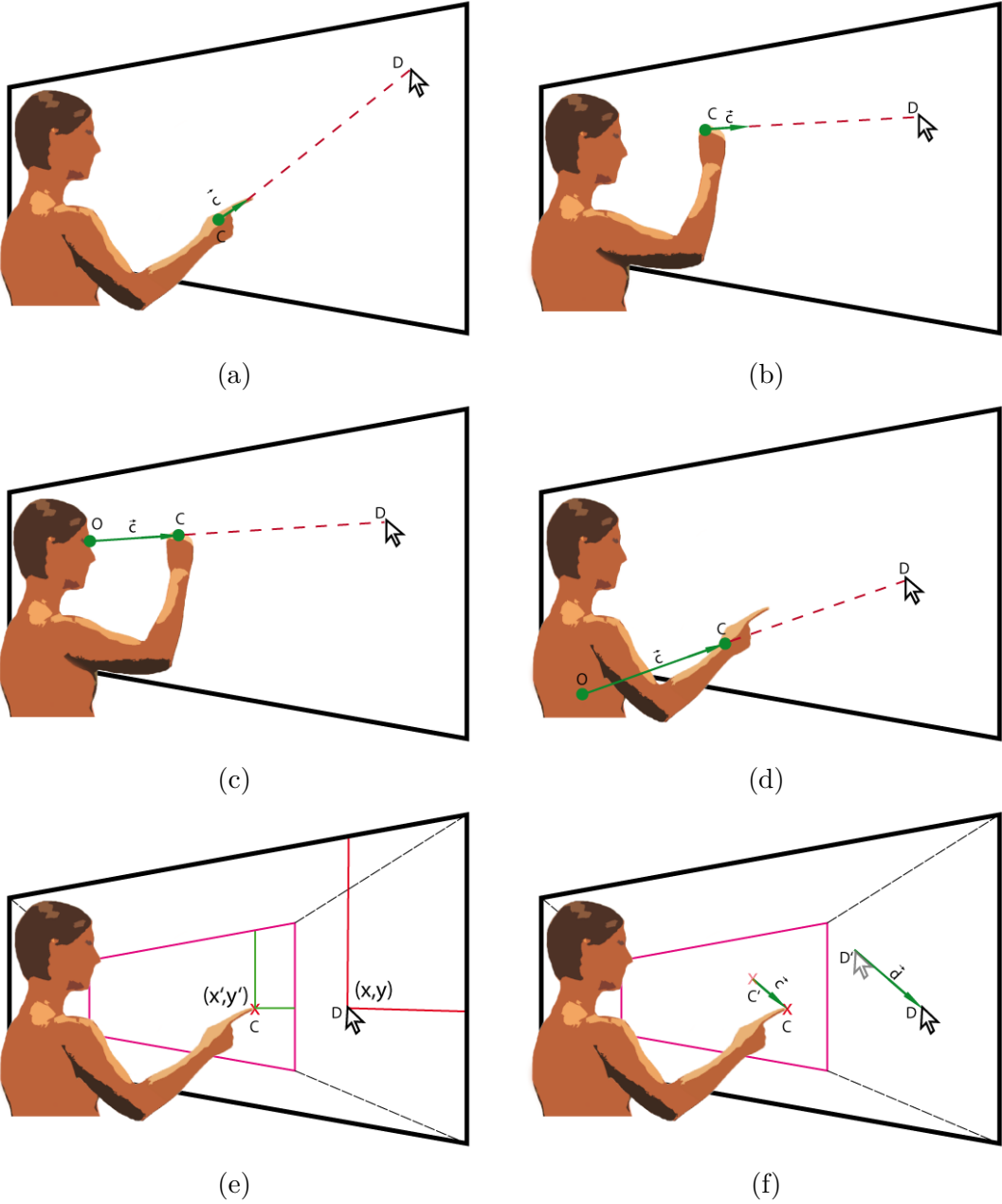


Figure 2.11.: Comparison of distant pointing techniques: (a) Raycast Pointing (b) Arrow-Pointing (c) Image-Plane pointing (d) Fixed-Origin Pointing (e) Virtual Plane Pointing (f) Relative Pointing



### 3. ${}^2\text{Hands}_4\text{Displays}$

Multi-display environments are commonly used in various situations. We defined several reasons for the use of bimanual pointing gestures. In this chapter we present the progress of the project  ${}^2\text{Hands}_4\text{Displays}$ . It combines the outcome of the research about collaborative MDEs (section 2.1), gesture interaction (section 2.2) and pointing interaction (section 2.3).

First we present the outline and an overall overview of the project in section 3.1. The development of the prototype contains three different sections. Section 3.2 presents the redesign of the tracking glove *Whitey* [Fohrenbach, 2009] which has been used in a previous project. *Optitrack*, a visual tracking system is introduced in section 3.3. Section 3.4 presents the integration of the visual tracking system into the interaction library *Squidy*. The visual prototype *Multidragger* (section 3.5) allows dragging objects from one display to another and also supports bimanual object manipulation. Finally the last section of this chapter introduces a novel approach on hybrid pointing which adaptively switches between rotational and positional pointing.

#### 3.1. Project Outline

At the beginning of the project we had a set of preset factors which define the frame for the entire project.

- *Optitrack*: An optical motion capturing system.
- *Whitey*: A cotton glove for gesture recognition.
- *Mediaroom*: A multi-display environment.

In the project background several projects from our work-group deal with multi-display interaction. *MedioVis 2.0* aims to provide a natural interface for seeking and exploring multimedia libraries [Reiterer et al., 2010], *Blended Interaction Design* investigates novel methods and techniques for creative interaction design activities [Geyer et al., 2011] or *PrIME* which investigates Primitive Interaction Tasks for Multi-Display Environments [Jenabi and Reiterer, 2010]. We assume that those projects could benefit from bimanual gesture interaction.

The research on the three required topics has been discussed in the previous sections (see figure 3.1 “Presets” and “Related Work”). Each of the preset factors can be mapped to one of the topics. *Whitey* needs to be redesigned for optimal bimanual interaction. The data from *Optitrack* are required for the calculation of the intersection points with one of the devices provided by the multi-display environment at the Mediaroom of the University of Konstanz (MRK).

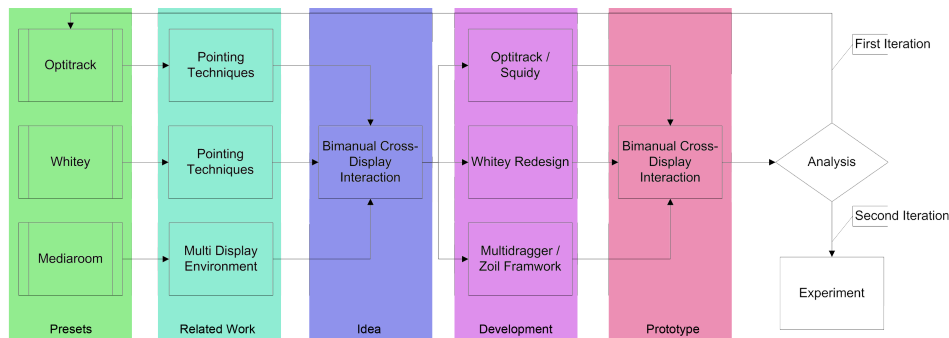


Figure 3.1.: Project Outline

We initially defined two goals for the project  ${}^2\text{Hands}_4\text{Displays}$ . Those two goals are the basic interactions required to complete the five tasks defined in section 2.1.

**Goal No 1:**  ${}^2\text{Hands}_4\text{Displays}$  transfers the drag-and-drop into 3D. It combines drag-and-drop with distant freehand pointing [Foehrenbach, 2009; Vogel and Balakrishnan, 2005] or Laserpointer interaction König et al. [2008] for cross-display object movement. Those pointing techniques have been successfully evaluated for large high-resolution displays.  ${}^2\text{Hands}_4\text{Displays}$  adapts those concepts for the use in MDEs such as in the MRK.

**Goal No 2:** The second concept introduced in  ${}^2\text{Hands}_4\text{Displays}$  is the transfer of the successful multitouch interaction into the three-dimensional space. Common multi-touch gestures like scaling or rotating objects on a table-top are mapped to bimanual freehand pointing.  ${}^2\text{Hands}_4\text{Displays}$  provides the technology to interact with any multi-touch enabled application running on Microsoft Windows 7. Touch-down events are simulated as if the user directly touches the display.

${}^2\text{Hands}_4\text{Displays}$  requires several hard- and software components. Gestures performed by the user are tracked by the visual tracking system *Optitrack*. Special gloves marked with retro-reflective markers can be recognized by the visual tracking system. The images from all cameras computed by *TrackingTools*. The *Squidy* interaction library computes the data from *TrackingTools* which are streamed over a network connection. All computation about gesture and pointing are performed by

squidy. *MultiDragger*, a GUI based on the ZOIL-Framwork is the last component in the chain. It allows drag & drop across multiple displays and supports bimanual object manipulation. The following sections 3.2-3.5 discuss those components in more detail.

As shown in figure 3.1 after the first prototype another iteration of the entire process was made. This step was necessary since first tests have shown problems in bimanual pointing interaction with the implemented pointing technique. Additional research and further refinements of the existing algorithms brought a new approach on distant pointing interaction. The last section (3.6) of this chapter focuses on the details of the novel approach which has been successfully tested in the experiment described in chapter 4.

## 3.2. Gesture Tracking

Capturing gestures using a visual tracking system such as Optitrack requires retro-reflective markers. Those markers reflect the infra-red light emerging from the cameras. “Whitey”, a tracking glove developed by [Foehrenbach \[2009\]](#), was designed for single handed interaction in a single display environment. The glove was used for pointing gestures and simple click gestures. Markers defining the rigid body were fixed on a base target which allows various individual marker sets. The markers for the fingers have been attached on screws which were glued inside the glove. This arrangement of markers was not comfortable for the user since the markers were loose and thus dangle while interacting.

Another quite similar approach from [Vogel and Balakrishnan \[2005\]](#) uses single markers for each finger. Each marker is fixed with an elastic strap around the finger. The markers for the base-target are mounted on an elastic plate on the wrist of the glove. The chosen arrangement of the markers and the use of a fingerless glove ensures the straps are attached on the best position for each user. This makes the glove very convenient to wear but requires recalibrating the system with every use. In addition these kinds of markers are very sensitive to skin-contact. Every time a user touches the markers while attaching them to a finger, grease and skin particles reduce the brightness of the marker.

The current context in the MRK requires bimanual gesture interaction. Multiple gestures need to be supported, especially grab-, click. and pointing gestures. For the creation of the perfect glove, advantages and disadvantages of previous models have to be considered. In addition the extended context of use has to be taken into account. Despite of accurate gesture recognition the following requirements have to be satisfied.

- Uncovered-Fingertips: Fingertips have to be uncovered in order to interact with touch-sensitive devices.
- Fixed markers: All markers have to be part of one single piece to reduce recalibration and touching single markers.
- Base-target: The base-target should be small and not disturbing.
- Light weight: The interaction device must be light and easy to fit.
- Sanitation: The glove must be washable since different users are wearing it during experiments and work.

For the redesign of *Whitey* the main focus was combining a fingerless glove with included markers. Especially the trade off between causality and tracking accuracy had to be considered. Markers which are attached tight are more comfortable whereas loose spherical-markers are better for recognition. Due to the fact that touching markers reduces its reflection quality and loose markers require time-consuming recalibration, each marker is placed on a single glove. This should ensure each marker keeps its position even when used by different persons.

For gesture recognition, the positioning of each marker is of great importance. Markers are likely to be very close to another marker while performing special gestures. If the distance between two markers drops below 20mm, both markers are recognized as a single marker. The result is that one of the two fingers cannot be recognized anymore which makes gesture recognition faulty. For this reason the position on which the marker is attached varies for each finger. Defining the best position for a marker depends on its importance and visibility. The most active finger for gesture recognition is the index finger. Since the middle finger is more likely to be hidden by other fingers its marker should be placed close to the fingertip. The difficulty is to maintain the action radius of the finger while keeping a minimal distance of 20mm between the markers. Therefore the marker for the index finger is positioned right before the second joint. Placing markers in front of a joint reduces marker movement. As illustrated in figure 3.3a the markers of middle and index finger have a sufficient distance even if both fingers are held together. The thumb and the small finger are easier to recognize due to their position. The ring-finger does not need a marker since its importance for gesture-recognition is very little. An elastic ribbon was inserted to avoid markers from shifting. To increase the comfort new semi-elastic clamps were made of Gorilla-Plastic<sup>8</sup>. A spherical marker is inserted to each clamp (see figure 3.2c).

The new concept for the base-target allows various target definitions. The side-view in figure 3.2c shows the three different layers of the base-target. The first layer

---

<sup>8</sup><http://www.gorilla-plastic.de>

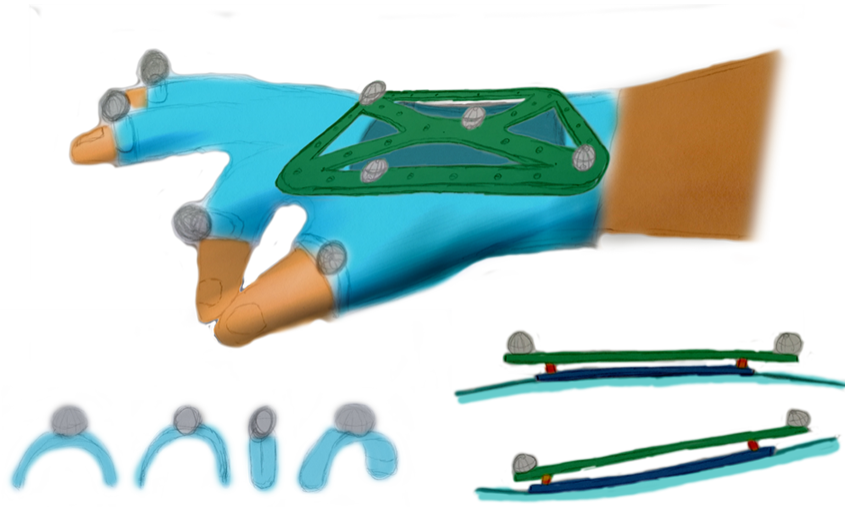


Figure 3.2.: Concept of a new *Whitey*: (a) Specialty formed clamps containing spherical markers are positioned on the ideal position of each finger. (b) Different perspectives of the semi-elastic marker-clamps. (c) Side-View of the base-target which remains still when the wrist is bending.

(blue) must be very flexible and is directly attached to the glove. The third layer (green) is a stiff alloy platform similar to the original Optitrack base. Markers can be arranged in various positions and fixed with the original thread pins. The middle layer connects the flexible first layer with the stiff third layer. This layer must be made of a semi-flexible material which allows the first-layer to bend, but still stiff enough to avoid dangling of the entire base-target. Unfortunately, the fabrication of the new base-target could not yet be completed. For the experiment the antler-like base-target as shown in figure 3.3b and 3.3c were used.

Replacing the dangling markers by comfortable semi-elastic marker-clamps turned out to be a good choice. Not only the better comfort, also better recognition results could be noticed. The cotton glove with the incorporated markers can easily be washed with water and soap. The use of disinfectant has not been tested since it could affect the reflective material of the markers. However washing with warm water and soap should satisfy the required sanitation. As for some people, the white color of the glove is associated with cleanliness, other people could associated the color of the glove with medical environments. For some users this can be irritating and additionally white gloves get dirty very quickly. As a future idea, colored gloves could help to identify personal cursors in a multi-user context. If the color of the glove corresponds with the visual representation on the screen any user can associate the user with the cursor (e.g. blue colored glove in figure 3.3c).

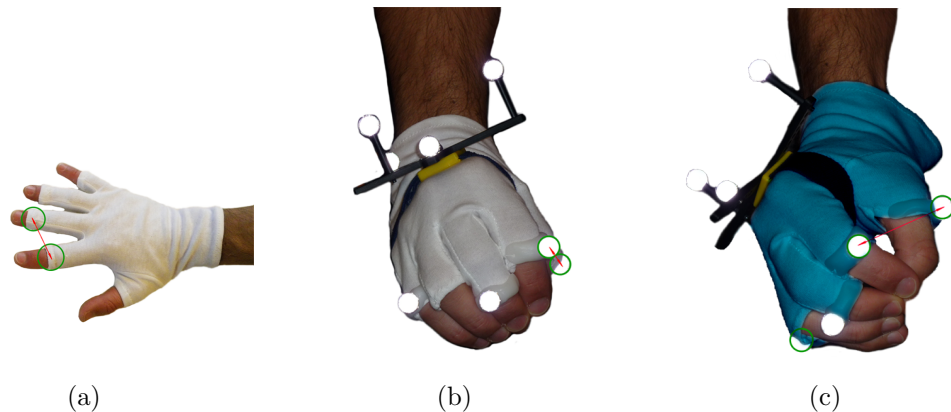


Figure 3.3.: Filter-issue in TrackingTools. (a) The relation between the blue hand and the white finger is correct for unfiltered trackables. (b) The lag of the filtered blue hand does affect the relation between the hand and the white fingers.

### 3.3. Optitrack

The redesign of *Whitey* results in a comfortable, multi functional glove with good recognition results. This section introduces *visual tracking* of gestures and the installation of the Optitrack system in the MRK. Using a low-budget tracking system for gesture recognition imposes several challenges which are discussed later in this section. For the reuse of this system, those factors should be considered since they cannot be found in the documentation of the system.

#### 3.3.1. Visual Tracking

The use of visual, non-contact tracking of hand- and body- movements has several arguments pro and contra as discussed previously. High flexibility versus high error rate and naturalness versus reduced precision have to be outbalanced. Even if such technologies remain challenging in the context of HCI, experience from earlier working-group projects (e.g. [Foehrenbach et al. \[2009\]](#)) have shown such challenges can be solved. The use of the already existing visual motion capturing system *Optitrack* was therefore well motivated.

Optitrack (OT) by NaturalPoint<sup>9</sup> is a low-budget motion capturing system which uses retro-reflective markers to reflect the infrared light emerging from each camera. The MRK is equipped with 12 OT cameras, each running at a frame rate of 100 frames per second. This allows to retrieve position, orientation and target-ID of multiple marker targets. The cameras are connected by USB-cables. This simplifies

<sup>9</sup><http://www.naturalpoint.com/optitrack>

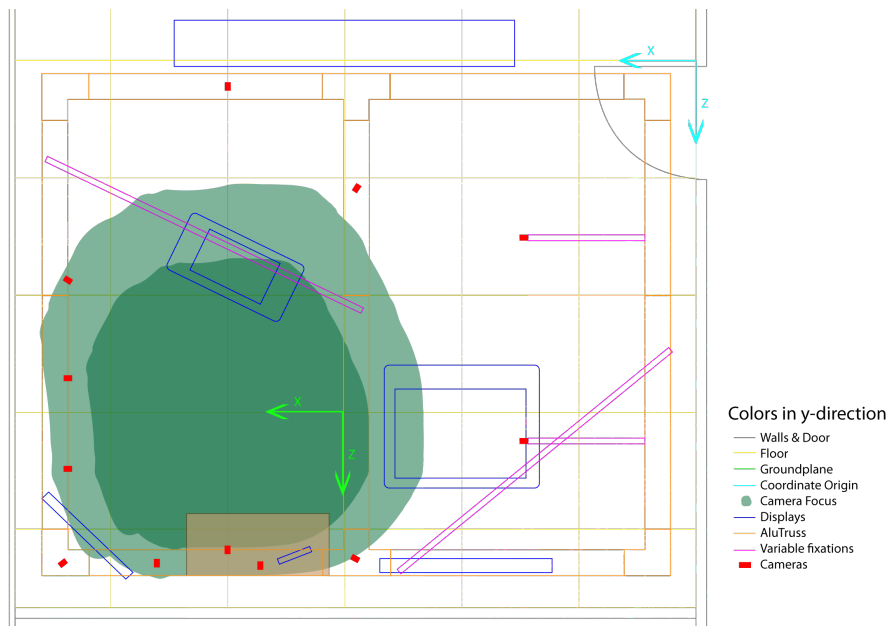


Figure 3.4.: Camera setup in the media room

the installation, but two major issues have to be considered. The infrared illumination needs additional power which is supplied by an ordinary USB-hub. In order to avoid loss of illumination power no extension cables should be used. Effects of bad cabling are reported in our project report [Fäh, 2011]. The second issue is about bandwidth. With the increasing count of reflective spots the amount of transmitted data increases. In a clean setting, each retro-reflective marker is recorded once by each camera, which is generally not a problem. Sunlight or other reflections (e.g. emerging infrared light from touch-tables) heavily increases the amount of data. In order to handle the incoming data, the cameras should be connected equally to each EHCI (Enhanced Host Controller Interface). A common personal computer has two EHCIs. Each can handle the full USB bandwidth. If too many cameras are connected to a single EHCI, the frame-rate is likely to decrease.

### 3.3.2. Installation

Another important issue is the arrangement of the cameras. The initial setup covered almost the entire ground surface (4m x 4m) of the MRK. This setup turned out to be inaccurate. Throughout several iterations the area has been adjusted in order to achieve best possible tracking result by maximizing the captured volume. The final setup, which was used for the experiment in chapter 4 uses 12 OT cameras to cover an area of  $5m^2$ . The size of the area depends on the required accuracy. If little

Table 3.1.: Optitrack compared with Vicon and AR-Tracking, the most common tracking system used for motion capturing [Fäh, 2011].

	<b>Optitrack</b>	<b>AR-Tracking</b>	<b>Vicon Tracking</b>
<b>Camera Name</b>	FLEX V100R2	ARTrack4	T160
<b>Resolution</b>	640 * 480	not specified	4704 * 3456
<b>FOV</b>	46.2 - 57.5	77.2 - 93.5	68.2 - 92.6
<b>Frame rate</b>	100fps	60fps	2000 fps
<b>Price</b>	\$599	aprox. \$6000	aprox. \$6500
<b>Used by</b>	Fäh [2011]	Vogel and Balakrishnan [2005]	Foehrenbach [2009]

accuracy is required the area can be increased to about  $10m^2$ . Two different camera types are currently installed (see figure 3.4).

**FLEX V100:** 8 pieces of the older version equipped with 4.5mm lenses cover each  $46.2^\circ$  FOV<sup>10</sup>.

**FLEX V100R2:** 4 pieces of the new version equipped with 3.4mm wide angle lenses cover a FOV of  $57.5^\circ$ .

Comparing OT with other tracking systems found in literature shows significant differences in performance (see table 3.1). Other tracking systems like Vicon<sup>11</sup> or AR-Tracking<sup>12</sup> provide more accurate tracking data. Higher resolution results in higher precision and a bigger field of view increases the capture volume and requires less cameras. In addition, higher frame rates allow better filtering since more data is available, which increases the responsiveness of the interaction. The limitations of a simple, affordable visual tracking system like Optitrack imposes several challenges which are mentioned later.

### 3.3.3. Software

Optitrack is distributed including the *TrackingTools* software and an API. The current software version 2.3.1 brought some major improvements compared to previous versions. Especially the calibration algorithms were improved, but it still has to be done very carefully.

- *Camera Placement:* Each camera requires overlapping with other cameras' FOVs. Even though experience has shown that placing cameras too close to each other affects the quality of the tracking negatively.

<sup>10</sup>Most tracking systems provides interchangeable lenses

<sup>11</sup>[www.vicon.com](http://www.vicon.com)

<sup>12</sup><http://www.ar-tracking.de>



Table 3.2.: Important values for 3-Marker-Calibration in TrackingTools [Fäh, 2011]

Value-Name	Optimal Value
Calibration Type	Full
Solver Scope	All
Optiwand	Standard
Sampling Speed	70%
Min Camera Coverage	30%

Table 3.3.: Important values for camera settings [Fäh, 2011]

Value-Name	Optimal Value
Exposure	55
Threshold	155
Illumination	15
Video Type	Precision Mode
Frame Rate	100%

- *Clean Setting:* Any reflective material should be removed or covered and no moving object or persons should be in the field of view (despite of the person calibrating the system). TrackingTools supports masking of reflecting spots, but this reduces the FOV of the camera and is therefore not recommended.
- *Equal Distribution:* During the calibration, the cameras record reference points from a special marker set which is moved through the capture volume. An equal distribution of the recorded reference points ensures good tracking quality. TrackingTools provide a *Spread-Value* for each camera during the calibration process. Spread-Values between 0.5 and 0.7 provide good tracking results. The higher the Spread-Value the better the tracking results.

In our project report [Fäh, 2011] we defined standard values which have been used for the setting. Table 3.2 shows the most important values for the calibration and table 3.3 shows the recommended camera settings for calibration and tracking.

Defining a trackable for the use with TrackingTools is simply done by holding the rigid body in the capture volume, selecting the markers with the mouse and pressing “create trackable from selection”. Trackables defined using the standard values from TrackingTools are likely to be misclassified. For rigid bodies with similar marker sets the values have to be adapted as shown in table 3.4.

