Abstract of “Principles and Applications of
Multi-touch Interaction” by Tomer Moscovich, Ph.D., Brown University, May 2007.

Many everyday activities rely on our hands’ ability to deftly control the physical attributes of objects. Most graphical interfaces only use the hand’s position as input, ignoring a rich array of other hand parameters such as orientation or finger configuration. This dissertation examines how multi-touch input lets us make better use of our manual dexterity.

We study the human factors governing multi-touch interaction, with special emphasis on finger coordination. Continuous multi-touch methods are emerging as a valuable form of high-degree-of-freedom input, yet few guidelines exist to aid the designer of multi-touch interfaces. The results of our studies reveal the roles played by the visual structure and the control structure of tasks in determining the effectiveness of mappings between users’ hands and fingers and the parameters they control in a software system. They increase our understanding of multi-touch interaction and its relationship to related methods such as bimanual interaction.

We also presents a number of novel interaction techniques that illustrate the benefits of multi-touch interaction. These techniques let users work faster and more fluently than do traditional single-point interaction methods. We describe a performance animation technique that lets users easily express ideas about motion and sequence while forgoing the extensive training and effort required for traditional animation. We also discuss a family of techniques that use multiple finger contacts to control digital instruments for manipulating graphical objects. These instruments allow for parallel control of multiple parameters (such as position and orientation) and reduce the need for separate interaction modes by unifying operations (such as grouping and moving objects).

Principles and Applications of Multi-touch Interaction

by
Tomer Moscovich
B. A., New York University., 1999
Sc. M., Brown University, 2001

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy in the
Department of Computer Science at Brown University

Providence, Rhode Island
May 2007

This dissertation by Tomer Moscovich is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date __________ ______________________________

John F. Hughes, Director

Recommended to the Graduate Council

Date __________ ______________________________

Andries van Dam, Reader

Date __________ ______________________________

Takeo Igarashi, Reader
(Tokyo University)

Approved by the Graduate Council

Date __________ ______________________________

Sheila Bonde
Dean of the Graduate School

Vita

The author was born in Jerusalem, Israel in 1977. After moving to the United States in 1987, he completed his secondary education in Tenafly, New Jersey. He continued his studies at New York University, in New York, New York, where he received a bachelor's degree in computer science and mathematics. The author then moved to Providence, Rhode Island in 1999 where he enrolled in the Ph.D. program at Brown University's department of Computer Science. His research at Brown focused on topics in computer graphics and human-computer interaction. The publication of this document represents the culmination of his stay at Brown University in December, 2006.

Acknowledgements

I want to thank the people who have guided me along the Ph.D. path, in particular, my advisor, John Hughes, for the support, advice, and example he provided since I first arrived at Brown. I also give thanks to the readers of this dissertation: To Takeo Igarashi for his invaluable help in providing the initial push that brought this work from an idea to a reality, and to Andy van Dam for keeping it on track. The tutelage of David Salesin and Dave Bargeron has also helped me greatly in accomplishing this research.

Many individuals here at Brown have provided me with encouragement, assistance, and enlightening discussion. I especially wish to thank Olga Karpenko, Peter Sibley, Liz Marai, Stefan Roth, Morgan McGuire, Dan Keefe, Yanif Ahmad, Joe LaViola, Cullen Jackson, Chad Jenkins, and Bob Zeleznik. I cannot imagine accomplishing this work without them. I also thank Jun Rekimoto and Kentaro Fukuchi for providing their unique touchpad.

Finally, I thank my parents and sister for their love and support, without which I would never have started this work in the first place.