Table 3.4.: Important values for trackable settings [Fäh, 2011]

Value-Name	Optimal Value
Trackable-ID	This value has to be equally set in Squidy
Min Marker Count	Depends on the number of markers $n$ and the size of the trackable. We experienced the best solution using $n - 1$ .
Min Hit Count	Between $n - 1$ and $n - 2$
Flexibility	20 – 40
Calculation Time	This setting depends on the CPU performance. We experienced good results for values between 10 and 15
Acquisition Delay	Between 2 and 5
Filter Position	0.2
Filter Rotation	2.2

### 3.3.4. TrackingTools Issues

In our project report [Fäh, 2011] we uncovered some issues using tracking tools. In order to understand some of the problems described later, we shortly summarize the most important issues and added recently encountered problems with the tool.

TrackingTools provides effective Kalman-filtering which can individually be applied for each defined trackable. The rotation and translation are filtered separately and allow to individually adjust the filtering. As discussed in section 2.3 lag is an important factor for pointing. Highly filtered data can also affect the recognition of gestures. Since single markers cannot be filtered, those markers are not affected by the lag. This effect is shown in figure 3.5. The white points representing unfiltered finger-markers are correctly positioned in relation to the hand (a). Whereas in (b) the lag affects the position of the hand and destroys the relation between finger and hand. This inconsistency of lag makes gesture recognition more difficult, since the relation between hand and finger is important.

The entire project setting can be stored in a single project file, which contains camera settings, calibration data and the definition of the trackables including all parameters of any trackable. Therefore also the filter-settings are stored with each trackable. When using the TrackingTools API, those filters cannot be applied even if the entire project file is loaded into the API. Therefore the use of the streaming option is recommended, even though it requires the TrackingTools software to run in the background.

TrackingTools allows the definition of unlimited trackables. Despite the fact that multiple trackables require high computational power, wrong recognition is more likely to occur since the placement of markers is limited on the provided base-targets. Trackables which are not in use should be disabled to avoid erroneous recognition.

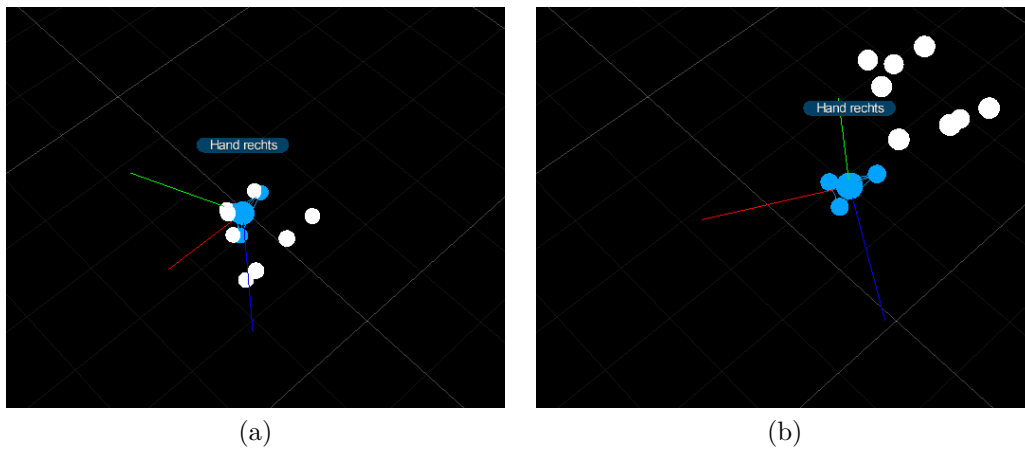


Figure 3.5.: Filter-issue in TrackingTools. (a) The relation between the blue hand and the white finger is correct for unfiltered trackables. (b) The lag of the filtered blue hand does affect the relation between the hand and the white fingers.

Using the streaming protocol only 9 trackables can be defined. When defining more than 9 trackables, no data will be streamed. This fact is not mentioned in the documentation and can cause unexpected errors.

### 3.4. Data handling

From the action of the user to the response of the system, data passes a chain of various components. Optitrack the first component was introduced above. Squidy, an interaction library developed at University of Konstanz was used for data handling. Squidy supports visual programming and provides a great variety of filters, in- and output devices which are ready to use. The simple drag and drop concept allows dragging several filters onto the workspace. Pipelines between the filters specify the data flow. This pipe-and-filter concept enables quick definition of a data flow.

Squidy supports different types of nodes. Input-nodes are typically used for input devices producing output in various data formats. Filter-nodes typically process incoming data from one or more connected input- or filter-nodes. Output-nodes perform actions based on incoming data (e.g. pointer visualization). For this project the following data are mainly used:

DataPosition2D *D2D* x- & y-coordinates (2DOF), attributes

DataPosition3D *D3D* *D2D* + z-coordinate (3DOF)

DataPosition6D *D6D* *D3D* + 9 double values for rotation matrix + 3 double values for rotation angles (6DOF)

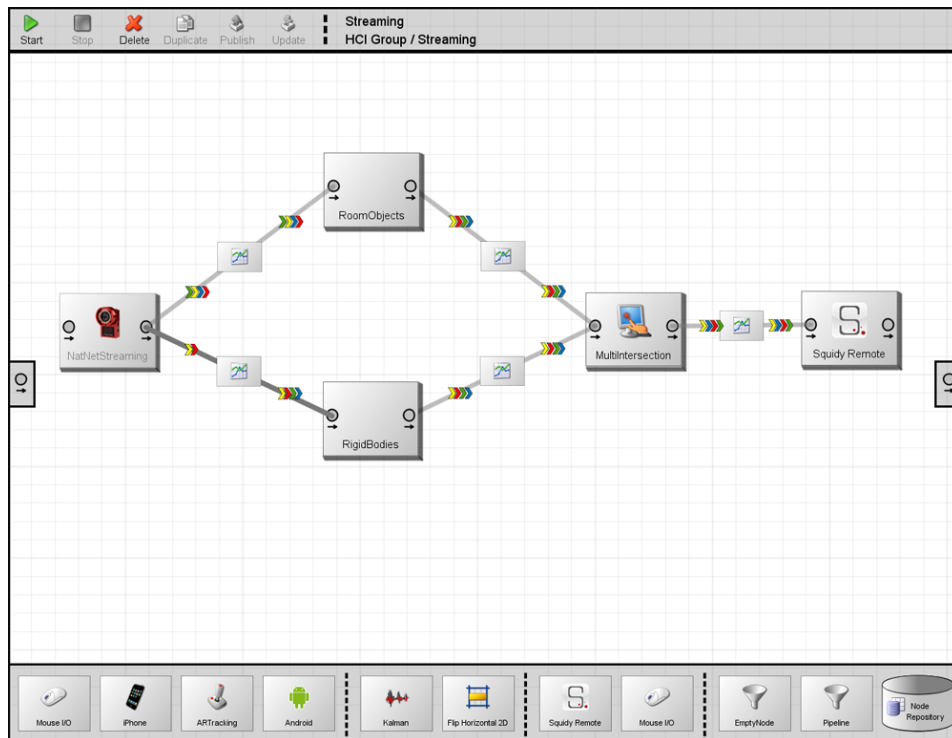


Figure 3.6.: Streaming Pipeline with sub-pipelines. The sub-pipelines are illustrated in figure 3.7.

In the following subsections we introduce the most important nodes which have been used for the experiment. For more detailed information about other nodes which can be used in combination with OT please refer to the project report [Fäh, 2011].

### 3.4.1. Data Fetching

TrackingTool supports data streaming and API calls to retrieve the data from the cameras. The first prototype used API calls via the Java Native Interface (JNI). JNI allows data flow in both directions. Calibration files or rigid body definitions can be sent directly to the API so the TrackingTool software is not required. Since the cameras can not actively send data via JNI, the data have to be fetched by Squidy. The refresh rate has to be adapted to the cameras which normally run on 100 frames per second. If the data are fetched too frequent, it is likely that several frames are processed multiple times. Too low refresh rates result in data loss.

First tests have shown the filter-settings did not apply well using the API (see section 44). Since those filters show great results in the TrackingTool's visualization we decided to implement another node for data fetching.

### NatNet-Streaming

NatNet streaming is TrackingTool’s own streaming protocol. It publishes the tracked data directly from the TrackingTool software and allows real-time changes in the setting. The streaming also requires a JNI but compared to the API, the streaming publishes the data continuously. This ensures that all frames are computed correctly. The *NatNet-Streaming* node requires only one property which is the dimension of the room. The incoming are normalized with the room dimensions and a rotation matrix is created from the quaternions (see equation 3.1). For each frame, a D6D is published containing the coordinates and the 3x3 matrix, which is stored in the 9 double values.

$$M_{rot} = \begin{pmatrix} 2 * (q_x^2 + q_w^2) - 1 & 2 * (q_x * q_y - q_z * q_w) & 2 * (q_x * q_z + q_y * q_w) \\ 2 * (q_x * q_y - q_z * q_w) & 2 * (q_y^2 + q_w^2) - 1 & 2 * (q_y * q_z + q_x * q_w) \\ 2 * (q_x * q_z - q_y * q_w) & 2 * (q_y * q_z + q_x * q_w) & 2 * (q_z^2 + q_w^2) - 1 \end{pmatrix} \quad (3.1)$$

### 3.4.2. Virtual Objects

The pipeline from figure 3.6a contains two different nodes for virtual objects. The *Rigid-Body* node, which defines properties for mobile objects and the *Room-Object* node defining stationary objects.

#### Rigid-Body

The *Rigid-Body* node attaches user-defined properties to the Dataposition6D which matches the specified property for the rigid body id. User specified attributes are added to the D6D. The most important properties for *Rigid-Body* node can be found in table 3.5.

Table 3.5.: Important properties for the *Rigid-Body* node

<b>Rigid Body ID</b>	A dropdown menu allows the user to choose from all defined rigid bodies.
<b>Role</b>	A <i>RigidBody</i> can either be a <i>Pointing Device</i> , a <i>Mobile Display</i> , a <i>Tracked Person</i> or a <i>Hand Gesture</i> .
<b>Pointing Mode</b>	The user can chose among three different methods for the intersection calculation. <i>Laser pointing</i> , <i>Relative pointing</i> or <i>Hybrid pointing</i>

The current version of *Rigid-Body* node supports different variants for the recognition of hand-postures. Gestures can also be defined as a rigid body in TrackingTools.

This allows accurate, but not very flexible hand-posture recognition. Different *roles* can be set for each *Rigid-Body* node. Several *RigidBodies*<sub>Role=HandGesture</sub> can be connected to one *RigidBody*<sub>Role=PointingDevice</sub> in such way, the gesture roles define the gesture which is applied to the RigidBody (e.g. Sub-Pipeline in figure 3.7d). Two other possibilities for gesture recognition required the *Static-Gesture Recognizer* node. In order to pre-filter incoming data for the *Static-Gesture Recognizer* node single markers represented as *D3D* in a predefined range are attached to a data container containing the *D6D<sub>Hand</sub>* of the hand and several *D3D<sub>Finger</sub>* of possible finger-markers. The TrackingTools filter issue specified previously complicate the elimination of non-finger markers since the positions of the filtered *D6D<sub>Hand</sub>* lags behind the position of the *D3D<sub>Finger</sub>*. This problem can be eliminated by defining two rigid bodies on a single target-base. One rigid body is filtered and defines the position and orientation of the hand, whereas the second rigid body is unfiltered and defines the area of the unfiltered finger-markers. This solution is illustrated in 3.7c where the unfiltered *Rigid-Body* node is attached to the *Static-Gesture-Recognizer* node. The upfollowing *Rigid-Body* node is the filtered definition of the hand. Gestures retrieved by the *Static-Gesture Recognizer* node are then attached to the filtered *D6D<sub>Hand</sub>*. Whatever solution is applied, the retrieved gesture ID is attached to the attributes of *D6D<sub>Hand</sub>*. The third possibility for static-gesture recognition requires only one *RigidBody* node and one *Static-Gesture Recognizer* node. The *Static-Gesture Recognizer* node is discussed in section 3.4.3.

### Room-Object

The *Room-Object* node specifies any object in the environment. It either is attached to a *Rigid-Body* node where it gets its coordinates from or the user specifies the coordinates. In the first case, the user has to specify the dimension of the object, the coordinates  $[x, y, z]$  in millimeters of three corners (Top-left, Bottom-left, Bottom-right) otherwise. Other properties such as IP and port can be set as well. An important factor for the calculation of intersection points is the Screen-Oversize property. It allows to virtually enlarge the object. This property (*P<sub>ScreenOversize</sub>*) is required by the *Multi-Intersection* node. Table 3.6 provides an overview of the most important properties of this node.

### 3.4.3. Filters

The two most important filter nodes are the *Static-Gesture Recognizer* node and the *Multi-Intersection* node. While the *Static-Gesture Recognizer* node attaches the id's of the recognized gestures, the *Multi-Intersection* node calculates display coordinates (*D2D<sub>Display</sub>*) from the data of the Hand *D6D<sub>Hand</sub>*.

Table 3.6.: Important properties for the *Room-Object* node

<b>Display Preset</b>	The positions of the current displays at the MRK are stored and can be chosen from a drop-down list. If the display is not pre-defined additional properties for the definition of the display are provided.
<b>Screen Oversize</b>	This value defines the virtual oversizing of the display (set in %). This function for example can be used to define the area where off-screen positions are displayed using the halos (see section 3.5).
<b>Backside Pointing</b>	This boolean value defines whether intersection from the backside of the display is computed or not. This can be useful if not the exact position, but only an object needs to be selected.
<b>Dimension / Position</b>	Additional several properties for position and size definition exist. Three corners of the display (Top-left, bottom-left and bottom-right) have to be specified for stationary displays. Mobile displays retrieve their position by a connected <i>Rigid-Body</i> node and therefore require only specifications for the dimensions.
<b>Remote Connection</b>	Two additional input fields (Host, Port) allow the specification of the remote connection. Those fields will be required by the prosecuting <i>Squidy-Remote</i> node.

### Static-Gesture Recognizer

The *Static-Gesture Recognizer* node calculates probabilities of a predefined static gesture based on distance of each finger to the hand origin. It can recognize static, semaphoric gestures such as *click*, *grab*, *right-click* or similar. All  $D3D_{Finger}$  in the incoming container are rotated to the normal position of the hand. Since the physiology of the hand reduces the upward finger-movement, all markers positioned higher than the pivot of the hand can be ignored. The incoming  $D3D_{Finger}$  are sorted from left to right. An octree algorithm efficiently retrieves the distances between pre-defined finger positions and the rotated  $D3D_{Finger}$  positions. The marker with the shortest distance to the optimal pre-defined distance is set to be the corresponding finger. The gesture with the lowest average deviation of the pre-defined finger positions are set to be recognized.

If the rotation- and / or translation speed of the hand exceeds the user-specified limit (specified in the properties, table 3.8), the last recognized gesture is continuously published. This mechanism was included to overcome the lag produced by the TrackingTools filter. It avoids false recognition during fast interaction movements. The calibration of the gesture file in listing 3.1 is performed automatically. Each

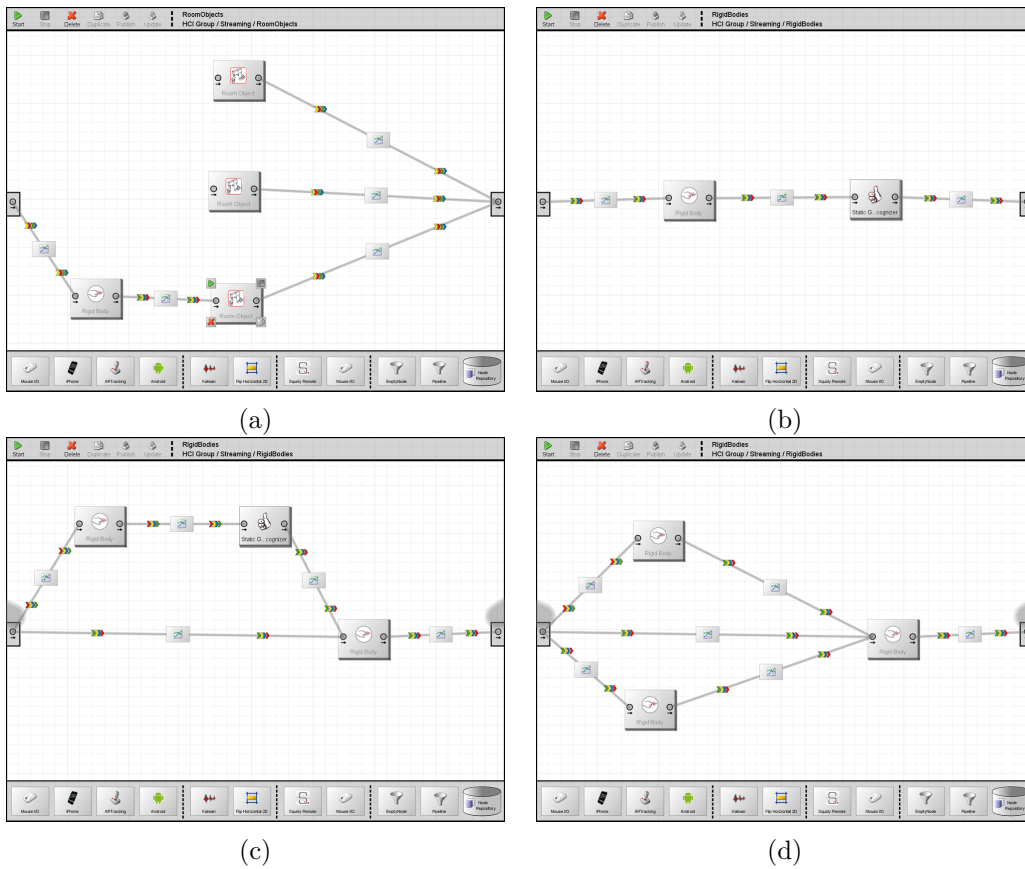


Figure 3.7.: Subpipelines for gesture recognition and object definition. (a) Defines 2 fixed room objects and one mobile. (b) A *Static-Gesture Recognizer* node which is attached to a *Rigid-Body* node. (c) One *Rigid-Body* node is used for the unfiltered position of the fingers and the other is used for pointing. (d) Two *Rigid-Body* nodes defining two pre-defined gestures are attached to the *Rigid-Body* node which is used for the pointing.

gesture can be assigned to a keystroke. During the calibration process the user performs the gesture while pressing the assigned keystroke. We experienced good result if the calibration process lasts about 3-5 seconds for each gesture. During the calibration process a truncated mean of the distance between  $D3D_{Finger}$  and  $D6D_{Hand}$  is calculated for each marker. Those values are stored in the gesture file for each gesture. For each finger a leaf in the octree is defined and assigned to its corresponding gesture. This simplifies the retrieval of gestures as proposed previously.

In order to provide good recognition results, the normal, relaxed posture of the hand is also defined as a default gesture. This increases the responsiveness since a gesture is set to be recognized if it has been recognized 5 times in a row by the octree algorithm. If a gesture already is performed the gesture remains recognized until another gesture is recognized 5 times or no gesture is recognized for 10 frames.



Listing 3.1: Gesture-Definition for a single gesture input

```

1 <?xml version="1.0" encoding="UTF-8" standalone="no"?><gestureset>
2   <gesture eventID="1" gestureID="6" gestureName="rightClick" handSide="1">
3     <finger fingerIndex="1" fingertype="thumb" x="-33.2203" y="-32.8740" z="-13.2723"/>
4     <finger fingerIndex="2" fingertype="index" x="18.8051" y="-14.8207" z="51.0753"/>
5     <finger fingerIndex="3" fingertype="middle" x="30.2969" y="-24.2331" z="23.2166"/>
6     <finger fingerIndex="4" fingertype="small" x="37.6171" y="-50.4728" z="-65.2396"/>
7   </gesture>
8   <gesture eventID="5" gestureID="7" gestureName="rightSinglePointClick" handSide="1">
9     \dots
10  </gesture>
11 </gestureset>

```

Therefore the definition of the normal posture requires less time to un-set a gesture. Table 3.7 illustrates a proposed set of gestures. As mentioned previously, increasing numbers of semaphoric gestures increase the cognitive load. Therefore the number of gestures should be kept as little as possible.

### Multi-Intersection

In the first prototype we introduced the *Intersection-Point* node [Fäh, 2011]. The *Intersection-Point* node calculates the intersection of one *Room-Object* node and several *Rigid-Bodies*. Despite of the new calculation algorithm (see section 3.6) several problems from the first test have been concluded in the extended *Multi-Intersection* node. The *Multi-Intersection* node can compute the intersection for several *Rigid-Body* nodes with multiple static or mobile *Room-Object* nodes. Since the *Room-Object* node contains properties for IP and port, the data can correctly be addressed to the corresponding device.

The *Multi-Intersection* node computes intersection points ( $D2D_{Display}$ ) for each connected *Rigid-Body* node with all *Room-Object* node. Three different pointing modes are available.

- Absolute Pointing is defined in analogy to the ray cast pointing [Vogel and Balakrishnan, 2005].
- Relative Pointing is a mapping function based on the position of the users hand.
- Hybrid Pointing is an adaptive weighted combination of the two previous pointing mode.

Those pointing modes and the corresponding algorithms are discussed in section 3.6. The absolute pointing mode does not require any properties to be adjusted. Hybrid and relative pointing require a set of properties which significantly change

Table 3.7.: Minimal set of semaphoric gestures defined for the use in *MultiDragger*









Image	Definition / Function
	<p><b>Normal Posture:</b> The normal, relaxed position of the hand is used for pointing, without performing an interaction. This normal posture can also be defined as a gesture.</p> <p><b>Iconic Visualization:</b></p>  <p>Left    Right</p>
	<p><b>Left-Click:</b> The left click gesture as proposed by <a href="#">Foehrenbach [2009]</a>. This gesture is generally applied to a mouse-click or a touch-down event.</p> <p><b>Iconic Visualization:</b></p>  <p>Left    Right</p>
	<p><b>Grab-Gesture:</b> We defined the grab gesture as grabbing more than a single object. It can be used for operations like holding an object or grab the entire display content.</p> <p><b>Iconic Visualization:</b></p>  <p>Left    Right</p>
	<p><b>Right-Click:</b> As the gesture interaction can also be mapped to a simulated mouse device, the right-click gesture can be mapped to the right-button of the mouse. Opening a context-menu would be a typical example for the use of this gesture.</p> <p><b>Iconic Visualization:</b></p>  <p>Left    Right</p>

Table 3.8.: Important properties for the *Static-Gesture Recognizer* node

<b>Gesture Definition File</b>	Path to the XML-File where the gesture definitions are stored.
<b>Simulate Mouse Buttons</b>	With this option enabled, digital values for mouse buttons are attached to the data. In combination with the <i>Multi-Intersection</i> node this allows direct control over the mouse.
<b>Maximum Target Translation</b>	Specifies the maximum distance (in millimeters) a target can move in a single frame, before the gesture recognition is set on hold. As described above, this option helps to avoid gesture-misclassification.
<b>Maximum Target Rotation</b>	Specifies the maximum angle a target is allowed to rotate in a single frame.

the pointing behavior. Detailed information of those properties are mentioned in table 3.9.

### Squidy-Remote

For the distribution of the data from *Multi-Intersection* node over network, the common *Squidy-Remote* node had to be adapted. If incoming data contain the attributes “Host” and “Port” *Squidy-Remote* node sends those data using OSC-connections. Since data for multiple displays can arrive, *Squidy-Remote* node keeps the connection open for the next data. With this additional option, the *Squidy-Remote* node can broadcast any data to the corresponding client simultaneously.

## 3.5. Multidragger

We previously discussed the first three main components required for bimanual gesture interaction. Optitrack cameras capture gestures performed by a user wearing the tracking gloves. Capturing data is piped through the filters and nodes in Squidy. At this point, we have position and gesture of each hand. The next step is the a visual interface on each display which can handle the gesture input.

In this section, we present *MultiDragger*, an application based on the ZOIL-framework which supports bimanual gesture input and cross-display interaction. ZOIL (Zoomable Object-Oriented Information Landscape) provides several basic functions required

Table 3.9.: Important properties for the *Multi-Intersection* node

<b>Force Pointing Mode</b>	This property can overwrite the pointing mode property set by any <i>Rigid-Body</i> node. This allows simple switching between modes and all device operate in the same mode. If this property is not set, each <i>Rigid-Body</i> node operates in its individual mode.
<b>Properties for hybrid pointing</b>	
<b>Min Velocity Threshold</b> $v_{uniMin}$	Defines the lower velocity threshold. Hand movements performed slower than $v_{uniMin}$ ( $\hat{v}_{uni} < v_{uniMin}$ ) are generally executed in relative pointing mode. (For exact definition of the algorithm please see section 3.6.)
<b>Max Velocity Threshold</b> $v_{uniMax}$	Hand movement with Velocity values higher than defined by this property ( $\hat{v}_{uni} > v_{uniMax}$ ) are executed in absolute pointing mode (Depending on other properties).
<b>Min Absolute Ratio</b> $v_{absMin}$	A minimum threshold for the percentage of absolute pointing smooths transitions from relative mode to absolute mode. If the value is set above 5% the pointing for slow movements behaves like on a rubber-band. Ideal values are between 0% and 2%.
<b>Max Absolute Ratio</b> $v_{absMax}$	This property allows to specify the maximum percentage of absolute pointing. This value can be set up to 150% since it also defines the maximum prefetching ratio.
<b>Min Translation Threshold</b> $v_{transMin}$	The minimum velocity threshold applied for the relative pointing algorithm.
<b>Max Translation Threshold</b> $v_{transMax}$	In analogy to $v_{transMin}$ the maximum value defines the upper threshold.
<b>Prefetching Threshold</b> $T_{prefetch}$	Sets the threshold for applying the prefetching algorithm. If this value is set higher than the <i>Maximum Absolute Ratio</i> prefetching is disabled.
<b>Properties for relative pointing</b>	
<b>Min Relative Threshold</b> $v_{relMin}$	The minimum velocity threshold applied for the relative pointing algorithm.
<b>Max Relative Threshold</b> $v_{relMax}$	In analogy to $v_{relMin}$ the maximum value defines the upper threshold.

for cross-display interaction. The functions supported in *MultiDragger* are kept simple. *MultiDragger* displays objects or images which can be manipulated and dragged from one display to another.

### 3.5.1. Data Interface

The first step is connecting *MultiDragger* to Squidy. An early prototype used TUIO-commands with the goal to simulate touch-down events. The initial idea was to use pointers to indicate the position of the hand and as soon as the user performs a gesture, a simulated touch-down event performs the interaction as if the user directly touches the surface. The main advantage of this approach is that any touch-sensitive application including Windows 7 could be controlled by bimanual gesture interaction without the need of any additional plug-in or coding. The TUIO-commands which were computed by Squidy were transmitted via OSC to Multi-Touch Vista. Multi-Touch Vista<sup>13</sup> is a user input management which can be used to simulate touch-events. Unfortunately the component-chain of Squidy, TUIO, Multi-Touch Vista and the operating system was very error prone. Finger-Id's were switched and unintended touch-up events made a smooth interaction impossible. The debugging of the component-chain which includes several black boxes turned out to be very difficult and time consuming. As for this project, gesture interaction on the layer of the operating system was not required, we decided to handle the gesture input on the software layer.

As ZOIL was originally designed for the Microsoft Surface an interface for simulating contact-events was required. Handling the gesture translation on software layer only limits the portability to other applications. Any ZOIL-Client can combine mouse-interaction, touch-interaction and also the simulated touch-interaction simultaneously. The *Bridge-Plugin* for ZOIL retrieves data from the ZOIL-Bridge and transforms the gesture data to contact events. Listing 3.2 shows an extract of the main steps from retrieving the hand and gesture id's to the transformation of the Surface-Events.

The use of the Bridge-Plugin allows the re-use of bimanual manipulation gestures which are already defined in ZOIL. A second plugin was directly included in the core application. This interface controls all commands for the GUI. For example the control of the hand-shaped cursors which give visual feedback on the performed gesture (see table 3.7). This plugin can be added to any ZOIL-Application and supports gestures with little programming effort.

---

<sup>13</sup><http://multitouchvista.codeplex.com>

Listing 3.2: Bridge-Plugin for gesture input

```

1
2  if ((String)d2d.Attributes["KEYWORD"] == "ONSCREEN")
3  {
4      rigidBody = (int)d2d.Attributes["RIGIDBODYID"];
5      handSide = (int)d2d.Attributes["RIGIDBODYID"];
6      gestureID = (int)d2d.Attributes["GESTUREID"];
7      identifier = (int)d2d.Attributes["GROUP_ID"];
8      SurfaceEventId sfEventID = SurfaceEventId.UnknownEvent;
9      if (gestureID >= Gestures.CLICK && tmpGesture != currentHand.Gesture)
10     {
11         if (currentHand.gestureChanged)
12             sfEventID = SurfaceEventId.ContactAdded;
13         else
14             sfEventID = SurfaceEventId.ContactUpdated;
15     }
16     else
17     {
18         if (gestureID == Gestures.DEFAULT || gestureID == 0)
19         {
20             if (currentHand.gestureChanged)
21                 sfEventID = SurfaceEventId.ContactRemoved;
22         }
23     }
24 }
25 if (sfEventID != SurfaceEventId.UnknownEvent)
26     this.RaisePacketEvent(rigidBody, gestureID, sfEventID, new PointF(((float)d2d.X
27         ), ((float)d2d.Y)), identifier, 3f);
}

```

### 3.5.2. Landscape Arrangement

ZOIL provides a zoomable landscape for object placement. The zooming option has been disabled for the current application. The landscape allows the placement of an individual view port for each display. It has to be large enough to place all required viewports. If the gaps between the viewports are too small, large objects dragged close to the display border are likely to appear on another display (e.g. the yellow star in figure 3.8b). ZOIL uses DB4O<sup>14</sup> (Database for objects) to synchronize all objects on each client. The size of the viewport also defines the size of the object. The viewport can be defined in such way, that the size of an object in millimeters is the same for each display. In most application high DPI results in smaller icons and buttons. We therefore decided to define the viewport in pixel size equal to the display resolution.

The illustration in figure 3.8b show a planar representation of the displays. The position of the displays in the room is independent of the viewport definition. As long as the user is pointing at a display the hand-cursor indicates the pointing-position. As soon as the user pointes outside the displays, the pointer is replaced by a halo. A halo is a circle which indicates direction and distance of the pointing-position according to its size and the position of its origin [Xiao et al., 2011]. Figure

<sup>14</sup><http://http://www.db4o.com/>

3.8 shows a user pointing between the two displays. The small circle on the small screen indicates the pointing-position is close to the top-right corner, while the large circle on the large screen indicate a pointing-position somewhere to the left to the display with a large distance.

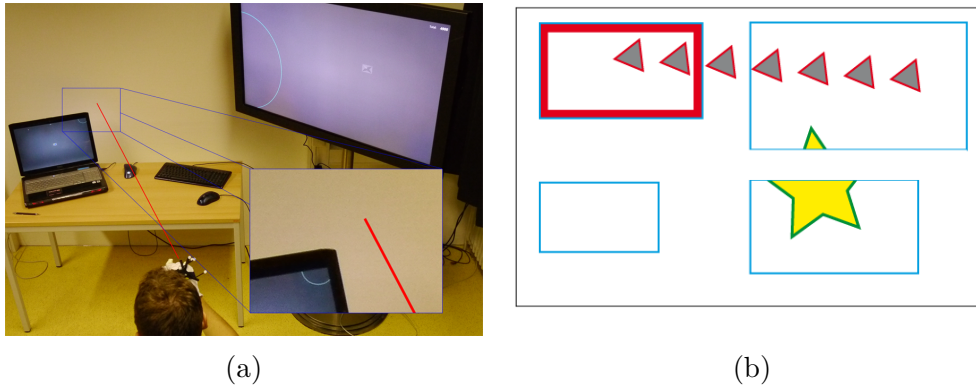


Figure 3.8.: (a) A halo indicates the direction of the users off-screen pointing. (b) On a single ZOIL landscape multiple displays containing several objects can be arranged.

### 3.5.3. Cross-Display Object Movement

*MultiDragger* supports single- and twohanded interaction. The very basic concept of drag & drop can be performed over display boundaries. An object is set *dragged-out* if the user is dragging an object over the boundaries. A drawback of the touch-simulation is that dragging out by touch is not possible since touch-events can not be set outside the display. A display margin on each side of the display could be defined. If an object is dragged into this margin the object could be set to *dragged-out*. Since distant pointing allows very fast interaction and the refresh rates of touch-input is limited the cursor can traverse large gaps in a single refresh rate. Therefore the display-margin is not a good solution since objects are likely dragged over the margin without having a refreshed position inside the margin. In figure 3.8b the triangle is dragged from on display to another. The refreshing rate of the system never shows a triangle in the red marked margin. To overcome this problem we defined several states for each object. For each state hand-ID and gesture-ID are stored with the object. The following states have been applied.

- Dragging: The user grabs an object. As long as the user holds the object this state is applied.
- Idle: An object has been released or never been grabbed.

- **Dragged-Out:** If an object has been set to “Dragging” and the hand which dragged the object is not on the display anymore, then the object is set to “Dragged-Out”. The object will be removed from the display.
- **Dragged-In:** If the hand which previously was dragging an object enters a viewport performing the same gesture as during the drag, then the object to which the hand is assigned is placed at the hands position and set to “Dragging”.
- **Manipulating:** The object is grabbed with both hands.

For drag & drop interaction the DB4O turned out to be too slow. A direct communication with OSC-Messages between the clients is used to synchronize the objects, object states and hand postures.

Drag & drop is a typical task to perform with one pointing device. Scaling and rotating object with a single pointing device requires bounding boxes or similar aids. Using the simulated touch-events the user can perform common scaling and rotating gestures as known from multi-touch devices. In order to perform a bimanual task, the user grabs the object with both hands. If one hand is released the object returns to the “Dragging” state.

#### 3.5.4. Datarecording

For the experiment described in chapter 4 a tool which collects data for the analysis is required. Since multiple displays, cursors and objects are involved the data recording has to be synchronized carefully. For each hand a synchronized object storing the data as shown in table 3.10 was created. The storage objects are updated with each incoming data sample.

Additionally a storage object is created for each object on the screen (Table 3.11). The continuously stored data are logged after reaching an important phase of the interaction.

- *Initialize:* Initial Position of all objects and hands when clicking the homing position.
- *First grab:* The source-object has been clicked the first time and the interaction has started. The data contains information about time, movement, overshooting or clicks outside the object. This data is important to evaluate data from the standard homing position to the first grab of the source-object.
- *Level 1:* The position and orientation of the source-object in relation to the target-object has reached a pre-defined value. This value indicates a minor accuracy level. Each time the user reaches the accuracy defined for level 1,



Table 3.10.: Data stored for each hand-controlled cursor

<b>Interaction Time</b>	The time in milliseconds is stored for each interaction step.
<b>Movement X</b>	Movement of the cursor on the screen in direction of the x-Axis. (in pixels)
<b>Movement Y</b>	Movement of the cursor on the screen in direction of the y-Axis. (in pixels)
<b>Movement</b>	Cursor movement on the display. (in pixels)
<b>Object Hit Counter</b>	Each time the cursor moves over an object, the counter is increased by 1.
<b>Click Counter</b>	Each click gesture is counted.
<b>Off Click Counter</b>	Each click gesture which does not hit an object is counted.
<b>Hit and Click Counter</b>	Each click which grabbed the object is counted.
<b>Overshoot Counter</b>	Each time the cursor is moving out of the object without any action performed the overshoot counter is increased by 1.
<b>Overshoot Time</b>	The time between the overshooting and the next successful interaction is logged (in milliseconds).
<b>Overshoot Hit</b>	The number of times the object has been hit between two successful interactions is counted.
<b>Overshoot Click</b>	The number of clicks between two successful interactions is counted

the data is logged. This level can be logged several times since the accuracy can drop again below this level. Level 1 is indicated by a orange icon on the source-object.

- *Level 2:* The accuracy of the source-object has reached the predefined values for Level 2. Also level 2 can be reached several times during an interaction cycle. Level 2 is indicated by a green icon on the source-object. If level 2 is reached, the user accuracy is good enough to drop the object.
- *Completed:* The object has been dropped at the final position.
- *Timeout:* The interaction exceeded the time limit defined for the task. A red icon indicates the user to drop the object.

This automatic summarizing of data allows faster data analysis, since only the important steps during the interaction are logged. However it requires an exact analysis

about what data are required for the experiment. We decided to choose this kind of data recording since it facilitates the synchronization between the displays and reduces the amount of data.

Table 3.11.: Data stored for each object on the screen

<b>Interaction Time</b>	The time passed since the first click on the object
<b>Transition Accuracy</b>	The transition accuracy is defined as the sum of the offsets of x- and y-axis. For example, if the target-object is positioned on P(500,500) and the source-object is dragged to the position P(502,496), the accuracy value is 6. The lower the value the better the accuracy.
<b>Rotation Accuracy</b>	The difference between the rotation angle of the target- and source-object in degrees.
<b>Size Accuracy</b>	The difference between the dimension of both objects on x- and y-axis.
<b>Object Movement</b>	The distance an object has been moved. (in pixels)
<b>Object Movement X</b>	The distance an object has been moved along the x-axis on the screen. (in pixels)
<b>Object Movement Y</b>	The distance an object has been moved along the y-axis on the screen. (in pixels)
<b>Object Rotation</b>	The sum of all changes of the rotation are stored. (in degrees)
<b>Object Width</b>	The sum of all scaling actions in direction of the x-axis of the object is stored. (in pixels)
<b>Object Height</b>	The sum of all scaling actions in direction of the y-axis of the object is stored. (in pixels)

### 3.6. Position-based Hybrid Pointing

In section 2.3 we found several advantages of hybrid pointing techniques. Absolute pointing is required for display selection. However for small devices and small targets absolute pointing is supposed to be inaccurate. In addition absolute pointing was another drawback when used in a bimanual context. If absolute pointing is applied in a bimanual context, an unintended left-right switch is likely to occur. This effect cannot be observed in a single-handed context. This problem is illustrated in figure 3.9a. On the left side of the illustration, both hands are pointing to the display. The ray's for the absolute pointing do not cross each other. On the right side of figure

**3.9a** the left hand is pointing to the right and the right hand to the left. The left hand cursor therefore is positioned right of the right hand cursor (green arrows). The user will likely associate the cursor on the left cursor with the left hand (blue arrows). This can lead to confusion. For several reasons, this side-switch can be done on purpose. For example if the user holds an object with the left hand and wants to grab another object or select a menu with the left hand, which is positioned on the left border of the display. Nevertheless, we found that this effect is likely to happen unintended. Especially if both cursors are close to each other. Even though each cursor-icon represents either a left- or a right hand (see table 3.7), it can be very difficult to distinguish between the cursors. Also for experienced users, the right hand is linked with the the cursor positioned on the right side (marked as blue rays in the illustration 3.9a). This effect reduces the intuitiveness of the pointing and can be frustrating. We assume that a hybrid pointing technique where the positional information of the hand is taken into account could help to prevent the side-switching effect.

Another difference between bimanual and single-handed pointing is the difficulty of hand synchronization. During early tests with members of our working-group, we observed different interaction patterns. While some users can control both cursors simultaneously others position one cursor after another (synchronous vs. asynchronous [Ulinski et al., 2009]). This requires to hold the first positioned cursor still at a position until the second cursor is positioned. Holding a cursor still is very difficult when using absolute pointing [König, 2010]. It is very likely the user requires several steps to position both cursor iteratively, one by one. In a manipulation task this reduces the efficiency of the interaction and also can cause frustration.

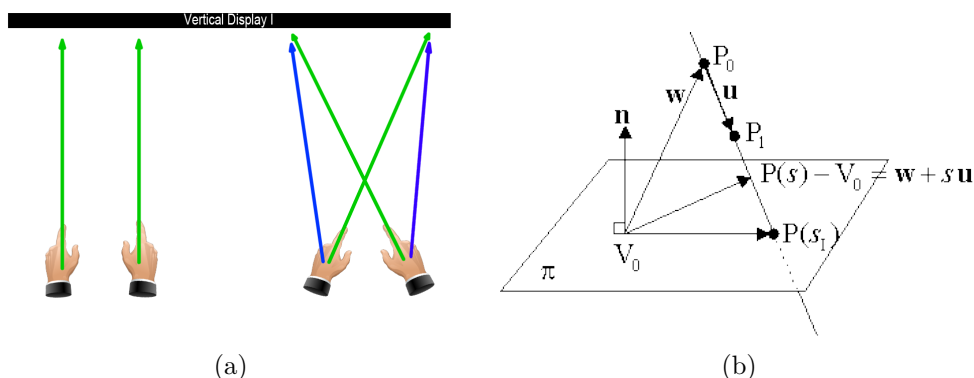


Figure 3.9.: (a) Cursor-crossing effect: The virtual representations of the cursor for absolute pointing do not agree with the users intention. (b) Definition of an intersection point on a plane.

To overcome those difficulties in order to provide accurate and intuitive bimanual pointing interaction we found the hybrid approach proposed in Vogel and Balakrish-

nan [2005] could be a good base. The adaptive gain proposed for *Adaptive Pointing* [König et al., 2009] does not consider how the interaction device is moved. We assume that the side-switching problem cannot be solved without information about the position of the hands. Instead of adaptively adjusting the CD-gain, a method which adaptively switches between absolute and relative pointing can replace the manual switch proposed by Vogel and Balakrishnan [2005]. An adaptive switching requires less semaphoric gestures and should therefore reduce the cognitive load of the user [Fikkert, 2010].

We therefore propose *HyPoba Pointing*, a hybrid position based technique which adaptively switches between rotational-absolute and positional-relative pointing. This approach is based on a combination of the techniques proposed by Vogel and Balakrishnan [2005] and König et al. [2009], with a special focus on bimanual interaction. We propose a weighed combination of absolute (rotational) and relative (positional)<sup>15</sup> pointing. This technique combines the direct *absolute pointing* for fast interactions and accurate *relative pointing* for slow and precise interaction. In analogy of *Adaptive Pointing* we assume that slow hand-movements are done with the intention of interacting precisely while fast movements are performed to overcome larger distances. The weighting is based on different velocity functions.

### 3.6.1. Velocity Functions

Like described above, fast hand movements are mapped to absolute pointing and slow, exact movements are mapped to relative pointing. This allows the user to quickly drag items from one display to another, while supporting high precision for tasks which are executed slowly. We define three different functions for the appropriate calculation of the interaction velocity.

**Transition velocity**  $v_{trans}$  is defined as the velocity of the movement of the hands origin among all three axes. This velocity measure is rotation independent and is used for the calculation of the transfer function for relative pointing. For each axis we define  $\Delta(t)$  as the movement of the hand along one axis between frame  $f_t$  and  $f_{t-1}$  where  $t$  denotes the current time.

$$\Delta x(t) = |x_{hand}(f_t) - x_{hand}(f_{t-1})| \quad \text{|| in millimeters} \quad (3.2)$$

To avoid outliers affecting the result we use a 10% truncated mean. In order to keep a high responsiveness we found trimming a number of  $n = 5$  samples

<sup>15</sup>In the following the term relative is related to the relative movement of the hand and is not affected by the rotation of the hand.

## 3.6. Position-based Hybrid Pointing

by 10% (  $z = \frac{10}{n}$  ) provides acceptable results. To calculate the  $v_{trans}$  we normalized the movement vector of the hand by the time between two frames.

$$\bar{v}_{trans}(t) = \frac{1}{n - 2z + 1} \sum_{i=z+1}^{n-z} \frac{\sqrt{\Delta x_i^2 + \Delta y_i^2 + \Delta z_i^2}}{time(t_i) - time(t_{i-1})} \quad (3.3)$$

To simplify further data handling the trimmed velocity  $\bar{v}_{trans}$  is normalized by a user defined minimum and maximum values. Those values can be defined in the *Multi-Intersection* node (Table 3.9). During the experiment the values were set to  $v_{transMin} = 1$  and  $v_{transMax} = 9$ .

$$\hat{v}_{trans}(t) = \begin{cases} 1 & \text{if } \bar{v}_{trans}(t) > v_{transMax} \\ 0 & \text{if } \bar{v}_{trans}(t) < v_{transMin} \\ \frac{\bar{v}_{trans}(t) - v_{transMin}}{v_{transMax} - v_{transMin}} & \text{otherwise} \end{cases} \quad (3.4)$$

**Rotation velocity**  $v_{rot}$  is defined as the angular speed of the hands rotation. This function is not affected by translations of the hand. We define the angular notation in the following list according to the *NASA Standard aeroplane*<sup>16</sup>

- $\phi$  heading = rotation about y-axis applied first
- $\theta$  attitude = rotation about z-axis applied second
- $\psi$  bank = rotation about x-axis applied last

The following equations 3.5 - 3.7 show the calculation of the trimmed angular speed similar to the calculation of  $v_{trans}$ .

$$\Delta\phi = |\phi_{hand}(f_i) - \phi_{hand}(f_{i-1})| \quad || \text{in radians} \quad (3.5)$$

$$\bar{v}_{rot}(t) = \frac{1}{n - 2z + 1} \sum_{i=z+1}^{n-z} \frac{\Delta\phi + \Delta\theta + \Delta\psi}{3 * (time(t_i) - time(t_{i-1}))} \quad (3.6)$$

$$\hat{v}_{rot}(t) = \begin{cases} 1 & \text{if } \bar{v}_{rot}(t) > v_{absMax} \\ 0 & \text{if } \bar{v}_{rot}(t) < v_{absMin} \\ \frac{\bar{v}_{rot}(t) - v_{absMin}}{v_{absMax} - v_{absMin}} & \text{otherwise} \end{cases} \quad (3.7)$$

**Unified velocity**  $v_{uni}$  defines a velocity measure which includes translation and rotation in a single value. During our research we uncovered the need of a velocity measure which combines hand movement and translation in a unified measure. The idea of unified velocity is simple. We define a point 10cm away of

<sup>16</sup><http://www.euclideanspace.com/maths/geometry/rotations/euler/index.htm>.

the hands origin in direction of the hand. This point is close to the finger-tip of the users index finger. Calculating the velocity of this point moving in 3D allows the mapping of both transitional and rotational movement on a single value.

The virtual projection of the point  $P_{uni}$  is calculated by rotating a vector  $\vec{z} = (0, 0, 10)$  to the direction of the hand and adding to the position of the hand (see equation 3.8). The velocity  $v_{uni}$  can be calculated the same ways as  $v_{trans}$  by replacing hand coordinates with the coordinates of  $P_{uni}$ . The calculation of the rotation matrix  $M_{rot}(t)$  is defined in equation 3.1.

$$P_{uni}(t) = M_{rot}(t) * \underbrace{\begin{pmatrix} 0 \\ 0 \\ 10 \end{pmatrix}}_{\vec{z}} * M_{rot}^{-1}(t) + P_{hand}(t) \quad (3.8)$$

The length of the vector  $\vec{z}$  defines how much the unified velocity is affected by the rotation. For example, if the length of  $\vec{z}$  equals the distance between hand-origin and display, then the  $v_{uni}$  equals the velocity of the pointer on the display. The mean velocity  $\hat{v}_{uni}$  is calculated in analogy to the calculation of  $\hat{v}_{trans}$  shown in equation 3.2 and 3.3. The main difference of the calculation of  $\hat{v}_{uni}$  is an adaptive threshold.

The minimal threshold is based on the rotational velocity  $\hat{v}_{rot}$ . It can vary between 0 and  $v_{absMin}$ . This dynamic lower threshold  $t_{dyn}$  has the effect that the ratio of absolute pointing is zero, only if the hand-rotation is very slow (see equation 3.9). It reduces the gap which can occur when slow hand-movements are followed by fast movements. The static threshold  $t_{rotMin}$  is set to 0.1. The dynamical threshold  $t_{dyn}$  is inserted in equation 3.10 instead of  $v_{absMin}$ . The effect of this dynamic threshold is explained together with the algorithm of *HyPoba*.

$$t_{dyn}(t) = \begin{cases} 0 & \text{if } \hat{v}_{rot}(t) < t_{rotMin} \\ v_{absMin} & \text{if } \hat{v}_{rot} > v_{absMin} \\ \hat{v}_{rot} - v_{absMin} & \text{otherwise} \end{cases} \quad (3.9)$$

$$\hat{v}_{uni}(t) = \begin{cases} 1 & \text{if } \bar{v}_{trans}(t) > v_{absMax} \\ 0 & \text{if } \bar{v}_{trans}(t) < t_{dyn}(t) \\ \frac{\bar{v}_{uni}(t) - t_{dyn}(t)}{v_{absMax} - t_{dyn}(t)} & \text{otherwise} \end{cases} \quad (3.10)$$

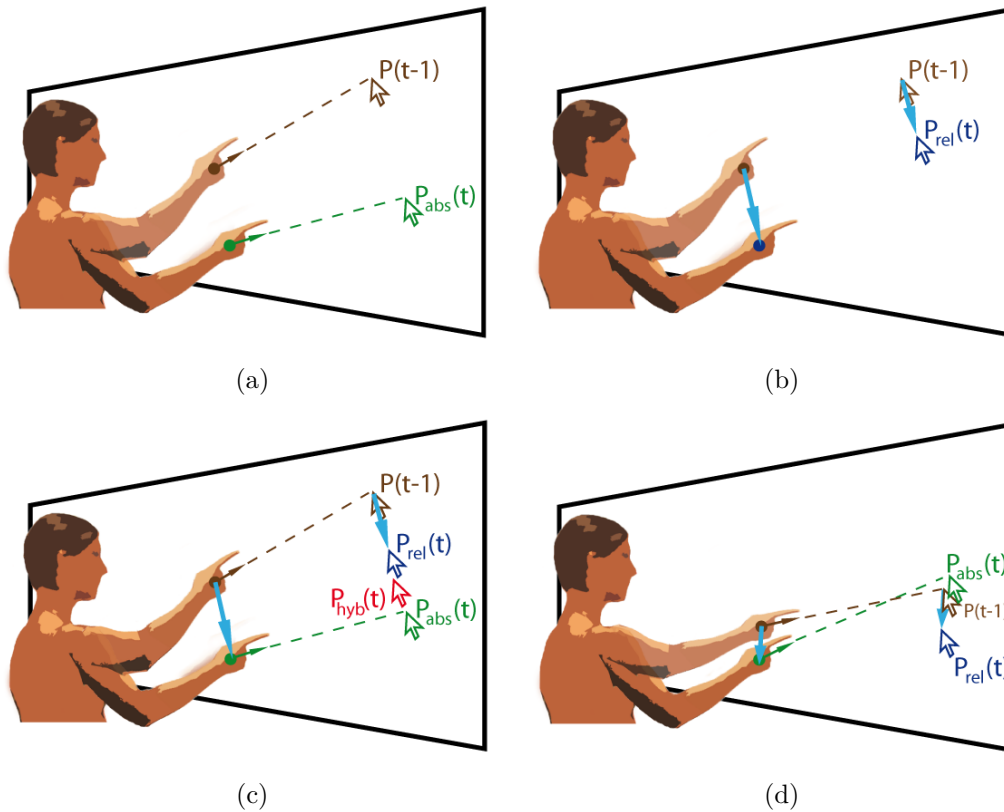


Figure 3.10.: Pointing Modes: (a) Absolute pointing (b) Relative pointing (c) *Hy-Pobapointing* (d) Acceleration problem with *HyPobapointing*.