Contents

List of Figures **ix**

1 Introduction **1**

1.1 Motivation . . . 2

1.2 Thesis statement . . . 5

1.3 Dissertation Overview . . . 5

1.4 Contributions . . . 6

1.5 Terminology . . . 6

2 Review of Literature and Technology **9**

2.1 Multi-touch and Whole-hand Input Technology . . . 10

2.1.1 Touchpads . . . 10

2.1.2 Vision-based Systems . . . 11

2.1.3 Whole Hand Input . . . 13

2.1.4 High Degree-of-freedom Input . . . 13

2.2 Multi-touch and Whole-hand Interaction Techniques . . . 14

2.2.1 Classification of Multi-touch and Whole-hand Interaction Techniques . . . 14

2.2.2 Hand Gesture Interfaces . . . 14

2.2.3 Bimanual Interaction . . . 16

2.2.4 Tangible Interfaces . . . 18

2.2.5 Continuous Multi-touch Interaction . . . 18

3 Case Studies **21**

3.1 Deformation and Animation . . . 21

3.1.1 Motivation 22
3.1.2 User Experience 23
3.1.3 Implementation Details 24
3.1.4 Related Work on Accessible Animation 26
3.1.5 Discussion and Future Work 27
3.2 Multi-finger Cursors 30
3.2.1 Motivation—Cursors as Tools 30
3.2.2 Design Principles 32
3.2.3 Techniques 32
3.2.4 A Relative Multi-point Touchpad 43
3.2.5 Evaluation 45
3.2.6 Discussion of Multi-finger Cursors 48
3.3 Discussion and Lessons Learned 51

4 Principles of Multi-touch Interaction 53
4.1 Introduction 53
4.2 Motivation 54
4.3 Parameter Mapping in Multi-finger and Bimanual Interaction 56
4.3.1 Multi-touch Interaction Using One and Two Hands 58
4.3.2 Measuring Coordination 60
4.4 Perception and Control of Multi-touch Input Using One and Two Hands 61
4.4.1 Experiment One 62
4.4.2 Experiment Two 68
4.4.3 Discussion and Future Work 73

5 Future Research Directions 75
5.1 Principles 75
5.1.1 Human Manipulation Abilities 75
5.1.2 Importance of Haptic Feedback in Multi-touch Interaction . . 77
5.2 Applications 78
5.2.1 3D Manipulation 78
5.2.2 Multi-touch Instruments 80

5.2.3 Other Applications and Interaction Techniques 81

6 Conclusions 83

6.1 Summary of Contributions 83

6.1.1 Human Abilities in Multi-touch Interaction 83

6.1.2 Multi-touch Interaction Techniques 84

6.2 Limitations and Open Issues 85

6.3 Closing Remarks 85

Bibliography 87

List of Figures

1.1 Many simple everyday tasks, such as tying one's shoes, require simultaneous coordinated control of multiple degrees-of-freedom. While most of us can perform this feat without even thinking, prevailing graphical interfaces do not make use of our manipulation abilities, relying on only a single point for input. 1

1.2 While our minds and bodies can control many streams of information, and computers can accept many streams of input, our interaction with the software must trickle through a single-point input device. 2

1.3 To set the leaf at the top of the branch using a standard drawing paradigm, the user selects the leaf and moves it near its destination. He then selects the scale widget and scales the leaf. Next, he selects the rotate widget and orients the leaf. Finally, he selects the leaf again and moves it to its goal position. This common operation requires four selections and three separate modes. 3

2.1 The SmartSkin touchpad returns an image of the distance from the touchpad to the hand. 11

2.2 (a) To draw an axis-aligned ellipse, a user specifies opposing corner points P and Q of its bounding rectangle. (b) When using one hand, a user must specify these points one at at time. However, determining the location of the first point is a challenging mental visualization task. For example, to tightly enclose a rectangle within an ellipse, the user must visualize the bounding rectangle and its associated ellipse to determine a valid position for the initial point. (c) Using two hands allows the user to externalize the visualization task by rapidly exploring the solution space. 16
2.3 Rekimoto *et al.*'s "potential field" interaction: Objects slide away from the hand on the surface. 19

3.1 Animation sequence of a crawling worm. The animator moves and bends the drawing with his fingers as though manipulating a real-world flexible object. 21
3.2 A user makes a whale open and close its mouth as it swims. 23
3.3 (a) Overview of our system setup. (b) Close-up of interaction area. . 25
3.4 Moving an internal constraint point instantly affects the entire shape. 25
3.5 The hand cursor lets the user move the puzzle pieces as though sliding objects on a physical table. 30
3.6 The hand cursor maps each finger contact on the touchpad (right) to a point on the screen (left). The position of the fingers is determined within a reference frame set by the position of the hand. Hand position is estimated as the mean finger position. The motion of the hand controls the relative position of the cursor on the screen, the extent and position of which are indicated by an enclosing circle. 33
3.7 (Left) Controlling many parameters at once is useful for performance animation. Here the user animates two swimming fish by using two hand cursors. (Right) Using several fingers allows users to control objects with many degrees of freedom, like the control points of this subdivision curve. 34