### 3.6.2. Calculation Algorithms

The different velocity measures are required by the intersection point calculation algorithms. For the mapping function of the relative movement, the rotation of the hand is not important. Therefore  $\hat{v}_{trans}$  will be applied to define the relation between the hand- and cursor movement in relative pointing mode. Whereas  $\hat{v}_{uni}$  defines ratio between absolute and relative pointing. The following three algorithms for intersection calculation are implemented in the *Multi-Intersection* node.

#### Absolute Pointing

The absolute pointing algorithm is a ray casting algorithm which selects displays and on- or off-screen positions for each display. One pointing device can intersect with only one display. If displays are overlapping the algorithm selects the display with the smallest distance between the calculated intersection point and the center of the display (*centerdistance()*). This selection criteria works for the setting in our experiment, but as soon as mobile devices are used this algorithm should be extended. In section 5.1.3 we provide solution for extending this algorithm for more accurate display selection. Algorithm 1 calculates intersection points for each

*Room-Object* node and compares each intersection point  $P_{abs}$  with the previous ones. This algorithm is executed for each frame to determine off screen positions for the calculation of the halo's for non-focused displays. The calculation of the absolute intersection point  $P_{abs} = absIntersection()$  is calculated by three steps.

1. *Normal vector*: For each object the normal vector is calculated first. The normal vector is defined as the cross product from two vectors defined by the corner of the display  $\vec{n} = P_{BR}P_{BL} \times P_{TR}P_{BR}$ .
2. *Frontface*: In the *Room-Object* node the backface option can be selected. This option also calculates intersection points when pointing backwards at an object. By default only the front face of a display is considered. If the angle between the normal vector of the display ( $\vec{n}$ ) and the pointing direction ( $\vec{u}_{hand}$ ) equals 0 the vectors are parallel and no intersection can be found ( $\alpha = \vec{n} \cdot \vec{u}$ ). If  $\alpha < 0$  the ray intersects the object from the back side.
3. *Intersection*: We calculate the intersection point as a ray emerging from the origin of the hand  $P_{hand}$  in pointing direction  $\vec{u}_{hand}$ .

$$P_{abs} = \frac{-\vec{n} \cdot \vec{w}}{\vec{n} \cdot \vec{u}} = \frac{-ax_0 + by_0 + cz_0 + d}{\vec{n} \cdot \vec{u}} \quad (3.11)$$

In figure 3.10a the pointing behavior of the absolute pointing is illustrated. The previous pointing position  $P(t - 1)$  is marked with the brown pointer. For the algorithm it does not matter whether the last position was calculated in absolute or in relative mode. The green pointer marks the new absolute intersection point  $P_{abs}(t)$ .

---

**Algorithm 1: Absolute Pointing**


---

**Input:** RigidBody (rb), n RoomObjects (ro)

**Output:** 0-1 IntersectionPoint, 0-n OversizePoint

```

1 begin
2    $P_{abs} = null$  ;
3   foreach RoomObject as  $ro_i$  do
4      $P_{temp} = absIntersection(ro_i, rb)$  ;
5     if  $centerdistance(P_{abs}, ro_i) < centerdisctance(P_{temp}, ro_i)$  then
6        $P_{abs} = P_{temp}$  ;
7   return  $P_{abs}$ 

```

---

**Relative Pointing**

Relative Pointing is an algorithm which calculates the intersection point  $P_{rel}(t)$  based on the previous point  $P(t - 1)$  in relation to the  $\hat{v}_{trans}$  and the dimension of the display. This algorithm requires an initial point on the display and cannot



be used standalone in MDE's. In equation 3.12 the exponential function of  $\hat{v}_{trans}$  is normalized by two factors  $v_{dispMax} = e^1$  and  $v_{dispMin} = 0.2$  where  $e$  denotes the Euler's constant. The relative movement of the hand  $\Delta p$  is defined as the movement relative to the normal vector of the display ( $\vec{n}_{display}$ ). We therefore simply apply the absolute pointing algorithm where the direction of the hand is replaced by the inverted normal vector of the display (see algorithm 2 line 2).

$$v_{rel}(t) = \frac{e^{\hat{v}_{trans}(t)} - v_{dispMin}}{v_{dispMax} - v_{dispMin}} \quad (3.12)$$

---

**Algorithm 2:** Relative Pointing
 

---

**Input:** 1 RigidBody (rb), RoomObject (ro)

**Output:** 1 IntersectionPoint

```

1 begin
2    $P_{temp} = absolutepointing(\vec{n}_{display} \cdot -1)$  ;
3    $\Delta p = (P_{temp} - P(t-1)) + (v_{rel} * displaydimension)$  ;
4    $P_{rel}(t) = P(t-1) + \Delta p$ ;
5    $P(t-1) = P_{rel}$  ;
6   return  $P_{rel}$ 
    
```

---

In Figure 3.10b the relative mode is illustrated. The cyan colored arrow indicates the hand movement between time  $(t - 1)$  and time  $(t)$ . The relative algorithm requires an initial position marked by the brown pointer at position  $P(t - 1)$ . For the calculation of the new relative intersection point it does not matter by which algorithm the previous point was computed. The relative movement of the hand is mapped according to the mapping function defined in equation 3.12. The new position is marked with the blue pointer  $P_{rel}(t)$ .

### HyPoba Pointing

HyPobaPointing is an algorithm that calculates the intersection point  $P_{hyb}$  as a weighted combination of the absolute pointing algorithm (1) and the relative pointing algorithm (2). The combination requires  $P_{abs}$  which can either be on the display or in the oversize area of the display and the intersection point from the previous frame  $P(t - 1)$ . The calculation of the hybrid intersection point is illustrated in figure 3.10c. Based on the initial intersection point  $P(t - 1)$  (brown pointer) a new relative intersection point  $P_{rel}(t)$  is calculated (blue pointer). Without respect to  $P(t - 1)$  a new absolute intersection point  $P_{abs}(t)$  (green pointer) is calculated as well. Depending on the above described velocity function, a weighted combination of  $P_{abs}$  and  $P_{rel}$  is calculated which is illustrated by the red pointer  $P_{hyb}(t)$ . In the illustrated example the user is interaction at a medium speed since the relative and absolute intersection point equally affect the hybrid intersection point.

An simplified version of the algorithm is shown in equation 3.13. This initial version can be used and with a good choice of the threshold parameters it already provides a good pointing behavior. However, for certain situation this simplified algorithm computes unintended cursor positions. Pointing interaction using this formula has the following characteristics:

- + Fast and direct pointing behavior for fast hand movements.
- + Slow and accurate pointing for slow hand movements.
- Switching from slow to fast hand movements can cause the pointer to “jump” to the next position.
- Slow movement is not affected by the rotation of the hand. If the user rotates the hand slowly to the left side the cursor is still operated in relative mode. If the slow rotation is followed by a fast upwards movement the cursor performs a transition to the absolute mode where the user already previously pointed to the left. This results in an unintended movement to the left side.
- After a fast movement, the user is likely to overshoot the target and hast to move the cursor back again.

$$P_{hyp}(t) = \begin{cases} P_{abs} & \text{if } \hat{v}_{uni}(t) > v_{uniMax} \\ P_{rel} & \text{if } \hat{v}_{uni}(t) < v_{uniMin} \\ P_{abs} \cdot \hat{v}_{uni}(t) + P_{rel} \cdot (1 - \hat{v}_{uni}(t)) & \text{otherwise} \end{cases} \quad (3.13)$$

In order to reduce the negative factors of *HyPobapointing* we already introduced the dynamic lower threshold  $t_{dyn}(t)$  which replaces  $v_{uniMin}$ . The effect of the dynamic threshold is to avoid large gaps between absolute and relative pointing positions. If  $\hat{v}_{rot}(t)$  is below  $v_{absMin}$  (e.g: if the user performs a click-gesture) the absolute ratio is 0. Thanks to the exponential function for relative pointing, the cursor does not move. This allows exact cursor placement which is, depending on the threshold values, not affected by natural hand tremor. For continuous slow rotation of the hand, the dynamic threshold applies a small ratio of absolute pointing, even though  $\hat{v}_{trans}(t)$  is below  $v_{uniMin}$ . This has the effect that the cursor is very slowly moving in direction of  $P_{abs}(t)$ . This trailing effect has two advantages. First, it reduces the gap of unintended hand rotation in slow movements. And second, it can be used for exact positioning of the cursor as well.

In the final algorithm (3) for *HyPobapointing* additional functions are applied to increase pointing accuracy and naturalness of pointing.

Overshooting for fast movements is prevented by applying a prefetching algorithm analog to *Adaptive Pointing*. This algorithm estimates a possible position of the

**Algorithm 3:** Hybrid Pointing**Input:** 1 RigidBody (rb), n RoomObjects (ro)**Output:** 0-1 IntersectionPoint, 0-n OversizePoint

---

```

1 begin
2    $P_{abs} = absolutepointing(rb, ro)$  ;
3   if  $P_{prev}$  then
4     if  $P_{abs}$  then
5        $get(P_{rel})$  ;
6       if  $\hat{v}_{uni} > t_{prefetch}$  then
7          $P_{hyp} = P_{abs} + historyDirection(4) \cdot -(\hat{v}_{uni})$  ;
8       else
9         if  $acceleration > 0$  then
10           $P_{hyp} = ignoreCrossover(P_{rel}, P_{abs})$  ;
11        else
12           $P_{hyp} = P_{abs} \cdot \hat{v}_{uni}(t) + P_{rel} \cdot (1 - \hat{v}_{uni}(t))$  ;
13        return  $P_{hyp}$ 
14      else
15        return null
16    else
17      return  $P_{abs}$ 

```

---

cursor after a few frames. This has the effect that the user reduces the interaction speed earlier since the prefetching cursor approaches the target prior to the real cursor position. When slowing down the interaction, the reduction of prefetching has a strong deceleration effect which reduces the risk of overshooting. We therefore calculate an outlier-resistant history of intersection points by truncating the last  $n$  intersection points. Based on the truncated history, a direction vector is calculated indicating the estimated direction of the next move (Listing 3.3). If  $\hat{v}_{uni}$  exceeds the threshold  $t_{prefetch}$  (see line 6 in algorithm 3) the mean direction vector is scaled by  $\hat{v}_{uni}$  and added to the absolute position (in line 7 of algorithm 3).

The crossover prevention (listing 3.4) reduces unnatural cursor movement when switching from relative to absolute mode. This algorithm prevents effects as illustrated in figure 3.10d. At the time  $(t - 1)$  the user is pointing at the position marked with the brown pointer  $P(t - 1)$ . If the user is moving his hand slowly down, he is operating in the relative mode which results in the blue pointer defining  $P_{rel}(t)$ . We observed that the hand is rotating during slow hand movements, especially when using both hands. If operating in the absolute mode, this rotation would result in the green pointer which defines  $P_{abs}(t)$ . If the user is pointing quickly downwards in the following frame  $t + 1$  the cursor which is positioned at the blue position  $P_{rel}(t)$  would be heavily affected by the above positioned  $P_{abs}(t)$ . The down-moving absolute intersection point would attract the hybrid intersection point and there-

Listing 3.3: Mean direction history

```

1 public Vector2d historyDirection(int size)
2 {
3     Vector2d predictDirection = new Vector2d();
4     Point2d pPrevious = null;
5     Iterator<Point2d> itr = this.iterator();
6     while(itr.hasNext())
7     {
8         Point2d pTemp = itr.next().clone();
9
10        if (size-- == 0)
11            break;
12        if (pPrevious == null)
13        {
14            pPrevious = pTemp;
15        }
16        else
17        {
18            pPrevious.x -= pTemp.x;
19            pPrevious.y -= pTemp.y;
20            predictDirection.add(pPrevious);
21            pPrevious = pTemp;
22        }
23    }
24    return predictDirection;
25 }

```

fore result in an upward movement of the cursor until the gap between  $P_{abs}(t)$  and  $P_{rel}(t)$  is closed. This unintended upwards moving of the cursor is eliminated by the algorithm listed in listing 3.4.

After several refinement of the algorithms, the pointing behavior of hybrid pointing became accurate and intuitive. In table 3.9 the optimal values for each parameter are defined.

### 3.7. Summary

In this chapter, we introduced a prototype for bimanual cross-display interaction. The prototype consists of three individual parts. The visual tracking which has been optimized using the redesign of “Whitey” and the re-installation of the Optitrack tracking system. *HyPobaa* novel algorithm for hybrid pointing has been implemented as a data filter in Squidy. In combination with other filters introduced in section 3.4, gestures and display intersection can be determined from the 6D- and 3D-Data provided by the TrackingTools software. Multidragger, the last component in the chain provides the necessary functions to drag objects over display borders. The included statistic tool automatically collects data required for the experiment in the following chapter. In the requirement analysis we defined five tasks for collaborative work in MDEs. The three of the tasks can be successfully completed using *MultiDragger* are marked with “+” in the following list.

Listing 3.4: Crossover prevention

```

1 public Point2d ignoreCrossover(Point2d pRelative, Point2d pLast, double velocity)
2 {
3     Point2d p2d = new Point2d(0,0);
4     Vector2d movement = dirQueue.historyDirection(4);
5     if (pRelative.x < pLast.x && movement.x > 0)
6         p2d.x = this.iPoint2d.x;
7     else if (pRelative.x > pLast.x && movement.x < 0)
8         p2d.x = pLast.x;
9     else
10        p2d.x = (pLast.x * (1-velocity) + pRelative.x * velocity);
11
12    if (pRelative.y < pLast.y && movement.y > 0)
13        p2d.y = pLast.y;
14    else if (pRelative.y > pLast.y && movement.y < 0)
15        p2d.y = pLast.y;
16    else
17        p2d.y = (pLast.y * (1-velocity) + pRelative.y * velocity);
18    return p2d;
19 }

```

- + T1 *Move content from any private device to a public display:* *MultiDragger* supports this task for devices of all sizes running on Windows7 or Windows Vista. However, distant pointing on a smart-phone can be very difficult without additional tools. Several concepts to extend *MultiDragger* are introduced in chapter 5.
- + T2 *Manipulate any content on any display:* Bimanual gestures allows the user to manipulate objects on any device on which *MultiDragger* is running.
- T3 *Access data and software on a private device:* Through the simulation of multi-touch events, also native applications could be controlled using gesture interaction. However, due to the complications using the multi-touch vista driver this functionality had to be rejected. The implemented bridge plug-in operates only on software level (*MultiDragger*) and not on operating system level.
- + T4 *Deleting objects from any display:* *MultiDragger* allows to define a view port for any device. If the position of a garbage bin is defined as a *Room-Object* node in Squidy, objects can be dragged onto the bin. The objects are kept in the invisible view port of the bin and can be restored. In chapter 5 we introduce a concept which simply allows restoring objects from the bin using drag&drop interaction.
- T5 *Printing any object from any display:* This function has not been tested yet. But basically the printer can be defined as an *Room-Object* node in Squidy and therefore be accessed through the gesture interaction. The only thing that has to be done is to send a print command when an object is dragged into the invisible view port of the printer.

Even though not all tasks can be performed by now. The main goal of providing bimanual cross-display interaction with the capability of applying multi-touch events to objects has been achieved. In the following chapter we describe the experiment using the tools and algorithms described in this chapter.

## 4. Experiment

This chapter presents the experiment which was carried out to examine the task-related difference between the proposed pointing techniques. Two different tasks, a single-handed drag & drop task and a bimanual manipulation task have been tested with each pointing technique. The 14 participants produced 846 valid test samples.

### 4.1. Goal of the Evaluation

In related work, various evaluation methods for evaluation pointing interaction can be found. This experiment compares two pointing techniques for two different tasks.

1. **Cross-display object movement:** Considering the previously described scenario; a user wants to move an object from one display to another. We assume, that for this task the interaction speed is the main factor.
2. **Bimanual object manipulation:** Bimanual interaction forces the user to control two cursors simultaneously. The user's attention is thus focused on two cursors rather than just one. This is likely to reduce the accuracy of the hand movements. Therefore accuracy is a very important factor for this task.

The main goal of this evaluation is to see whether the preferences are task-dependent or not. A second goal is to examine the characteristics of the introduced *HyPoba Pointing*. Therefore we made the following hypotheses.

<b><i>Hypothesis H0</i></b>	There is no task-dependent difference in accuracy or speed between <i>HyPoba Pointing</i> and <i>Absolute Pointing</i> .
<b><i>Hypothesis H1</i></b>	<i>Absolute Pointing</i> allows faster cross-display object movement than <i>HyPoba Pointing</i>
<b><i>Hypothesis H2</i></b>	<i>HyPoba Pointing</i> results in higher accuracy for cross-display object movement than <i>Absolute Pointing</i>
<b><i>Hypothesis H3</i></b>	<i>HyPoba Pointing</i> allows faster bimanual object manipulation than <i>Absolute Pointing</i>
<b><i>Hypothesis H4</i></b>	<i>HyPoba Pointing</i> results in higher accuracy for bimanual object manipulation than <i>Absolute Pointing</i>

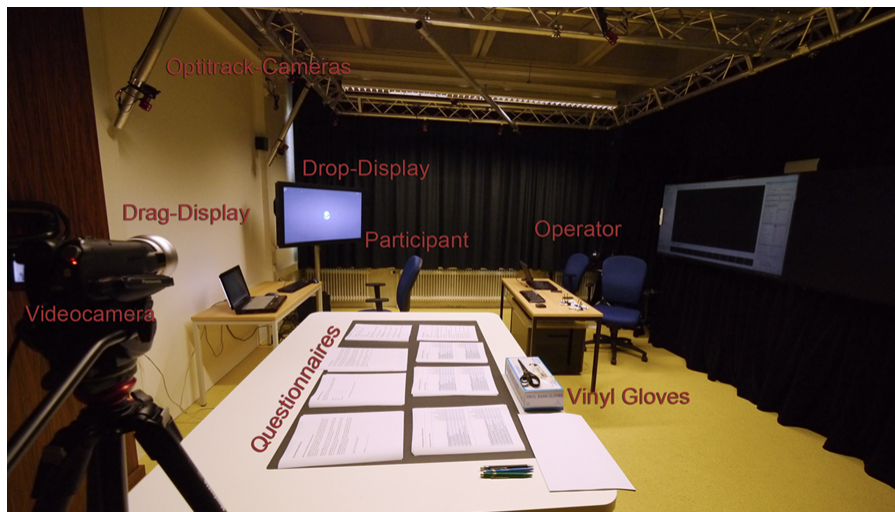


Figure 4.1.: Evaluation Setting

## 4.2. Evaluation Setting

The experiment used the Optitrack motion tracking (see chapter 3) system with retro-reflective markers attached to cotton gloves worn by the participants, a notebook display (17") and a 50" overview display. The participants were seated in a central position 1.8m away from each display (see figure 4.1). After a pre-test we decided to do the evaluation seated to prevent overfatigue since the experiment lasts for about 90 minutes. For hygenical reasons the participants had to wear vinyl exam gloves underneath the cotton glove. During the experiment all participants have been filmed by a camcorder.

### 4.2.1. Participants

Advertisements about the experiment were published on bulletin-boards all over the University of Konstanz. From all interested persons 6 women and 8 men were invited to the experiment. All participants were students or phd's with varying backgrounds ranging in age from 19 to 55. Only two male students have an information-science background. For one male participant the gesture recognition did not work at all. Due to limited time for the experiment this participant could not be replaced and is therefore not taken into account in the analysis. All participants were right handed and none of them has ever used similar interaction techniques on MDEs. Most of them are frequent computer users and familiar with simple multi-touch gestures like scaling and rotating objects (see 4.1).



Table 4.1.: Computer background of all participants.

Participants being familiar ...	Yes	No
... with (non-digital) laserpointer	11	2
... with MDE's	2	11
... multi-touch smartphones	9	4
... multi-touch displays	4	9
... multi-touch scaling gestures	12	1

### 4.2.2. Tasks

Since *HyPoba Pointing* is based on *Absolute Pointing* and *Relative Pointing* the goal of the initial design was to compare all three pointing techniques. Tabbing tests or other common evaluation methods for evaluating pointing techniques do not support bimanual pointing. An adaption of a tabbing test for bimanual pointing could be created using two targets at the same time. If only the precision of the technique is important, such a test could be applied. However, dragging and manipulating objects does not rely only on pointing precision. Other aspects such as the corresponding gesture and the synchronous hand coordination while manipulating objects is also important.

We therefore decided to split up one of the main tasks as described in section 2.1 into two individual tasks. We explicitly define the main task as: “Move an object from the private device onto the overview display and scale it so everyone can see it”. This task can be split up into a single handed drag & drop task to move the object from one display to another, and a bimanual manipulation task in which the user places and scales the object to the desired position, size and rotation. Therefore we designed two different test to consider both types of interaction.

#### Drag & Drop Task

The drag & drop task was a single-handed task to test the capability of cross-display object movement. Figure 4.2 shows one participant performing the drag & drop task with a large object. The participant has to move the source-object to a congruent position with target-object. We designed our test in analogy of common tabbing tests [König, 2010; ISO-9241-16:1999, 1999]. We therefore defined a homing position where each sample task is started by performing a click-gesture. The homing position is always located on the center large display. After clicking on the homing button, the source-object is displayed on the center of the notebook display. As the position remains the same for all objects, we have a standard measure for the first grab of the source-object (SO). We also defined three different object sizes ( $SO_{Large} = 90 * 60$ ,  $SO_{Medium} = 250 * 166$ ,  $SO_{Small} = 500 * 322$ ). Figure 4.2b

shows one of the users grabbing a large source-object from the notebook. When the object is successfully grabbed, a dataset is recorded. This dataset contains data about time, hand-movement and more (see table 3.10) from the homing position until the first grab. The user drags the object onto the large display (Figure 4.2c). When the source-object is close to the target-object, a predefined accuracy level is reached. Two different accuracy levels are defined. Level 1 is indicated with the orange icon, so the user see that the task is almost completed. When reaching level 2 the green icon (Figure 4.2d) indicates that the object can be released. The data are recorded for each level. The values for level 2 are defined in table 4.3. Level 1 is defined as 20% less accurate level 2.

Table 4.2.: Different Datasets for the analysis of the experiment

Level	Description
<b>First grab</b>	Contains data recorded between clicking on the homing position and successfully grabbing the source-object
<b>Level 1</b>	Three criteria have to match to reach this Level. The manhattan distance between two objects is less than 15 pixels (Transition accuracy), the difference between the two angles is less then 10 degrees (Rotation accuracy) and the the difference between the sums of width and height is less than 25 (Size accuracy).
<b>Level 2</b>	The same criteria as for level 1 only the values require more accurate placement. The transition accuracy is reduced to 8 pixels, the rotation accuracy must be below 6 pixels and the maximum difference of the size is set to 12 pixels.
<b>Successful</b>	Contains data of the entire task until the user drops the source-object with an accuracy of level 2.
<b>Timeout</b>	If the maximum time for the task is exceeded, a red icon indicates the user to abort the interaction. The dataset contains all data from of the interaction including the accuracy values at the end of the task.

### Manipulation task

For the bimanual task, the well known scalling gesture for multi-touch devices has been transfered into distant freehand pointing. The task is ideal for the evaluation of bimanual freehand pointing, since the operation itself is most probably known by the participants. Only the execution by freehand pointing is new. This fact should allow the user to focus on the interaction since the task is simple to complete. An example of the bimanual manipulation task is shown in figure 4.3. The task starts by clicking on the homing button (Figure 4.3a). Source- and target-object are display on the same display. After clicking the homing button, the source-object is displayed right of the homing button. The target-objects are display on the left of the homing

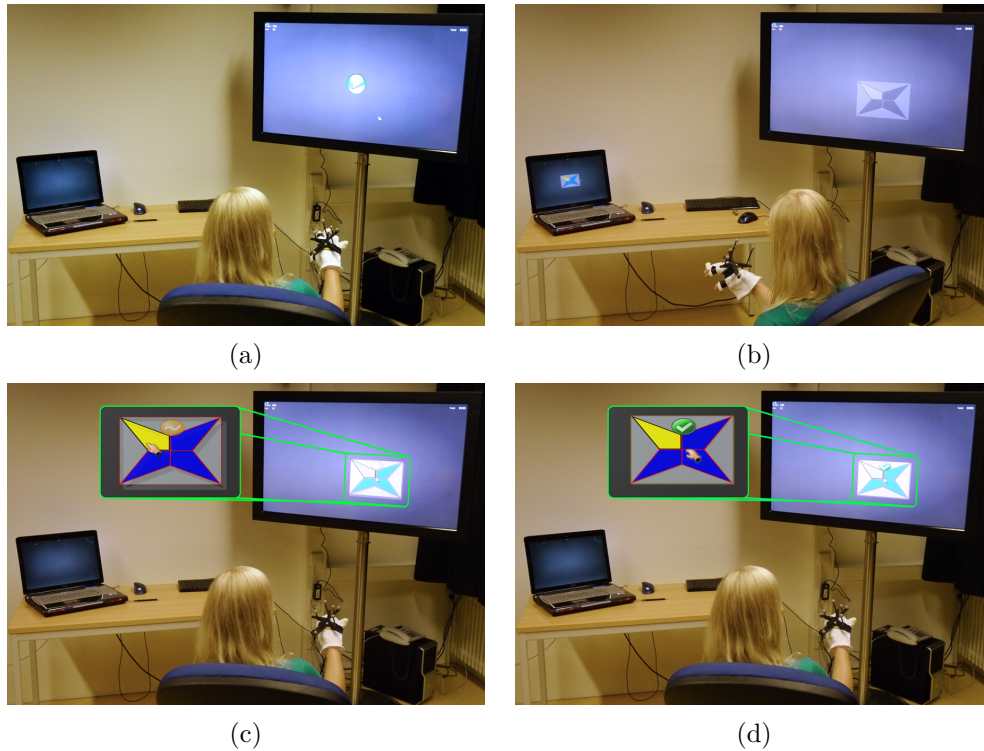


Figure 4.2.: Drag & Drop: (a) The user initializes the task by clicking on the homing position (green button). (b) The objects appear on both displays. (c) The orange icon in the dragged object indicates the accuracy is almost good enough. (d) The green icon indicates a good accuracy and the user can release the object.

position in varying positions, sizes, and orientations. Like in the drag & drop task, the user had to bring both objects to a congruent position. Small objects had to be enlarged, large objects minimized. The medium objects did not vary in size but in rotation.

In addition the aspect of different screen sizes in MDE's was also taken into account. The iteration cycle as described in figure 2.2 requires object movements from private devices to large public devices and back. In addition arrangements and object manipulation on large displays have to be supported. This leads to three additional conditions for the evaluation

- **Small** → **Large**: Dragging an object from a small notebook device onto a large public display.
- **Large** → **Small**: Dragging an object from a large public display to a small private notebook.
- **Large** → **Large**: Dragging an object between two equal sized displays. For privacy reason dragging an object from a small private device to another small

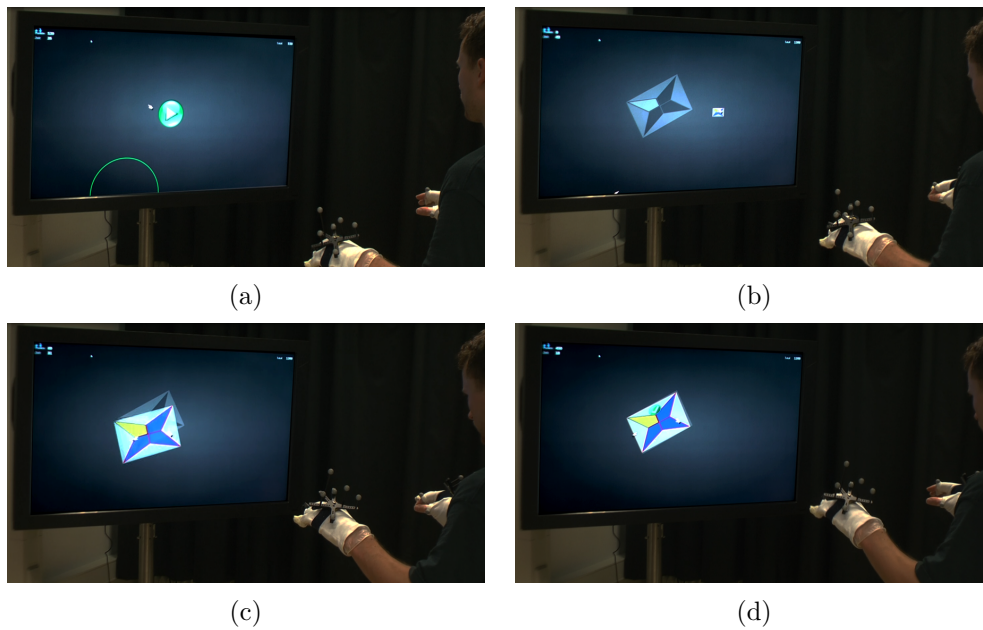


Figure 4.3.: Manipulation: (a) Like with the drag & drop the participant starts from a central position of the display. (b) The participant needs to adjust the colored object to the gray scale object. (c) The participant needs both hands to scale and rotate the object. (d) Finally the objects are aligned accurate enough.

private device is not considered, since only the owner should be able to access his private device.

Considering all those aspects requires an evaluation design with 18 different conditions ( $3_{techniques} * 3_{screensizes} * 2_{tasks}$ ). The number of conditions considering all aspects requires too many participants and too much time for each participant. Therefore we had to reduce the amount of conditions. A pre-test should help to find out the best parameters to ensure a good amount of valid samples. These parameter include the required time for each sample, the specification of a good accuracy level and the specification of parameters for the pointing algorithms.

### 4.2.3. Pre-Test

The goal of the pre-test was to adjust various parameters and to specify a good value for the accuracy measurement. We asked 4 students from our working-group to participate on an informal pre-test. We tested different parameter settings. We optimized time and accuracy parameters in such way the majority of users can complete the tasks by remaining on a high accuracy level. We therefore defined the parameters as specified in table 4.3. In addition the properties for the Hy-

*Poba Pointing* algorithm have also been adjusted for each user. The average of the user preferences were set as default parameters for the experiment.

Table 4.3.: Accuracy and time parameters retrieved from the pre-test.

Parameter	Value
Transition Accuracy	8 pixels
Rotation Accuracy	6 degrees
Size Accuracy	12 pixels
Time Drag & Drop	25 seconds
Time Scaling Task	50 seconds

The pre-tests showed that a manipulation task requires between 20 and 60 seconds in average. To avoid participants from endlessly trying, we decided to set a maximum time of 50 seconds for each sample for the manipulation task, 25 seconds for the drag & drop task. This allows us to calculate the maximum time required. The maximum interaction time for each participant should not exceed 45 minutes. We decided not to reduce the maximum time, since it could lead to wrong conclusions and to frustration of the user. The number of samples when testing all three techniques is too small (see equation 4.1). Testing only two techniques provides 50% more samples (equation 4.2). In order to get more samples per technique we decided to test only two techniques per participant (*HyPoba Pointing* and *Absolute Pointing*). In analogy to other experiments we compared our new technique and *Absolute Pointing* [Vogel and Balakrishnan, 2005; König et al., 2009]. This decision led to the schedule in table 4.4. It shows the planned schedule for a single participant with a maximum required time of 93 minutes. Assuming that not all participants require the maximum time for the interaction, each user should be able to conclude the test in less than 90 minutes.

$$45' = 2700'' \Rightarrow \begin{cases} 3_{techniques} * 12_{samples} * 20'' \\ 3_{techniques} * 12_{samples} * 50'' \end{cases} \quad (4.1)$$

$$45' = 2700'' \Rightarrow \begin{cases} 2_{techniques} * 18_{samples} * 25'' \\ 2_{techniques} * 18_{samples} * 50'' \end{cases} \quad (4.2)$$

#### 4.2.4. Experimental Design

We used a counter-balanced within-subject design for 2 tasks with 2 conditions each. The pretest has shown that the preference for a pointing technique is very

Table 4.4.: Evaluation schedule for each participant

Task	Estimated time
Welcome and Introduction	5 min
Pre-Test Questionnaire	2 min
Calibration and Tryout	10 min
Experiment Drag Method A	7 min
Questionnaire Drag A	3 min
Experiment Drag Method B	7 min
Questionnaire Drag B	3 min
Calibration and Tryout	10 min
Experiment Manipulation Method A	15 min
Questionnaire Manipulation A	3 min
Experiment Manipulation Method B	15 min
Questionnaire Manipulation B	3 min
Post-Test Questionnaire & Discussion	10 min
Total Time	93 min

user dependent. A within-subject design allows us to test each participant with both techniques. In order to keep the task in the original order, the drag & drop task is always performed before the manipulation task. We fully counter-balanced the pointing-techniques. One half of the participants started with *HyPoba Pointing* and the other group started with *Absolute Pointing*. Before the experiment the users were shortly introduced to the system and the tasks (Appendix A.1). A *Pre-Test Questionnaire* was used to collect the demographical data and information about individual computer knowledge (Table 4.1). In order to provide good results the calibration of gestures and testing was important. For the drag & drop task we calibrated only the required hand. We allowed a maximum of 10 trials for the task. After completing one technique, a short questionnaire (Appendix A.4) had to be filled out. This time (approx. 3 minutes) was used to relax before the following task. After completing both techniques for the drag & drop task, the second hand was calibrated. Another 5 trials of the manipulation task helped the user to understand how bimanual interaction works. The manipulation task was executed in the same order like the drag & drop task. Participants which began with *HyPoba Pointing* for drag & drop also started with *HyPoba Pointing* for the manipulation task. Also for the manipulation tasks individual questionnaires for each pointing technique had to be completed. A final post-text questionnaire comparing both pointing techniques with an informal discussion closed the experiment. The schedule for the entire experiment is illustrated in table 4.4. For each pointing technique and each task  $3 * 6 = 18$  samples had to be completed, six samples for each object size (small, medium large).



We prepared three different questionnaires to collect the required data.

**Pre-Test Questionnaire** : The pre-test questionnaire was used to collect demographical data and information about computer knowledge of each user (Appendix A.3).

**In-Test Questionnaire** : We created four different in-test questionnaires for each task and pointing technique (Appendix A.4 and A.5). The questions on each questionnaire were almost identical and differ only in task-dependent questions between drag & drop task and manipulation task. Each questionnaire contained 20 questions which could be categorized into 10 different categories. We initially planned to ask five questions about each category, but this would have been too time-consuming and boring for the participant. We asked questions regarding the *ease of use, cognitive load, fun, precision, physical load, object size, learning curve, task adequacy, system accuracy* and *intuitiveness*.

**Post-Test Questionnaire** : In the post-test questionnaire we asked several questions to distinguish between the pointing techniques (Appendix A.6).

## 4.3. Results

We selected 14 participants out of the 17 responding on the announcements. Thirteen of the 14 participants produced 846 valid test samples. For one male participant, the gesture recognition was not working at all. After 30min of trying we had to cancel the experiment. We found that during lunch-break some people took photographs which probably somehow affected the cameras. After recalibration the tracking system, we could keep the schedule with the following participant. Due to restricted time, we could not replace the missing male user. The following results are all based on the 13 remaining participants.

### 4.3.1. Overall Analysis

To get an overall impression of the results we first present the results from the post-test questionnaires. The analysis of this questionnaire allows the comparison between *Absolute Pointing* and *HyPoba Pointing*. To increase the readability of the diagrams, we replace *Absolute Pointing* by “Absolute” or “Abs” and *HyPoba Pointing* by “Hybrid” or “Hyb”.

*Absolute Pointing* has a better ranking according to *pointing accuracy, interaction speed, fun* and *intuition*. Whereas more users preferred *HyPoba Pointing* for *manipulation, drag & drop, interacting with small objects* and would rather use *Hy-*

*Poba Pointing* in such an environment. Figure 4.4 visualizes the the number of votes for each technique. The results for *intuition* and *manipulation* show a strong preference for either one technique. For the other aspects the results are more equal. Most users chose *Absolute Pointing* as the more intuitive pointing technique (abs = 9, hyb = 3). Even though *Absolute Pointing* seems to be more intuitive, more accurate, more fun and faster, *HyPoba Pointing* is preferred for drag & drop (abs = 4, hyb = 6) and manipulation (abs = 4, hyb = 8). This leads to the question; why would people choose the slower less accurate pointing technique as their favorite? We found two possible answers:

- The preferences are very user dependent and cannot be compared between individual participants. This could be an indication for a bad choice threshold parameters for the hybrid algorithm.
- The participants completed the post-questionnaire at the end of the experiment and possibly had problems remembering the techniques. However we recommended to compare the impression of the last technique used with the one used before, which should minimize the chance of confusion.

To over come the possible problem of confusing the two techniques we also used in-test questionnaires. Those questionnaires have always been filled out directly after the completion of each task. As for the post-questionnaire, the user could solely decide between *Absolute Pointing*, *HyPoba Pointing* or “no difference (equal)” the in-test questionnaire required a ranking of the given statements.

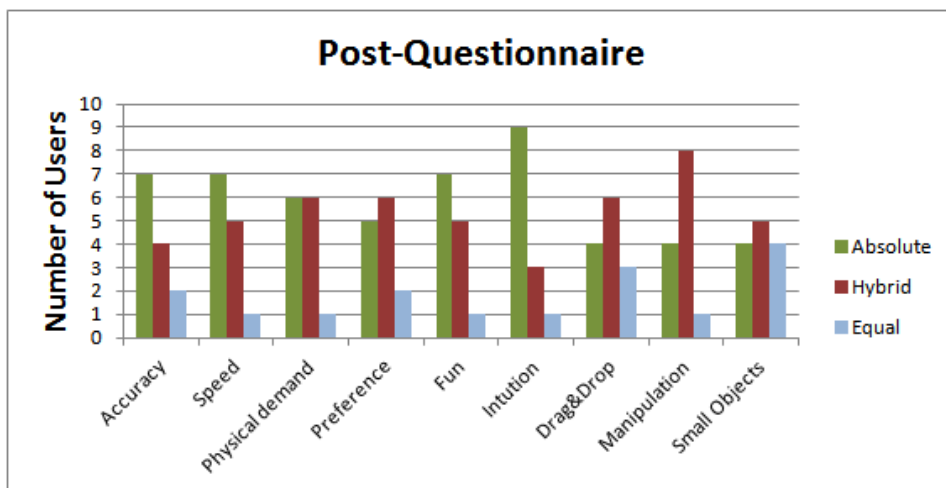


Figure 4.4.: Preferences of the pointing techniques based on the post-test questionnaire.

In the in-test questionnaires for the drag & drop task the user rated *Absolute Pointing* easier to use and less physically and mentally demanding (Figure 4.5). Fun,



precision, good for small targets and progressive learning were rated equally. *HyPoba Pointing* was found more suitable for the current task, more intuitive and with a better system-accuracy.

The in-test questionnaire for the manipulation task contained the same questions like for the drag & drop task. Only the task dependent questions varied slightly. No difference between the two techniques could be found regarding the ease of use, physical demand, task suitability, intuitiveness and the preference for small targets. *Absolute Pointing* was rated higher for precision and system accuracy, whereas more users rated *HyPoba Pointing* as less mentally demanding, more fun and a better learning curve.

Probably the most obvious result from the in-test questionnaires is the difference in mental and physical demand between the two tasks. The manipulation task turned out to be difficult and required high concentration. This results in a high mental and physical demand. The drag & drop task is rated as less demanding especially when using *Absolute Pointing*. According to the accuracy, we can note a light preference of *Absolute Pointing* for the drag and *HyPoba Pointing* for the manipulation task. Generally those results show large variances and only a larger number of participants could lead to significant results.

In addition to the questionnaire *MultiDragger* recorded a large amount of data for more detailed analysis. We counted a total of 846 valid samples. The division of 2 interaction techniques and 2 different tasks provides more than 200 samples for each condition (Table 4.5).

Table 4.5.: 846 Valid samples of all participants during the experiment.

Technique	Task	Valid samples
Absolute	Drag	219
Hybrid	Drag	206
Absolute	Manipulation	207
Hybrid	Manipulation	214
Total		846

Based on the valid samples we first calculated success rate for each condition. Figure 4.6a shows that both techniques provide similar rates. For the drag task *HyPoba Pointing* performs slightly better than *Absolute Pointing* (abs = 90%, hyb = 92%). For the manipulation task, its exactly the other way round (abs = 68%, hyb = 65%). The bimanual manipulation tasks achieved much lower success rate. This was expected, since the bimanual task is more demanding. For the following analysis we define three main measures.

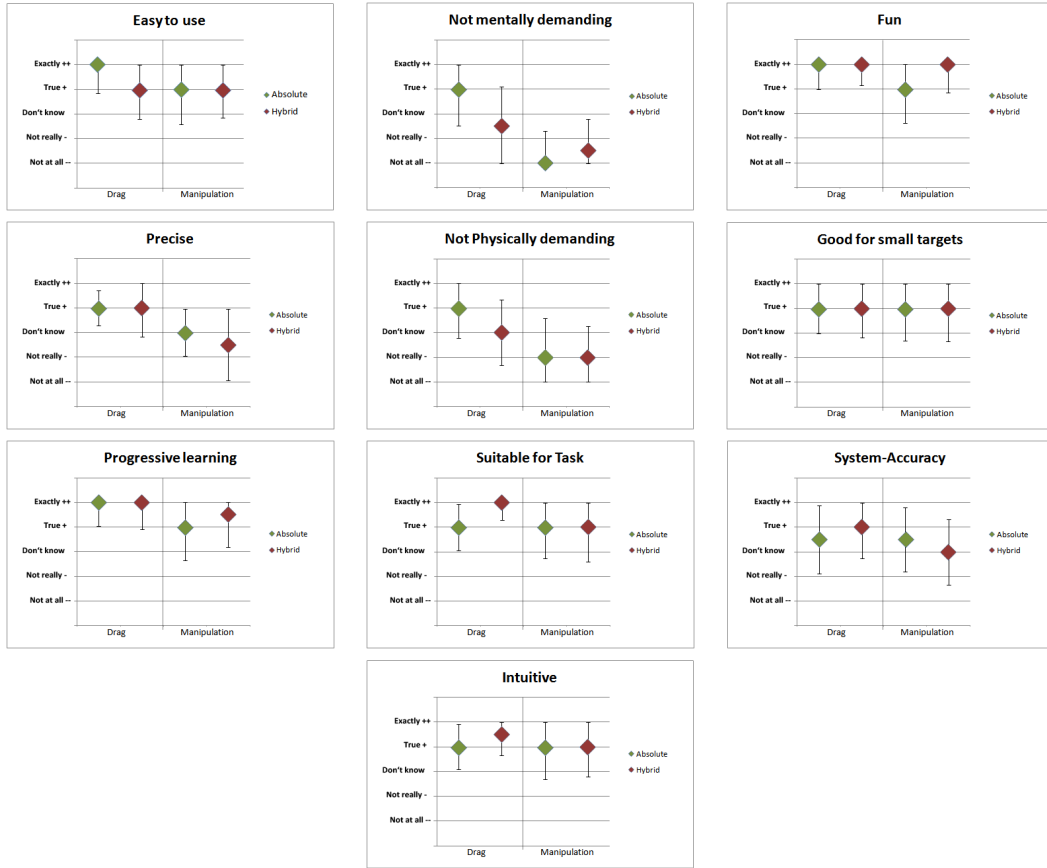


Figure 4.5.: Results of the in-test questionnaire

**Task Completion Time:** The task completion time is the number of milliseconds between clicking on the homing position and successfully placing the source-object on the designated position.

**Transition Accuracy:** The transition accuracy is the Manhattan-distance between the position of the target-object and the position of the source-object.

**Level Changes:** We previously defined the accuracy levels. This measure counts how often a level is reached. For example if a user reached the accuracy level 2 and moves out again then 2 level changes will be counted. As he moves in again, another 2 level changes will be added. Only if the user moves very quickly from level 0 to level 2 then only 1 level change is counted. We handle this value as an information about the accuracy. The smaller the number of level changes the less correction was necessary to achieve the goal.

We generally use boxplots or bar charts to visualize the results. The percentages provided inside the diagram shows the success rate of the individual boxes. In addition we used an independent-sample *t test* with a 95% confidence level. Results showing a two-tailed *p-values* below 5% ( $p < 0.05$ ) are set to be significant.

## BIMANUAL POINTING TECHNIQUES FOR CROSS-DISPLAY INTERACTION

### 4.3. Results

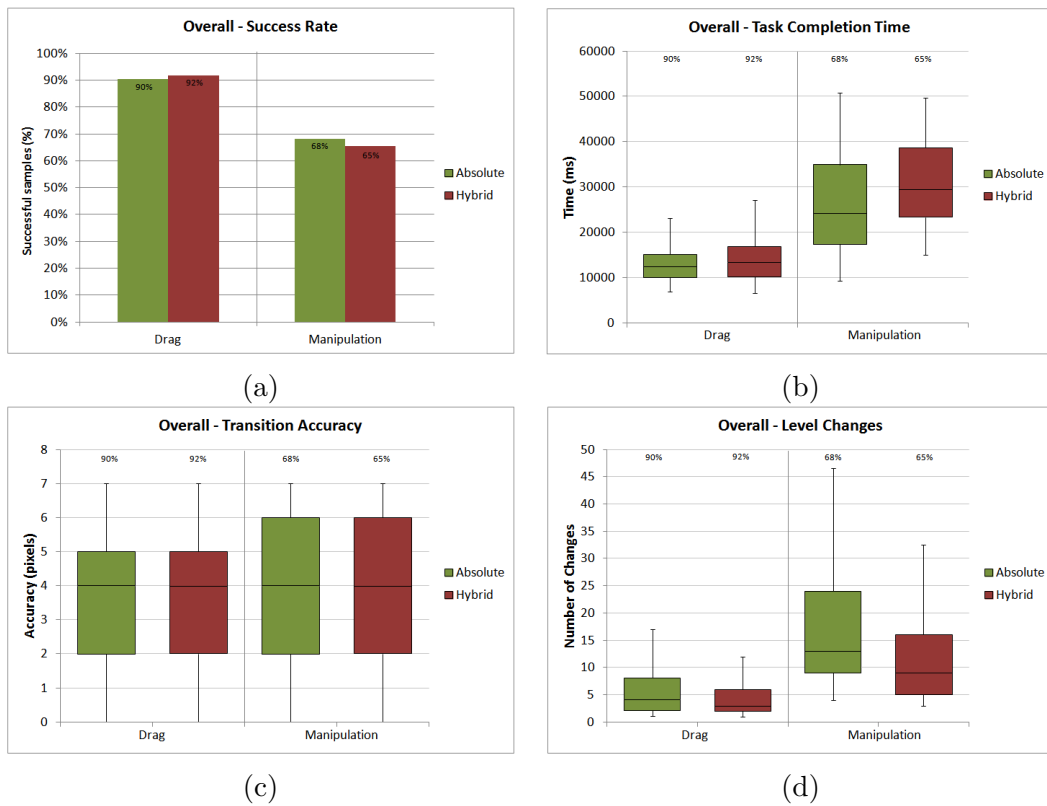


Figure 4.6.: Overall Results: (a) The success rates for each task is very similar for both techniques. (b) The analysis of the task completion time shows a significant difference between the two task and minor differences between the two techniques. (c) The transition accuracy is almost equal for all tasks. (d) The number of level changes shows significant better results for *HyPoba Pointing* in both tasks.

The comparison of the overall task completion time also shows similar effects as the success rate (Compare figure 4.6a and 4.6b). The easier drag tasks have been completed much faster than the manipulation task. As mentioned above, this effect is not part of the analysis as this effect is based on the largely varying difficulty of the two tasks. That's why we do not compare the two tasks, we only compare the pointing techniques for each task individually. This indicates that participants which reached the accuracy level 2 did not overshoot the target as often as participants using *Absolute Pointing*.

Comparing the task completion time for the drag task does not provide significant differences between *Absolute Pointing* and *HyPoba Pointing*. Figure 4.6b and table 4.6 show minor differences between the two techniques. The transition accuracy shows almost equal results. However the number of level changes favor the *HyPoba Pointing* significantly ( $p < 0.001$ ).

The results for the manipulation task indicate a higher deviation between the individual samples. Regarding the task completion time, the mean value for *Absolute Pointing* is lower than the mean value for *HyPoba Pointing*. The T-statistic show a significant better result for the absolute technique ( $p = 0.001$ ). Surprisingly the transition accuracy behaves almost identically as we already noticed for the drag & drop task (Figure 4.6c). Regarding the number of level changes, the *HyPoba Pointing* shows a significant better result ( $p < 0.001$ ).

Summarizing the overall analysis we have found a significant smaller number of level changes of *HyPoba Pointing* for both tasks. This results in a higher accuracy since less correction movements were required and therefore proves the hypotheses H2 and H4. *Absolute Pointing* provides shorter task completion time for the drag & drop task, however the difference between both techniques is not significant. H1 could therefore not be proved. The significant faster task completion time of *Absolute Pointing* for the manipulation rejects H3.

Table 4.6.: T-statistic values showing the significance between the two techniques for both tasks.

	Drag	Manipulation
Task completion time	$t(382) = 1.403, p = 0.161$	$t(275) = 3.482, p = \mathbf{0.001}$
Transition Accuracy	$t(385) = 0.205, p = 0.838$	$t(277) = 0.003, p = 0.997$
Level Changes	$t(339) = 4.860, p < \mathbf{0.001}$	$t(228) = 4.707, p < \mathbf{0.001}$

### 4.3.2. Per User Analysis

From the results of the in- and post-test questionnaires we found divergent results regarding the two pointing techniques. Also the observation of the users during the experiment has shown, that some users felt more comfortable using one particular technique. Therefore we performed an analysis for each user.

#### Drag & drop task

The overall analysis showed no significant difference between the two techniques according the task completion time and transition accuracy. Figure 4.7 shows varying results for each user. Even though the means of *HyPoba Pointing* and *Absolute Pointing* are very close, we see different patterns regarding individual users.

In figure 4.8 we chose 8 participants showing interesting result patterns in combination with the overall visualization in the middle. The diagrams on the left side show users which perform better using *Absolute Pointing*, while the diagrams on the right side show users which favor the *HyPoba Pointing*. For example user 13 has

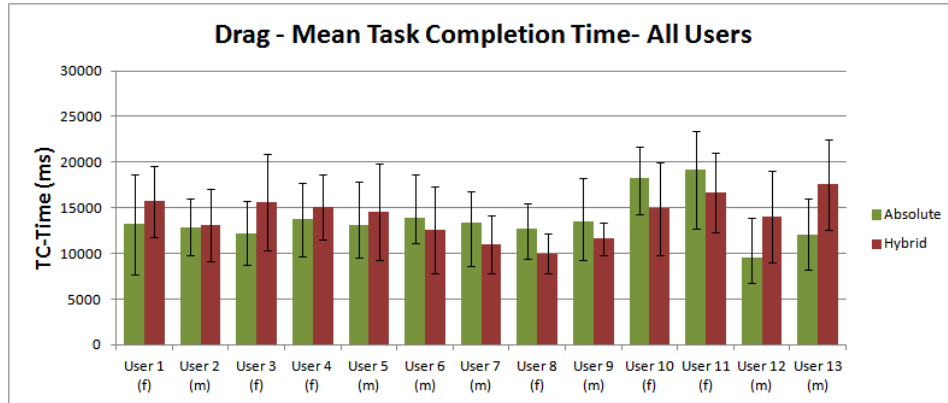


Figure 4.7.: Mean task completion for each user including the standard deviation.

a significant faster task completion time using the *Absolute Pointing* ( $p = 0.001$ ). Whereas user 8 performs significant better using the *HyPoba Pointing* ( $p = 0.013$ ). Other users show similar, but less significant results (Table 4.7). Overall we count 6 users preferring *Absolute Pointing* and 7 users preferring *HyPoba Pointing*.

Table 4.7.: T-statistic for a selection of users performing the drag & drop task.

	Task completion time	Preferred
User 1	$t(25) = 1.404, p = 0.173$	<i>Absolute Pointing</i>
User 3	$t(22) = 2.016, p = 0.056$	<i>Absolute Pointing</i>
User 6	$t(25) = 1.023, p = 0.316$	<i>HyPoba Pointing</i>
User 8	$t(24) = 2.684, p = 0.013$	<i>HyPoba Pointing</i>
User 10	$t(25) = 1.963, p = 0.061$	<i>HyPoba Pointing</i>
User 13	$t(30) = 3.58, p = 0.001$	<i>Absolute Pointing</i>

Those results indicate a strong user dependence. Not only for the task completion time, also the mean values for transition accuracy show large deviation between the users (Figure 4.9). Since the number of level changes already shows significant results for all users we do not provide an additional individual analysis on this aspect.

### Manipulation task

The overall analysis has shown a light preference for *Absolute Pointing* according the task completion time. However comparing the values of each individual user shows similar effects like for the drag & drop task. Figure 4.10 visualizes the mean task completion time in milliseconds. The standard deviation is indicated by the error bars. In order to provide more details of the results, figure 4.11 shows a matrix

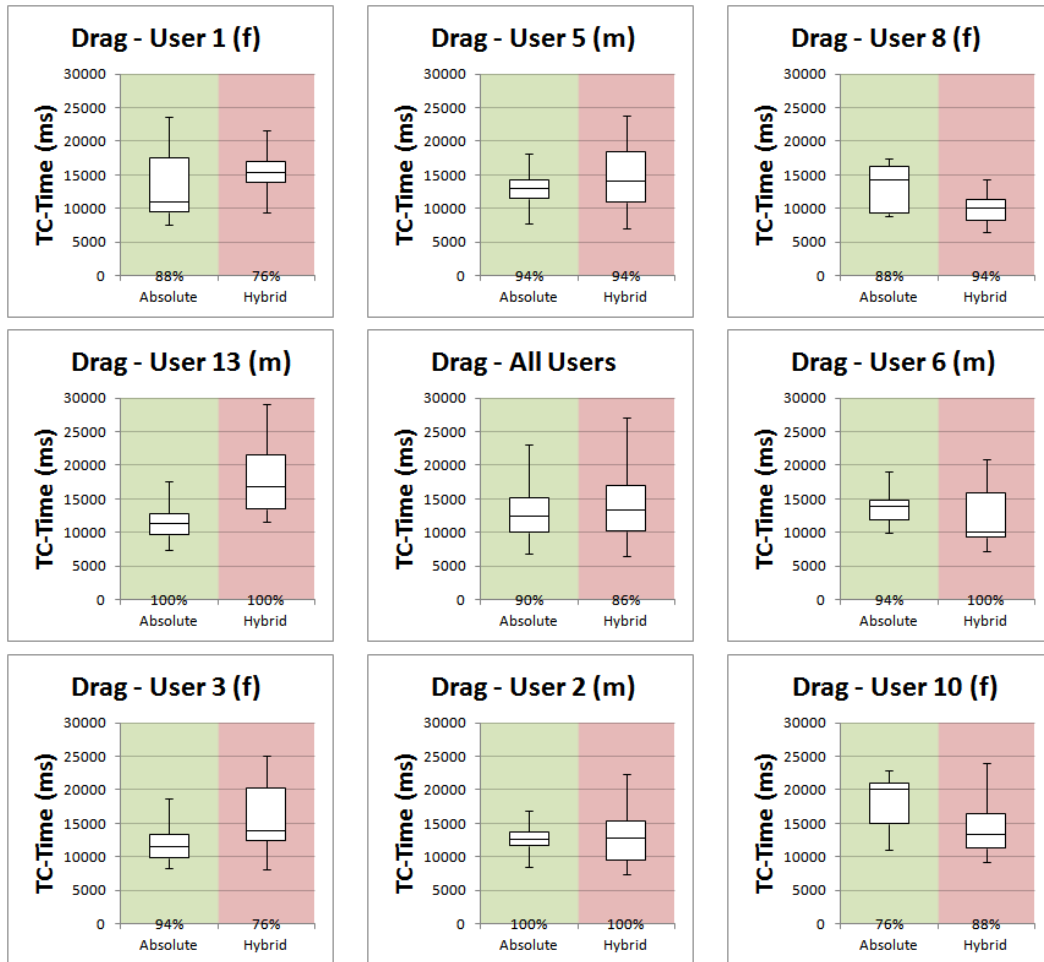


Figure 4.8.: Boxplot diagrams comparing a selection of 8 users and the values of all users. The percentage number at the bottom line indicates the success rate of each task.

of boxplots like the one we used for the drag & drop task. The plot in the center visualizes the values for all users together for comparison.

Comparing the individual users in figure 4.11 and table 4.8 we found several users with better task completion time using *Absolute Pointing*. Those results agree with the overall analysis. However also for this task there are users which performed better using the *HyPoba Pointing*.

The mean transition accuracy comparison between all users (Figure 4.12) shows varying results between the individual users. Taking a look at user 12 we see a significant higher accuracy for *HyPoba Pointing* ( $t(29) = 2.097$ ,  $p = 0.045$ ). Whereas user 5 prefers *Absolute Pointing* with more than 90% confidence ( $t(24) = 1.782$ ,  $p = 0.087$ ). For the transition accuracy we note 5 users which performed similar with both techniques, 3 users which perform better using *Absolute Pointing* and 4 users

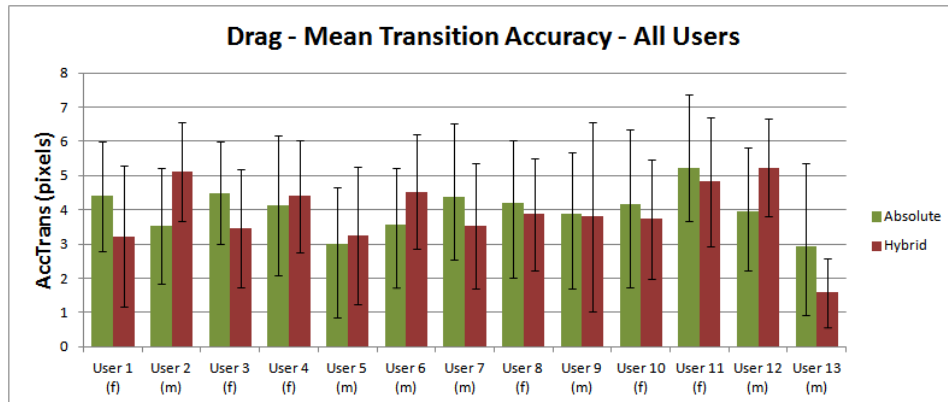


Figure 4.9.: Mean transition accuracy of each user for the drag &amp; drop task

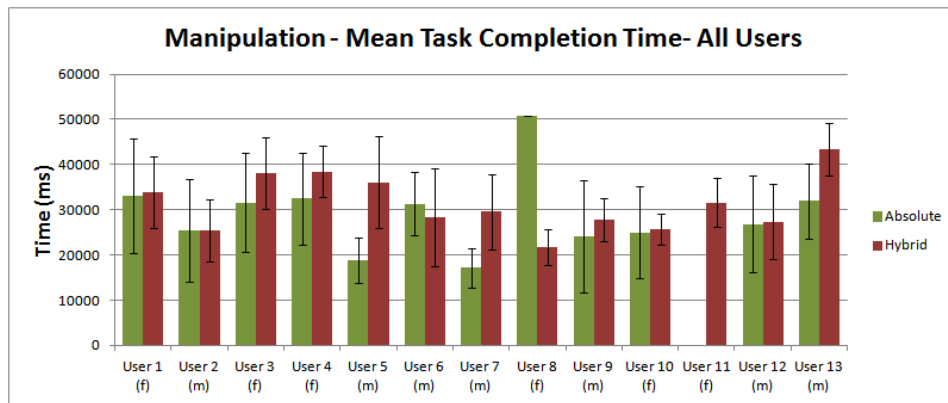


Figure 4.10.: Mean task completion time of each individual user for the manipulation task.

achieving higher accuracy using *HyPoba Pointing*. We did not rate user 11 since she only had one successful sample using *Absolute Pointing*.

For both, the drag & drop task and the manipulation task we found significant differences between *Absolute Pointing* and *HyPoba Pointing* for individual users. We assume that the parameters for *HyPoba Pointing* have a strong impact on users interaction. For users interacting well using *HyPoba Pointing* the choice of parameters fit well. For the other users, its very likely the chosen parameters did not match the users preference. A long term study with several sessions for each user could answer this question.

### 4.3.3. Object Size Analysis

The analysis of the different object sizes should be used as an indicator for accuracy. Placing the cursor-icons on small objects is considered to be more difficult and requires a higher accuracy of the pointing technique. However scaling down an

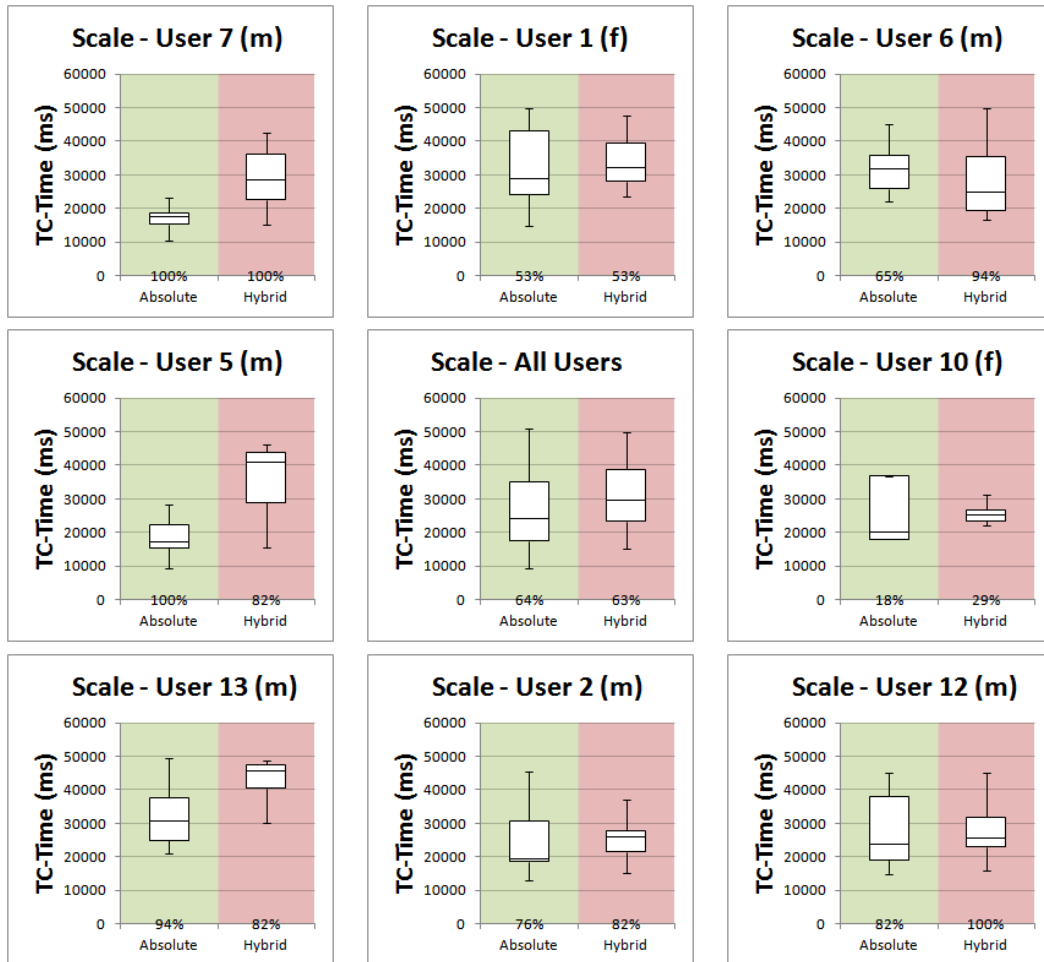


Figure 4.11.: Boxplot diagrams comparing a selection of 8 users performing the manipulation task. The values of all users are displayed in the center. The percentage number at the bottom line indicates the success rate of each task.

image also requires high accuracy as small cursor movements have a great impact on object rotation and size. We therefore first provide the analysis for the drag & drop task.

### Drag & drop task

As stated in the hypotheses we expected *HyPoba Pointing* to perform better for small objects than *Absolute Pointing*. For large objects less accuracy is required and therefore *Absolute Pointing* was expected to perform better. The success rate of the drag and drop task is grouped by object size (Figure 4.13a). The success rate of *Absolute Pointing* for small objects is 11% smaller than the one of *HyPoba Pointing* (Abs = 77%; Hyb = 88%). While the large objects provide equal results (Abs = 96%; Hyb = 96%) *Absolute Pointing* provides better success rates for the medium sized objects (Abs = 96%; Hyb = 90%).



Table 4.8.: T-statistic for a selection of users performing the manipulation task.

	Task completion time	Preferred
User 5	$t(19) = 5.767, p < 0.001$	<i>Absolute Pointing</i>
User 6	$t(24) = 0.886, p = 0.385$	<i>HyPoba Pointing</i>
User 7	$t(24) = 5.455, p < 0.001$	<i>Absolute Pointing</i>
User 10	$t(2) = 0.126, p = 0.911$	<i>HyPoba Pointing</i>
User 12	$t(26) = 0.149, p = 0.883$	<i>HyPoba Pointing</i>
User 13	$t(26) = 4.342, p < 0.001$	<i>Absolute Pointing</i>

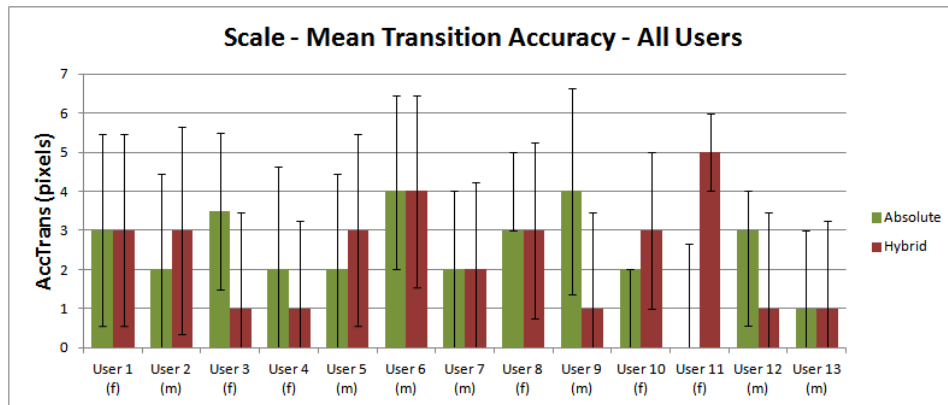


Figure 4.12.: Mean transition accuracy values for each user showing varying results.

In analogy to the overall analysis we inspected task completion time, transition accuracy and the number of level changes for each object size. In Figure 4.13b the task completion time shows an overall lower median for *Absolute Pointing*. However none provides a significant difference ( $t_{small}(99) = 1.571, p = 0.119$ ;  $t_{medium}(134) = 1.006, p = 0.316$ ;  $t_{large}(143) = 0.184, p = 0.854$ ).

The analysis of the transition accuracy also shows a better performance of *HyPoba Pointing* for the small ( $p_{small} = 0.603$ ) and large ( $p_{large} = 0.648$ ) objects (Figure 4.13c). The number of level changes show significant results for all object sizes ( $t_{small}(101) = 3.268, p = 0.001$ ;  $t_{medium}(125) = 2.202, p = 0.030$ ;  $t_{large}(128) = 3.121, p = 0.002$ ).

Regarding the different object sizes we noticed better success rate and transition accuracy for *HyPoba Pointing*. However *Absolute Pointing* results in shorter task completion times. The number of level changes does not show noticeable differences for the individual object sizes, but overall remains significantly lower for *HyPoba Pointing*.

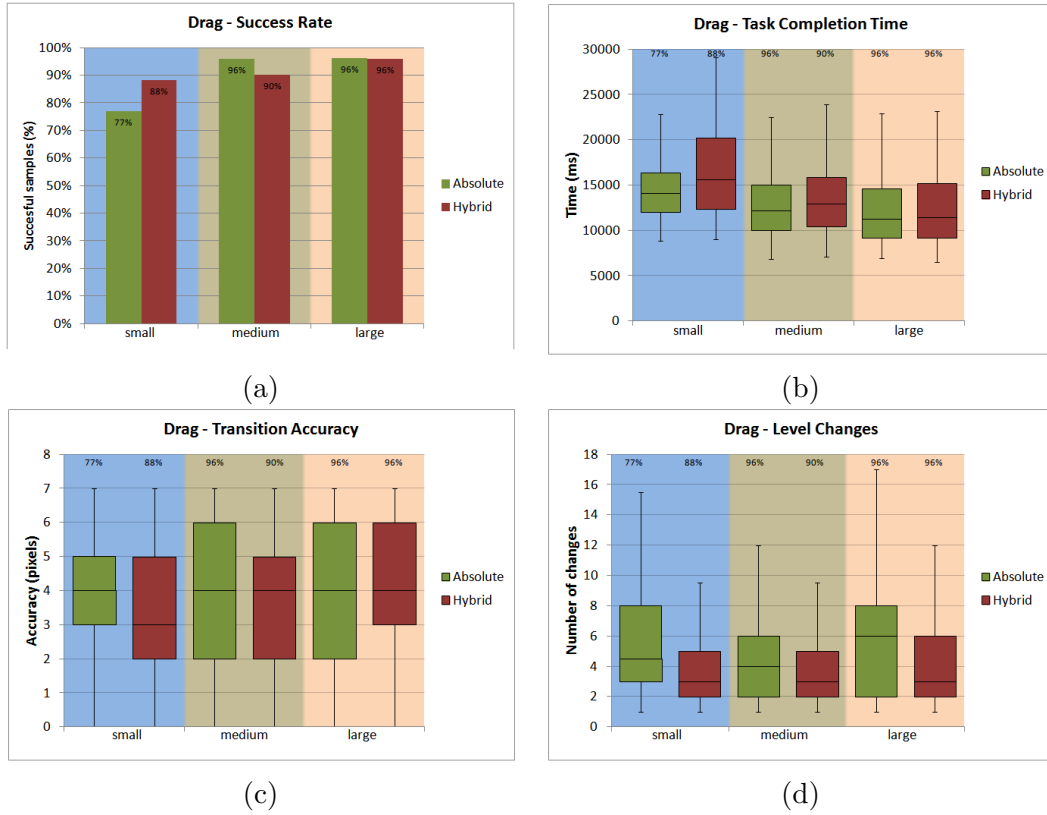


Figure 4.13.: Analysis of target size for drag & drop: (a) The success rate for drag & drop indicates a lower success rate for small objects using *Absolute Pointing*. (b) The difference of the task completion time is getting smaller for larger objects. (c) The transition accuracy does not indicate significant results. (d) *HyPoba Pointing* performs better according to the number of level changes for all target sizes.

Another interesting effect is the progression of the task. If we compare the first half of samples of any task and the last half we found better results for the last half. This effect has to be expected as all users were novice users. However we realized the learning effect for *Absolute Pointing* was much smaller than for *HyPoba Pointing*. This effect is illustrated in figure 4.14a for the task completion time and 4.14b for the number of level changes. Additionally we calculated the independent sample t-test. With more than 93% confidence the task completion time is lower for the second half when using *HyPoba Pointing* (4.9).

### Manipulation task

For the manipulation task, the size of the object has two different impacts on the results which makes the interpretation of the results more difficult. For this task, two different patterns of interaction could be observed. Some users first dragged the object to the designated area using one hand. After that, they grabbed the source-object with the second hand and enlarges the object to fit to the target-object.

Table 4.9.: T-statistic of the drag &amp; drop task comparing the task completion time of the first half and the last half of samples for each user.

	Absolute	Hybrid
Small	$t(45) = 0.520$ ; $p = 0.606$	$t(51) = 1.906$ ; $p = 0.062$
Medium	$t(69) = 0.604$ ; $p = 0.548$	$t(63) = 1.539$ ; $p = 0.065$
Large	$t(64) = 0.101$ ; $p = 0.920$	$t(68) = 0.886$ ; $p = 0.379$

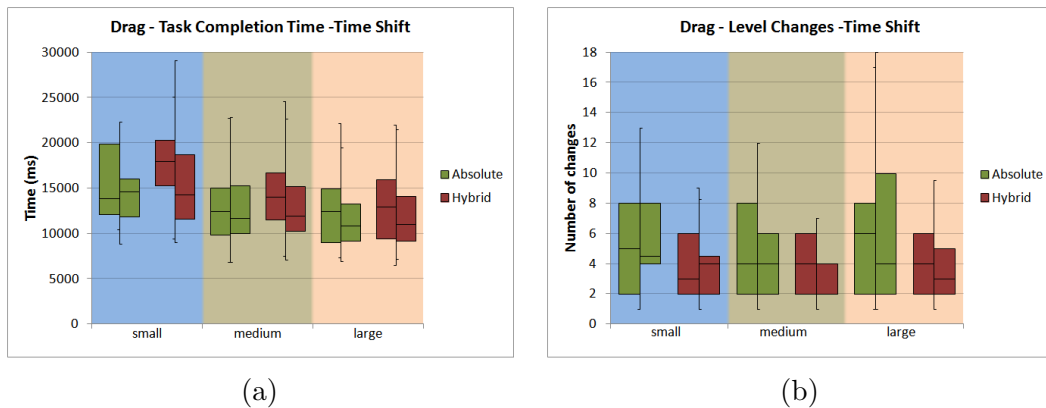


Figure 4.14.: Special boxplot visualization for time shift results. The left side of the box indicates the first half of the samples and the right side of the box the last half of samples. This allows the visualization of the shift of mean values and deviation for samples which have been performed at the beginning of the task and from those at the end of the task

Therefore positioning both cursor on the small object is difficult, while enlarging the object is more simple. If the positions of both cursors are close to the center of the small object, manipulation is very difficult since small cursor movements have a large impact on the object. Due to simple mathematics, a changing one cursor position by as a higher impact on the rotation angle of the object when the object is small. As the level of precision is not relative to the object size, scaling down objects gets more difficult the smaller an object gets. Therefore we do not rate the small objects as more difficult than large objects. This effect is possibly also the reason why the manipulation task did not show the expected results as for the drag & drop task. The success rate for *Absolute Pointing* and *HyPoba Pointing* show almost equal rates (Figure 4.15a). Only the large objects show a slightly higher success rate for the absolute approach (Abs = 75%; Hyb = 64%).

According to the task completion time, *Absolute Pointing* performed better than *HyPoba Pointing*, for large objects even significantly better ( $t_{large}(104) = 3.646$ ,  $p < 0.001$ ). The visual analysis of the illustration in figure 4.15b justifies that finding. The analysis of the transition accuracy does not favor any of the techniques.

A light preference of *Absolute Pointing* can be found for small objects and on the opposite *HyPoba Pointing* performs better for large objects. However none of these results provide significant values. Like for the drag & drop task the *HyPoba Pointing* requires significant less attempts than *Absolute Pointing* ( $t_{small}(85) = 1.885$ ,  $p = 0.063$ ;  $t_{medium}(58) = 2.973$ ,  $p = 0.004$ ;  $t_{large}(83) = 3.226$ ,  $p = 0.002$ ).

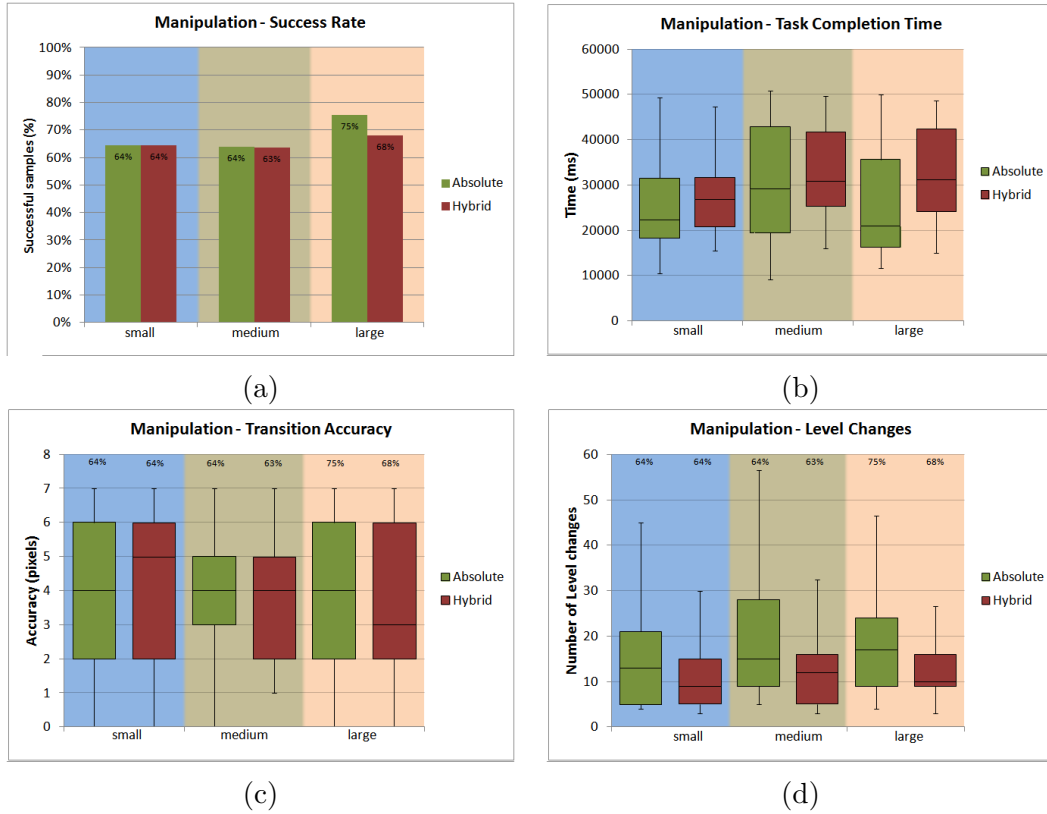


Figure 4.15.: Analysis of the object size for the manipulation task.

#### 4.3.4. Gender Analysis

Even though the gender analysis was not in the focus of our experiment, we found surprising differences between the female and male users. Overall the success rate for female user was lower, however for the manipulation task the success rate was more than 50% smaller than those of the male user. In figure 4.16 this effect is illustrated. Comparing the number of successful samples for each user using an independent sample t-test show some significant results. Significant differences could be proofed for both techniques in the manipulation task (*Absolute Pointing*  $t(6) = 2.980$ ;  $p = 0.025$  and *HyPoba Pointing*  $t(10) = 5.028$ ;  $p < 0.001$ ). The drag & drop task shows less significant results (*Absolute Pointing*  $t(5) = 2.4328$ ;  $p = 0.059$  and *HyPoba Pointing*  $t(8) = 0.706$ ;  $p = 0.500$ ).

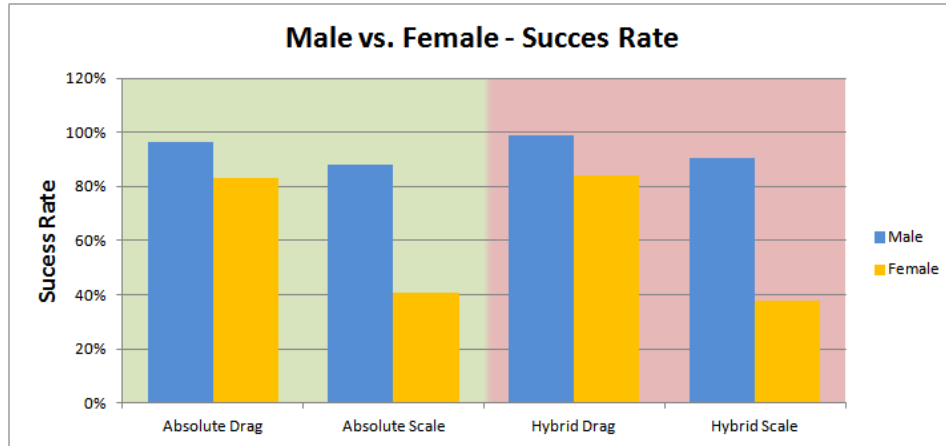


Figure 4.16.: Successrates for comparison between male and female users

Despite the much lower success rate also the task completion time indicates gender dependent differences. Male user perform significantly faster in the drag & drop task using *Absolute Pointing* ( $t(164) = 2.853$ ;  $p = 0.005$ ) compared to the female users. While the male users have significant shorter task completion times using *Absolute Pointing* ( $t_{time}(203) = 1.853$ ,  $p = 0.065$ ), the task completion time of female users show almost no differences between the two techniques. ( $t_{time}(163) = 0.021$ ,  $p = 0.983$ ). Both, male and female user show better results using *HyPoba Pointing* according the number of level changes (see table 4.10).

Table 4.10.: T-statistic comparing the results of the drag &amp; drop task for all male and all female users

	Male	Female
Time Preferred	$t(203) = 1.853$ , $p = 0.065$ <i>Absolute Pointing</i>	$t(163) = 0.021$ , $p = 0.983$ <i>Absolute Pointing</i>
Trans.Acc Preferred	$t(217) = 1.038$ , $p = 0.301$ <i>HyPoba Pointing</i>	$t(164) = 1.688$ , $p = 0.093$ <i>HyPoba Pointing</i>
Level Changes Preferred	$t(219) = 1.832$ , $p = 0.068$ <i>HyPoba Pointing</i>	$t(116) = 5.138$ , $p < \mathbf{0.001}$ <i>HyPoba Pointing</i>

Comparing the manipulation tasks in 4.17 wide varying means are reported for the *Absolute Pointing* ( $t(96) = 3.675$ ;  $p < 0.001$ ) while for *HyPoba Pointing* the mean completion time between male and female users are close by similar variances ( $t(120) = 0.442$ ;  $p = 0.658$ ). The male users perform significant faster using *Absolute Pointing* ( $t(205) = -4.617$ ,  $p < 0.001$ ), while the female user have shorter task completion times using *HyPoba Pointing* ( $t(68) = 0.871$ ,  $p = 0.387$ ). Both, male and

female users show better results using *HyPoba Pointing* according to the transition accuracy and the number of level changes.

However, regarding the varying success rate conclusions are difficult to make. Based on the interpretation of the diagrams and the results of the t-tests we concluded that the male users perform significant better in the bimanual manipulation task. Even though female users performed significantly slower than male user, we found that *HyPoba Pointing* has a higher acceptance for the female users.

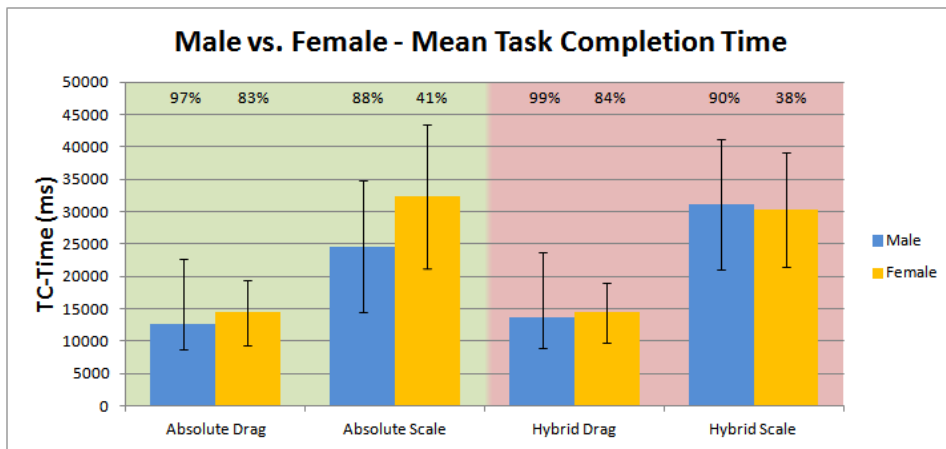


Figure 4.17.: Mean task completion time for male and female users indicating faster interaction for the male users.

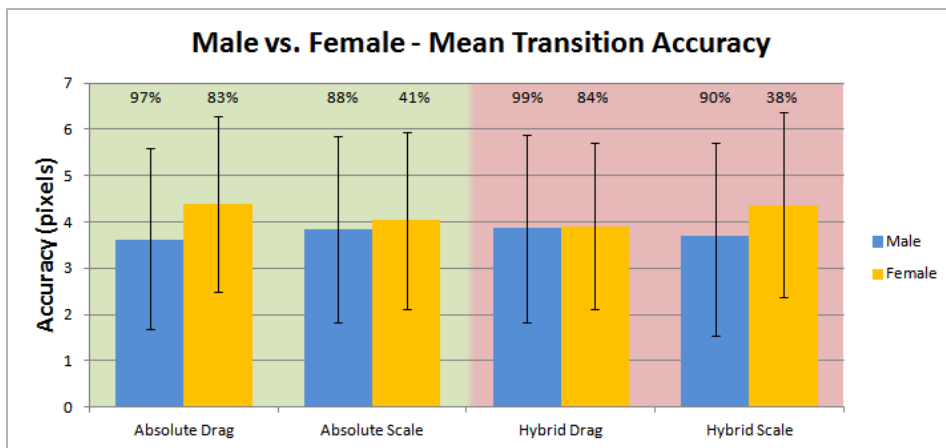


Figure 4.18.: Mean transition accuracy for male and female users show more accurate interaction for male users.

### 4.3.5. Interpretation of the Results

After the experiment we analyzed the data with different scopes. We found that task completion time, transition accuracy and the number of level changes provide

good data for further analysis. We also analyzed target overshooting which did not show differences between the two pointing techniques. Object movement gives feedback of how directly the user could perform the task. For example if one object is located 400 pixels higher (y-axis) than the other, and the user moved the object 800 pixels in y direction, we could analyze how directly the target position is reached. However, the movement data did not show any interesting patterns, neither did the click analysis.

The analysis of task completion time, transition accuracy and number of level changes showed several significant results which lead to interesting interpretations. One of the most interesting result is one we have not been looking for. Especially in the bimanual task male users performed significantly better than female users. And male users perform significantly faster using *Absolute Pointing* while female user perform equally using both techniques. Further experiments with a higher number of users would allow to perform deeper analysis for each gender like we did for both genders together.

The results did not prove, that *Absolute Pointing* has allows faster cross-display object movement (H1). We found a strong dependency on each individual user. While some user interact significantly faster using one technique, others perform significantly better using the other technique. Significant lower number of level changes for *HyPoba Pointing* throughout all analyses proves that *HyPoba Pointing* results in higher accuracy for cross-display object movement (H2). For the bimanual object manipulation *HyPoba Pointing* also performs more accurate. Therefore also hypothesis H4 can be proved. The results indicate a higher accuracy for *HyPoba Pointing* for both tasks. The good learning curve of *HyPoba Pointing* qualifies the lower intuitiveness. The significant faster task completion time of *Absolute Pointing* rejects the hypothesis, that *HyPoba Pointing* allows faster bimanual object manipulation (H3).

In addition we found that only *HyPoba Pointing* has a progressive learning effect. Comparing the first samples and last samples of a task show a significant improvement of task completion time for *HyPoba Pointing*. This results agree with the questionnaires where *Absolute Pointing* was rated more intuitive and *HyPoba Pointing* is meant to have a better learning effect. We assume that for some users the choice of parameters for *HyPoba Pointing* was a good one. However we found different patterns of interaction, which require also different parameters for the pointing techniques.

In general, both techniques performed well for the drag & drop task. Even though the objects on the notebook have been very small, some users could grab them very quickly. The idea of dragging objects between the displays is very intuitive and most

users liked this task. The bimanual task turned out to be more difficult. For several users, especially female users, it turned out to be a problem to synchronize the two cursors with their hands. However, also for this task the users achieved very accurate results. With a little experience, the high task completion time can be divided in half. A comparison with an in official experienced user performing similar amount of samples like all other users together is shown in table 4.11. We therefore recommend the use of *HyPoba Pointing*, even though it can take longer to get used to it. With a little experience and good choice of the parameters *HyPoba Pointing* can perform much faster and more accurate than *Absolute Pointing*.

Table 4.11.: Comparison of the mean task completion times of an in official experienced user.

	Participants	Experienced User
<i>Absolute Pointing</i> Drag & Drop	13380 ms	9147 ms
<i>Absolute Pointing</i> Manipulation	26572 ms	11852 ms
<i>HyPoba Pointing</i> Drag & Drop	14035 ms	9324 ms
<i>HyPoba Pointing</i> Manipulation	30836 ms	16578 ms



## 5. Two Hands and Beyond

In this work we proposed a novel pointing technique for bimanual cross display interaction. We successfully tested the technique compared to the common absolute pointing. However, during this work many ideas have been rejected due to several reasons. Some of those ideas and concepts have already been partly implemented. In the following we introduce those ideas to enhance the power of bimanual cross-display interaction.

### 5.1. Future Ideas

#### 5.1.1. Remote View

Dragging an object from one display to another is simple, easy to understand and the interaction itself is not complicated at all. Arranging several objects to multiple displays many drag & drop interactions have to be performed. This can be time consuming and annoying. The *Remote View* is a concept which has already been implemented in *MultiDragger*. Due to the changing focus of this work, we could not prosecute this concept. The idea is very simple and profits from the bimanual interaction. It simply displays the content of on display in a scalable window on the second display. For example, the user performs a grab gesture, which implied to grab the entire display content, on the first display. While keeping the grab gesture, he can drag the window with the content of the first display on to any other display. He can either drop the *Remote View* or even drag single object using the other hand onto the second display. The *Remote View* is synchronized with the source display which even allows to manipulate objects in the *Remote View*. In figure 5.1 two screenshots of the working prototype show the visualization of the remote view and its content.

#### 5.1.2. Non-Display Objects

The *Remote View* could also be used to visualize contents of non-display objects. In figure 2.1 we illustrated the different tasks for collaborative work. In order to support the most realistic re-build of the use of traditional media, we added non display

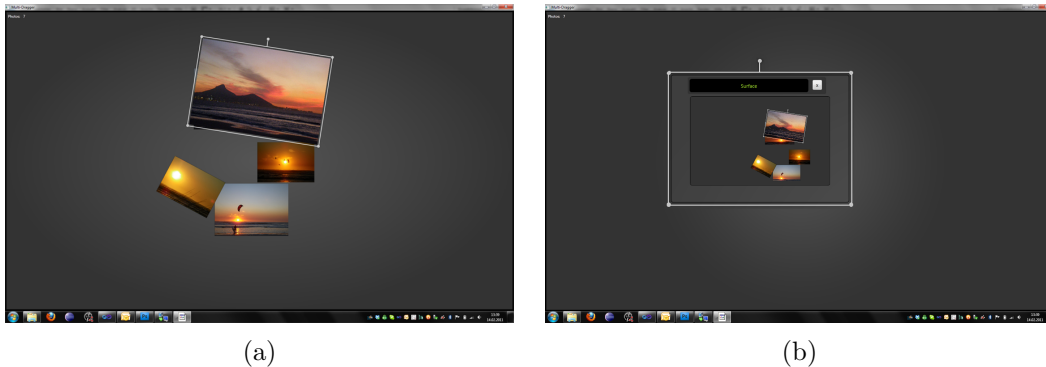


Figure 5.1.: Remote View: (a) The content of display 1 can be dragged to display 2. (b) The content of display 1 inside the Remote View

devices such as printer, garbage bin and digital paper. We already introduced how objects can be thrown into the garbage bin. The use of the *Remove View* transforms the garbage bin into a recycle bin. Since a grab gesture can pick up all contents from a display, it could also pick up all deleted objects. Any deleted object can be restored by pointing at the physical garbage bin, perform a grab gesture and drag the entire content to the desired display. Any object can be restored by dragging it out of the remote view. This technique can be extended to any objects in the room. For example, a book shelf could be used for storing objects in a certain physical folder. The physical action of storing can help to relocate objects stored in the past. Display devices have the big advantage that they can visualize cursor positions and off-screen positions using halos. Non-display objects do not provide any feedback to the user. [Xiao et al. \[2011\]](#) provide a solution for off-screen visualization using laser-beam projections. Highlighting selected objects using a spotlight or acoustic feedback could be applied as well. Further research will be necessary to find an optimal solution. Nevertheless we assume, that the use of physical non-display and even non-digital object can enhance MDEs in different ways.

### 5.1.3. Mobile Display Integration

Integrating with small mobile displays and other small devices require several considerations. First, the displays are very small and therefore more difficult to interact with, second the mobility of the device can disturb the interaction with large displays.

#### Magnetic Displays

Pointing at a small target on a large screen from a large distance is a difficult task. Pointing at a small display from a large distance is even more difficult since the user cannot follow the cursor when it is off the displays [[Xiao et al., 2011](#)]. Accessing

data from a personal device which lays on the table several meters away can be very difficult. Without additional support, the user has to walk over to the device. Magnetic displays could help to make pointing at small devices easier. A virtually enlarged bounding-box around the display can prevent losing the focus of an object. If the user is pointing at this bounding-box the cursor is already displayed on the device. However, if multiple small objects are positioned close to each other, this method has to be implemented carefully in order to maintain a natural pointing behavior.

### **Invisible Displays**

Mobile displays can be carried around, which means, that one users' device can disturb another users' interaction. For example if a user wants to drag an item between two displays and another user walks in between with another device, then the interaction can be faulty. One possibility of handling the occurrence of multiple mobile devices is to restrict the access on the device to its owner. Regarding the privacy requirement, this is likely to be the case. However if one device does not belong to the user, or the owner grants access for other users the problem remains. One solution to this problem could be to set fast moving displays to invisible for the intersection algorithm. This can avoid unintended intersections with moving devices. In addition, if a user is interacting on a display and a mobile display moves in between, the focus can remain on the active display. Such solutions have to be applied carefully since it is very difficult to predict which movement was intended and which was not.

## **5.2. Conclusion**

In this thesis we have introduced the combination of three research areas in HCI. Cross-display interaction (MDEs), gesture interaction and pointing interaction can be combined with bimanual cross-display interaction, a concept which enhances common multi display environments. We defined a set of requirements with the special focus on collaborative tasks in MDEs. With the above mentioned extensions to our approach most of these requirements can be fulfilled.

**Privacy:** The use of mobile devices allows the user to keep the required privacy. He can keep personal data saved on the private device. In addition the private device supports private interaction like taking notes or similar.

**Awareness:** Pointing interaction allows each participation user to see what other users are doing.

**Shareability:** Users can drag and drop object from the private device onto the public device and back. However, due to the privacy requirement only the owner of a private device can interact with his own device.

**Mobility:** Users are free to move in the environment and can interact with any device from any position.

**Reachability:** Each user can reach any display in the environment. However, small devices tend to be difficult to select from large distances.

**Accuracy:** *HyPoba Pointing* increases the accuracy of the pointing interaction compared to *Absolute Pointing*. The experiment has shown, that pixel-wise interaction is possible for experienced users.

**Pointing and Selection Speed:** Fast object movement between the displays has been suspect to the experiment. Results have shown faster task completion time for *Absolute Pointing*.

**Comfortable Use:** The re-design of the cotton gloves increased the wearing comfort. Considering the advances in camera technology, in the near future the gloves are not necessary anymore. The increased mental and physical demand for the difficult bimanual manipulation task is the only drawback we have to accept. However, tasks requiring less precision can be performed very easily.

The analysis of related work in all three research fields have shown many interesting approaches for cross-display interaction. Gestural interaction was predicted to have a high potential regarding collaborative tasks as we described for our scenario. However gestural interaction also has several drawbacks. As gestural interaction works fine for large displays, privacy is difficult to handle on these. Additional private displays can enhance existing MDEs and provide privacy without reducing the mobility of the user. The use of small devices imposes additional concerns about pointing accuracy.

*HyPoba Pointing* is a new hybrid pointing technique which allows exact cursor positioning while maintaining the directness required to select the various displays in an MDE. The results of the experiment comparing *HyPoba Pointing* and *Absolute Pointing* have shown significant better accuracy for *HyPoba Pointing*, whereas *Absolute Pointing* results in shorter task completion time. The preferences between the users showed large variances. Some users generally preferred *Absolute Pointing* others preferred *HyPoba Pointing*. Further investigations and evaluations would be necessary to optimize *HyPoba Pointing*. The results have shown that the parameters need to be adjusted individually for each user. Having a large amount of sample data can help find patterns of hand movements which prefer one of the techniques. We assume that those patterns could be retrieved automatically if the hand

movement of the interacting user is compared with the known patterns from the samples. An automatic parameter adaption could be provided which allows each user to interact in the most natural way.

Beside the comparison of these pointing techniques the experiment has shown the common bimanual gestures for scaling and rotating objects work well also for distant midair interaction. The experiment required very precise interaction which turned out to be error prone and physically demanding. However, the initially described scenario does not require such high accuracy. For example, dragging an object from a personal device to a large device in order to present to other persons, does not require exact position neither does the size matter. The important thing is that the objects can quickly be taken from the personal display and arranged on the large displays. With further improvements on camera technology and pointing algorithms we believe that bimanual cross-display interaction has a high potential for any kind of collaborative work in multi-display environments.



## Bibliography

- Argelaguet, F. and Andujar, C. (2009). Visual feedback techniques for virtual pointing on stereoscopic displays. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST '09*, pages 163–170, New York, NY, USA. ACM. [2.3.3.1](#)
- Bader, T., Heck, A., and Beyerer, J. (2010). Lift-and-drop: crossing boundaries in a multi-display environment by airlift. In *AVI*, pages 139–146. [\(document\)](#), [2.1.3](#), [2.3](#)
- Balakrishnan, R. (2004). “beating fits” law: virtual enhancements for pointing facilitation. *Int. J. Hum.-Comput. Stud.*, 61:857–874. [2.3.3.2](#)
- Balakrishnan, R. and Hinckley, K. (2000). Symmetric bimanual interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '00*, pages 33–40, New York, NY, USA. ACM. [2.2.3](#)
- Baudel, T. and Beaudouin-Lafon, M. (1993). Charade: remote control of objects using free-hand gestures. *Commun. ACM*, 36:28–35. [2.2](#)
- Baudisch, P., Cutrell, E., Hinckley, K., and Eversole, A. (2005). Snap-and-go: Helping users align objects without the modality of traditional snapping. In *CHI*, pages 301–310. Press. [2.3.3.2](#)
- Bieg, J. (2008). Laserpointer and eye gaze interaction - design and evaluation. mastersthesis, University of Konstanz. [2.3.1](#), [2.3.1.1](#)
- Bolt, R. A. (1980). “put-that-there”: Voice and gesture at the graphics interface. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques, SIGGRAPH '80*, pages 262–270, New York, NY, USA. ACM. [2.2.2](#), [2.3.2](#), [2.3.3](#)
- Boring, S., Baur, D., Butz, A., Gustafson, S., and Baudisch, P. (2010). Touch projector: mobile interaction through video. In *CHI*, pages 2287–2296. [\(document\)](#), [2.1.3](#), [2.3](#)
- Buxton, B. (2010). A directory of sources for input technologies. [2.3.1](#)

- 
- Buxton, W. and Myers, B. (1986). A study in two-handed input. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '86, pages 321–326, New York, NY, USA. ACM. [2.2.3](#)
- Casiez, G., Vogel, D., Balakrishnan, R., and Cockburn, A. (2008). The impact of control-display gain on user performance in pointing tasks. *HUMAN-COMPUTER INTERACTION*, 23(3):215–250. [2.3.1.2](#)
- Fäh, S. (2010). Bimanual gesture interaction in multi-display environments. Seminar for the Master-Project in Human-Computer Interaction. ([document](#)), [2.1.2](#), [2.1.3](#), [2.1.3](#), [3](#), [2.1](#)
- Fäh, S. (2011). 2hands4displays: Visual tracking for bimanual gesture recognition and cross-display interaction. Master-Project in Human-Computer Interaction. ([document](#)), [3.3.1](#), [3.1](#), [3.2](#), [3.3](#), [3.3.3](#), [3.4](#), [3.3.4](#), [3.4](#), [3.4.3](#)
- Fikkert, F. W. (2010). *Gesture Interaction at a Distance*. PhD thesis, Universiteit Twente, Enschede. SIKS Dissertation Series No. 2010-07. [2.2](#), [2.2.2](#), [3.6](#)
- Foehrenbach, S. (2009). Hand gesture interaction for large high-resolution displays: Design and evaluation. mastersthesis, University of Konstanz. [2.2](#), [3](#), [3.1](#), [3.2](#), [3.1](#), [3.7](#)
- Foehrenbach, S., König, W. A., Gerken, J., and Reiterer, H. (2009). Tactile feedback enhanced hand gesture interaction at large, high-resolution displays. *Journal of Visual Languages and Computing, Special Issue on Multimodal Interaction Through Haptic Feedback*, 20:341–351. [2.2](#), [2.3.2](#), [3.3.1](#)
- Forlines, C., Vogel, D., and Balakrishnan, R. (2006). Hybridpointing: fluid switching between absolute and relative pointing with a direct input device. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 211–220, New York, NY, USA. ACM. [2.3.3.2](#)
- Frees, S., Kessler, G. D., and Kay, E. (2007). Prism interaction for enhancing control in immersive virtual environments. *ACM Trans. Comput.-Hum. Interact.*, 14. ([document](#)), [2.3.3.2](#), [2.10](#)
- Geyer, F., Pfeil, U., Höchtel, A., Budzinski, J., and Reiterer, H. (2011). Designing reality-based interfaces for creative group work. In *To appear in C&C'11: Proceedings of the 8th ACM Conference on Creativity and Cognition, Atlanta, USA*. ACM Press. ([document](#)), [1](#), [2.1](#), [3.1](#)
- Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *The Journal of Motor Behaviour*, 19:486–517. [2.2.3](#)



- Harrison, C., Benko, H., and Wilson, A. D. (2011). Omnitouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 441–450, New York, NY, USA. ACM. [2.1.3](#)
- Hascoet, M. (2003). Throwing models for large displays. In *HCI'03*. [2.1.3](#)
- Hinckley, K. (2008). Input technologies and techniques. In Jacko, J. A. and Sears, A., editors, *The human-computer interaction handbook*, pages 161–176. L. Erlbaum Associates Inc. [2.3.1](#), [2.3.1.1](#), [2.3.1.2](#)
- Hinckley, K., Pausch, R., Goble, J. C., and Kassell, N. F. (1994). A survey of design issues in spatial input. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, UIST '94, pages 213–222, New York, NY, USA. ACM. [2.3.2](#), [2.3.3.1](#)
- ISO-9241-16:1999 (1999). Ergonomic requirements for office work with visual display terminals (vdts) - part 9: Guidance on usability. Part 9: Guidance on usability. [2.1.2](#), [2.3.1](#), [4.2.2](#)
- Jenabi, M. and Reiterer, H. (2010). Primitive interaction tasks for multi-display environments (prime): A hands-on approach. In *AVI 2010: the International working conference on advanced visual interfaces, workshop on coupled display visual interfaces*. [2.1.2](#), [2.1.3](#), [3.1](#)
- Jota, R., Nacenta, M. A., Jorge, J. A., Carpendale, S., and Greenberg, S. (2010). A comparison of ray pointing techniques for very large displays. In *Proceedings of Graphics Interface 2010*, GI '10, pages 269–276, Toronto, Ont., Canada, Canada. Canadian Information Processing Society. ([document](#)), [2.1.2](#), [2.3.2](#), [2.7](#), [2.3.3.1](#), [6](#)
- Karam, M. and Schraefel, M. C. (2005). A study on the use of semaphoric gestures to support secondary task interactions. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, pages 1961–1964, New York, NY, USA. ACM. [2.2.2](#)
- Kendon, A. (2004). Gesture. visible action as utterance. [2.2](#)
- König, W. A. (2010). *Design and evaluation of novel input devices and interaction techniques for large, high-resolution displays*. phdthesis, University of Konstanz. ([document](#)), [2.2.1](#), [2.2.1](#), [2.3.1](#), [2.3.1.1](#), [2.3.1.2](#), [2.3.3.2](#), [2.9](#), [3.6](#), [4.2.2](#)
- König, W. A., Böttger, J., Völzow, N., and Reiterer, H. (2008). Laserpointer-interaction between art and science. In *IUI'08: Proceedings of the 13th international conference on Intelligent User Interfaces*, pages 423 – 424. ACM Press. Demonstration Session. ([document](#)), [2.3.2](#), [2.7](#), [2.3.3.1](#), [3.1](#)

- König, W. A., Gerken, J., Dierdorf, S., and Reiterer, H. (2009). Adaptive pointing design and evaluation of a precision enhancing technique for absolute pointing devices. In *Interact 2009: Proceedings of the the twelfth IFIP conference on Human-Computer Interaction*, pages 658–671. Springer. (document), 2.3.3.2, 2.3.3.2, 2.10, 3.6, 4.2.3
- MacKenzie, I. S. (1995). *Input devices and interaction techniques for advanced computing*, pages 437–470. Oxford University Press, Inc., New York, NY, USA. 2.3.1, 2.3.1.2
- Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E., and Smith, J. E. K. (1988). Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological Review*, 95:340–370. 2.3.3.2
- Myers, B. A., Bhatnagar, R., Nichols, J., Peck, C. H., Kong, D., Miller, R., and Long, A. C. (2002). 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*, CHI '02, pages 33–40, New York, NY, USA. ACM. (document), 2.2.1, 2.2
- Oh, J.-Y. and Stuerzlinger, W. (2002). Laser pointers as collaborative pointing devices. 2.3.1.2
- Prante, T., Streitz, N. A., and Tandler, P. (2004). Roomware: Computers disappear and interaction evolves. *IEEE Computer*, 37(12):47–54. 1
- Quek, F., McNeill, D., Bryll, R., Duncan, S., Ma, X.-F., Kirbas, C., McCullough, K. E., and Ansari, R. (2002). Multimodal human discourse: gesture and speech. *ACM Trans. Comput.-Hum. Interact.*, 9:171–193. 2.2.2
- Reiterer, H., Heilig, M., Rexhausen, S., and Demarmels, M. (2010). Idee der blended library - neue formen der wissensvermittlung durch vermischung der realen und digitalen welt. In *Ein neuer Blick auf Bibliotheken*, pages 108–116. Dinges & Frick, Wiesbaden. 3.1
- Rekimoto, J. (1997). Pick-and-drop: a direct manipulation technique for multiple computer environments. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 31–39, New York, NY, USA. ACM. 2.2.2
- Shoemaker, G., Tang, A., and Booth, K. S. (2007). Shadow reaching: a new perspective on interaction for large displays. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 53–56, New York, NY, USA. ACM. 2.3.3.1

- Sturman, D. J. (1992). *Whole-Hand Input*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA. [2.2](#)
- Terrenghi, L., Quigley, A., and Dix, A. (2009). A taxonomy for and analysis of multi-person-display ecosystems. *Personal Ubiquitous Comput.*, 13:583–598. [2.1.2](#)
- Ulinski, A. C., Wartell, Z., Goolkasian, P., Suma, E. A., and Hodges, L. F. (2009). Selection performance based on classes of bimanual actions. In *Proceedings of the 2009 IEEE Symposium on 3D User Interfaces, 3DUI '09*, pages 51–58, Washington, DC, USA. IEEE Computer Society. ([document](#)), [2.2.3](#), [2.6](#), [2.2.3](#), [3.6](#)
- Vogel, D. and Balakrishnan, R. (2005). Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology, UIST '05*, pages 33–42, New York, NY, USA. ACM. ([document](#)), [2.1.2](#), [2.3.2](#), [2.7](#), [2.3.3.1](#), [2.3.3.2](#), [2.8](#), [7](#), [2.3.3.2](#), [3.1](#), [3.2](#), [3.1](#), [3.4.3](#), [3.6](#), [4.2.3](#)
- Vyas, D., Heylen, D., and Nijholt, A. (2009). Collaborative practices that support creativity in design. In *Proceedings of the 11th European Conference on Computer-Supported Cooperative Work*, pages 151–170. [2.1](#)
- Warr, A. and O’Neill, E. (2005). Understanding design as a social creative process. In *Proceedings of the 5th conference on Creativity & cognition, C&C '05*, pages 118–127, New York, NY, USA. ACM. [2.1.1](#)
- Weiser, M. (1999). The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3:3–11. [2.1.3](#)
- Wigdor, D., Jiang, H., Forlines, C., Borkin, M., and Shen, C. (2009). Wespace: the design development and deployment of a walk-up and share multi-surface visual collaboration system. In *CHI*, pages 1237–1246. ([document](#)), [2.1.2](#), [2.1.3](#), [2.3](#)
- Wilson, A. D. and Benko, H. (2010). Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology, UIST '10*, pages 273–282, New York, NY, USA. ACM. ([document](#)), [2.1.3](#), [2.5](#), [2.3.2](#)
- Wingrave, C. A., Tintner, R., Walker, B. N., Bowman, D. A., and Hodges, L. F. (2005). Exploring individual differences in raybased selection; strategies and traits. In *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality, VR '05*, pages 163–170, Washington, DC, USA. IEEE Computer Society. [2.3.3.1](#)
- Xiao, R., Nacenta, M. A., Mandryk, R. L., Cockburn, A., and Gutwin, C. (2011). Ubiquitous cursor: a comparison of direct and indirect pointing feedback in multi-display environments. In *Proceedings of Graphics Interface 2011, GI '11*, pages

135–142, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society. [3.5.2](#), [5.1.2](#), [5.1.3](#)

Zigelbaum, J., Browning, A., Leithinger, D., Bau, O., and Ishii, H. (2010). g-stalt: a chirocentric, spatiotemporal, and telekinetic gestural interface. In Coelho, M., Zigelbaum, J., Ishii, H., Jacob, R. J. K., Maes, P., Pederson, T., Shaer, O., and Wakkary, R., editors, *Tangible and Embedded Interaction*, pages 261–264. ACM. [\(document\)](#), [2.1.3](#), [2.4](#), [2.2.2](#)

## A. Appendix

### A.1. Experiment: Welcome Text

#### Herzlich Willkommen

Zunächst möchten wir uns bei Ihnen bedanken, dass Sie sich bereit erklärt haben, an unserer Untersuchung teilzunehmen. Bevor es nun gleich losgeht, wollen wir Ihnen mit Hilfe dieser kurzen Einführung vermitteln, um was es bei dieser Untersuchung geht und welche Rolle Sie dabei spielen.

Multi-Display Umgebungen werden immer häufiger in den unterschiedlichsten Bereichen eingesetzt. Vor allem für kollaborative Aufgaben wie Brainstorming können mehrere Benutzer die Vorteile solcher Systeme nutzen. Obwohl die meisten Displays über Touch-Input bedient werden können, gibt es zahlreiche Situationen in denen auf entfernte Displays zugegriffen werden soll.

Das heutige Szenario bildet folgende Situation ab:

Sie haben auf Ihrem Notebook ein Projekt vorbereitet welches Sie in der Gruppe diskutieren möchten. Da die anderen Gruppenmitglieder ebenfalls eigenes Material zur Diskussion vorbereitet haben kommt es vor, dass jedes Gruppenmitglied mehrmals auf das private Notebook zugreifen muss. Um diesen Vorgang zu erleichtern setzten wir nun die Hand-Gesten-Steuerung ein. Sie haben so die Möglichkeit aus beliebiger Entfernung auf Ihr privates Notebook zuzugreifen und die Inhalte auf einem anderen Display darzustellen.

Im Verlaufe dieser Studie werden Sie die gleichen Aufgaben mit zwei Unterschiedlichen Pointing-Techniken ausführen. Ziel der Studie ist es, herauszufinden welche Techniken für welche Aufgaben bevorzugt werden.

Es stehen bei dieser Untersuchung die Steuerungsgeräte auf dem Prüfstand und nicht Sie als Benutzer. Sie sind vielmehr in der Rolle des Prüfers, welcher uns die Möglichkeit gibt, Benutzungsprobleme mit der Steuerung und dem Display zu erkennen.

Abschließend wünschen wir Ihnen viel Spaß und möchten uns noch einmal für Ihre Teilnahme bedanken!

## Aufgaben

### 1. Drag & Drop:

Sie müssen jeweils ein Objekt von dem Notebook auf das große Display verschieben. Dabei wird Ihnen die Ziel-Position auf dem Display angezeigt. Sobald Sie eine gute Genauigkeit erreicht haben, wird Ihnen ein grüner Haken in dem Objekt angezeigt. Die Aufgabe ist dann erledigt, wenn Sie das Objekt losgelassen haben. Sobald Sie den Startknopf gedrückt haben werden all ihre Handbewegungen und die Bewegungen des Objekts aufgezeichnet. Der Startknopf dient dazu eine einheitliche Ausgangslage für alle Objekte zu schaffen. Die beiliegenden Abbildungen verdeutlichen den Ablauf.

**Diese Teil-Aufgabe wird zweimal für jede Pointing-Technik durchgeführt.**

### 2. Manipulation:

Nach dem Sie nun die Objekte auf dem großen Display platziert haben, müssen Sie diese noch an dem Vorgabe-Objekt ausrichten. Dabei wird die Genauigkeit bezüglich Größe, Position und Rotation berücksichtigt. Auch hier wird Ihnen wieder angezeigt, wenn Sie das Objekt ausreichend genau platziert haben.

**Diese Teil-Aufgabe wird zweimal für jede Pointing-Technik durchgeführt.**

## Signale & Icons



Genauigkeit noch nicht ganz ausreichend



Gute Genauigkeit, das Objekt kann so abgelegt werden

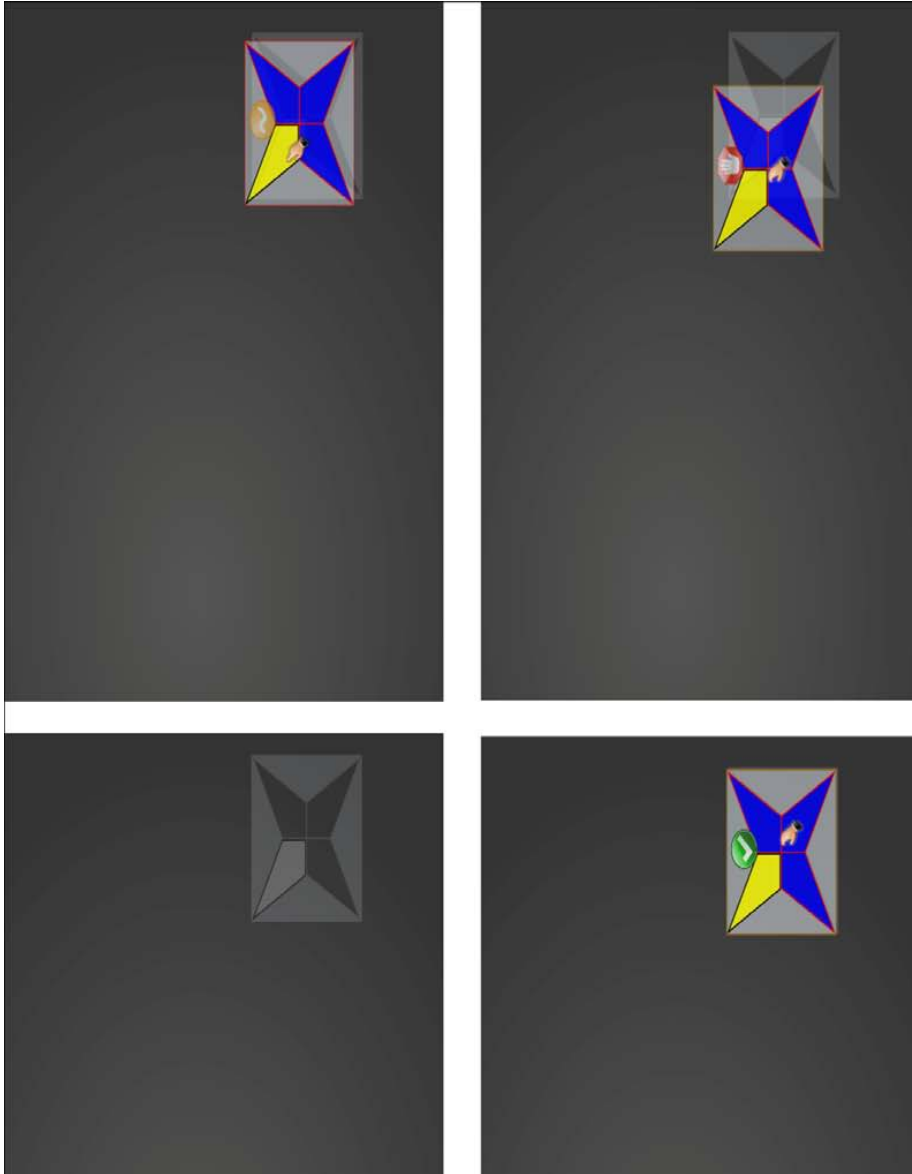


Zeit abgelaufen, das Objekt sofort ablegen

# BIMANUAL POINTING TECHNIQUES FOR CROSS-DISPLAY INTERACTION

## A.1. *Experiment: Welcome Text*

---



## A.2. Experiment: Agreement

### Einverständniserklärung

**Bitte lesen Sie die folgenden Zeilen aufmerksam durch.**

Vertraulichkeit: Alle Informationen, die sich auf teilnehmende Personen beziehen, werden in digitaler Form auf einem Server abgelegt, der nur den am Experiment mitwirkenden wissenschaftlichen Mitarbeitern zugänglich ist. Die Ergebnisse können auch anderen Forscher zur Verfügung gestellt werden, aber keine Informationen, durch die die Teilnehmer identifiziert werden können.

Den Test abbrechen: Die Teilnehmer sind nicht verpflichtet, an dem Test teilzunehmen, und können es jederzeit verlassen. Der Testleiter kann auch entscheiden, die Sitzung wegen Software-Fehler oder aus einem anderen Grund abzubrechen.

Testaufzeichnung: Um eine bessere Auswertung der gewonnenen Daten zu erreichen, wird der Test aufgezeichnet (Audio + Video). Die Aufzeichnungen werden anonymisiert und lediglich zu Auswertungszwecken verwendet.

Freiwilliges Einverständnis: Die oben aufgeführten Informationen wurden mir erklärt und meine Fragen dazu wurden beantwortet. Ich weiß, dass zukünftige Fragen ebenso vom Testleiter beantwortet werden. Mit meiner Unterschrift bestätige ich, dass ich an dem beschriebenen Usability-Test teilnehmen möchte.

---

(Name und Unterschrift des Teilnehmers)

Konstanz, \_\_\_\_\_



## A.3. Experiment: Pre-Test Questionnaire

### Angaben zur Person

---

Alter: \_\_\_\_\_ Geschlecht:  männlich  weiblich

Momentane Tätigkeit: \_\_\_\_\_  
(bei Studium auch den Studiengang bitte nennen)

**Wie viele Stunden verbringen Sie pro Tag an einem Computer?**

- 0 – 1 Stunde   
1 – 2 Stunden   
2 – 3 Stunden   
Mehr als 3 Stunden

**Sind Sie Rechts- oder Linkshänder?**

Rechts  Links

**Haben Sie schon einmal einen Handschuh für Virtual-Reality Anwendungen ausprobiert?**

Ja  Nein

**Haben Sie schon einmal einen Laserpointer ausprobiert?**

Ja  Nein

**Haben Sie schon einmal mit einem Multi-Display Environment gearbeitet? (Dual-Desktop-Systeme ausgenommen)**

Ja  Nein

**Haben Sie schon einmal ein Smartphone mit Multi-Touch-Screen bedient.**

Ja  Nein

**Haben Sie schon einmal mit einem Multi-Touch Display gearbeitet?**

Ja  Nein

**Falls Ja: Ist Ihnen bekannt wie Objekte manipuliert werden können. Z.B. vergrößern eines Bildes?**

Ja  Nein

## A.4. Experiment: In-Test Questionnaire for Drag and Drop

### Interviewfragen (In-Test Questionnaire)

#### Direktes Pointing – Drag & Drop

	Trifft voll zu	Trifft teilweise zu	Weiß nicht	Trifft eher nicht zu	Trifft nicht zu
<b>Die Bedienung mit dem Handschuh...</b>					
...ist mir leicht gefallen	++	+		-	--
...war mental Anstrengend	++	+		-	--
...hat mir Spaß gemacht	++	+		-	--
...ermöglicht genaues arbeiten	++	+		-	--
...wurde mit der Zeit langweilig	++	+		-	--
...war körperlich Anstrengend	++	+		-	--
...funktioniert auch bei kleinen Objekten gut	++	+		-	--
...erforderte viel Konzentration	++	+		-	--
...fiel mir mit der Zeit immer leichter	++	+		-	--
...ermöglicht schnelles wechseln zwischen Displays	++	+		-	--
...empfand ich als ungenau	++	+		-	--

	Trifft voll zu	Trifft teilweise zu	Weiß nicht	Trifft eher nicht zu	Trifft nicht zu
<b>Ich hatte das Gefühl, dass...</b>					
...ich je länger je besser mit dem System interagieren konnte	++	+		-	--
...die Cursorposition meiner Intuition entsprachen	++	+		-	--
...das System verzögert reagierte	++	+		-	--
...ich die Objekte sehr genau positionieren konnte	++	+		-	--
...ich mich nur sehr langsam an das System gewöhnen konnte	++	+		-	--
...die Genauigkeit vom System nicht ausreichend ist	++	+		-	--
...die Cursor falsch positioniert waren	++	+		-	--
...der Wechsel zwischen den Displays umständlich war	++	+		-	--
...eine flüssige Interaktion möglich war	++	+		-	--
...es kaum ein Unterschied zwischen den Objektgrößen gab	++	+		-	--
...die Interaktion körperlich Anstrengend war	++	+		-	--

## A.5. Experiment: In-Test Questionnaire for Manipulation

### Interviewfragen (In-Test Questionnaire)

#### Direktes Pointing – Manipulation

	Trifft voll zu	Trifft teilweise zu	Weiß nicht	Trifft eher nicht zu	Trifft nicht zu
<b>Die Bedienung mit dem Handschuh...</b>					
...ist mir leicht gefallen	++	+		-	--
...war mental Anstrengend	++	+		-	--
...hat mir Spaß gemacht	++	+		-	--
...ermöglicht genaues arbeiten	++	+		-	--
...wurde mit der Zeit langweilig	++	+		-	--
...war körperlich Anstrengend	++	+		-	--
...funktioniert auch bei kleinen Objekten gut	++	+		-	--
...erforderte viel Konzentration	++	+		-	--
...fiel mir mit der Zeit immer leichter	++	+		-	--
...ermöglicht schnelle Manipulation von Objekten	++	+		-	--
...empfund ich als ungenau	++	+		-	--

	Trifft voll zu	Trifft teilweise zu	Weiß nicht	Trifft eher nicht zu	Trifft nicht zu
<b>Ich hatte das Gefühl, dass...</b>					
...ich je länger je besser mit dem System interagieren konnte	++	+		-	--
...die Cursorposition meiner Intuition entsprachen	++	+		-	--
...das System verzögert reagierte	++	+		-	--
...ich die Objekte sehr genau positionieren konnte	++	+		-	--
...ich mich nur sehr langsam an das System gewöhnen konnte	++	+		-	--
...die Genauigkeit vom System nicht ausreichend ist	++	+		-	--
...die Cursor falsch positioniert waren	++	+		-	--
...die Manipulation von Objekten umständlich war	++	+		-	--
...eine flüssige Interaktion möglich war	++	+		-	--
...es kaum ein Unterschied zwischen den Objektgrößen gab	++	+		-	--
...die Interaktion körperlich Anstrengend war	++	+		-	--

## A.6. Experiment: Post-Test Questionnaire

### Zur Evaluation (Post Questionnaire)

---

#### Welche Pointing-Technik...

##### ... war genauer

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... war schneller

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... war anstrengender zu benutzen (Ermüdung der Finger, Handgelenk, Arm, Schulter)

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... hat Dir mehr Spaß gemacht

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... würdest du lieber für eine solche Anwendung verwenden

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... erscheint dir Intuitiver

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... ist für Drag & Drop besser geeignet

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... ist für Manipulation besser geeignet

Direktes Pointing Hybrides Pointing Kein Unterschied 

##### ... ist für kleine Objekte besser geeignet

Direktes Pointing Hybrides Pointing Kein Unterschied