3.8 A user simultaneously translates, rotates, and scales a leaf using the similarity cursor. Parallel control of an object's properties allows for more fluid interaction, and may decrease task completion time. . . . 37
3.9 The similarity cursor is controlled using two fingers on the touchpad. Change in the position of their centroid controls the relative position of the cursor. Change in the slope of the line they define controls the relative orientation of the cursor. The changing distance between the points adjusts the size of the object manipulated by the cursor, and is indicated by animated stripes. 38
3.10 (Left) The large activation area of the adjustable area cursor reduces the time and precision needed to acquire isolated targets. (Right) The selection of fixed-radius area cursors is ambiguous in crowded regions. This ambiguity is resolved with the adjustable area cursor by minimizing the activation area. 40
3.11 (Left) Traditional area cursors make it impossible to click on empty space near selectable objects. (Center) The adjustable area cursor can emulate a point cursor without requiring the user to switch modes. (Right) Since the adjustable area cursor does not suffer from the ambiguity of fixed-area cursors, it can be used to control groups of nearby objects. 42
3.12 The motion of the centroid of the fingers on the touchpad controls the relative position of the adjustable area cursor. The greatest distance between contact points is mapped to the radius of the cursor. 43
3.13 Our prototype touchpad uses a tactile switch to allow users to distinguish tracking from dragging motion. 44
3.14 Our prototype illustration system uses the similarity cursor and the adjustable area cursor for shape and curve manipulation. Drawing is provided by courtesy of one of our artist participants. 46
3.15 More cursor possibilities: A shaped area cursor for more precise selections (left) and a parametrized 2D sculpting tool (right). 50

4.1 Visual display and mappings for the tracking task. Users were asked to track a moving segment using two fingers on one and two hands. For each hand condition, the segment was manipulated using both an aligned display (left) and a rotated display (right). 62
4.2 Results for Experiment One. Note that rotating the visual stimulus resulted in a large increase in error in the bimanual condition, but only a minor increase in the unimanual condition. 66
4.3 Visual display and mappings for the alignment task. Subjects were asked to align a control-shape to a congruent target-shape using two fingers on one and two hands. For each hand condition users manipulated both a round, featureless shape (left) and a thin, pointed shape who's key features were aligned with the subject's fingers (right). . . 68
4.4 Results for Experiment Two. Subjects aligned the pointed shape slightly faster using two hands than one. This result is explained by the greater efficiency of bimanual control of two separate points than unimanual control. 71

Chapter 1

Introduction

Figure 1.1: Many simple everyday tasks, such as tying one's shoes, require simultaneous coordinated control of multiple degrees-of-freedom. While most of us can perform this feat without even thinking, prevailing graphical interfaces do not make use of our manipulation abilities, relying on only a single point for input.

Our everyday interaction with the world is complex, fluid, and often transparent. But when we interact with modern graphical interfaces, we are stripped of our dexterity, and are left poking clumsily at the digital world with the single index finger of the mouse pointer. Single-point interaction can be difficult—imagine tying your shoes with only one finger. Using two hands improves the quality of interaction. But even with one finger on each hand most of us would be hard-pressed to get dressed in the morning. With the use of a finger and thumb on one hand, our manipulative ability skyrockets. The goal of our research is to make interaction with a computer more facile and satisfying by using abilities of the hand beyond simple position control.

While multi-finger interaction is a common experience for most touch-typists, typing, clicking a mouse, and executing keyboard shortcuts (*e.g.,* *control-C* for copy) are discrete, serial actions. This dissertation will focus on interactions that are continuous and coordinated, like the movements of an artist controlling a paintbrush. Multi-point touchpads make this type of interaction possible. These touchpads detect every point where their surface is in contact with the user's hands or fingers. Thus, they can be used to control many more parameters than traditional pointing devices.

1.1 Motivation

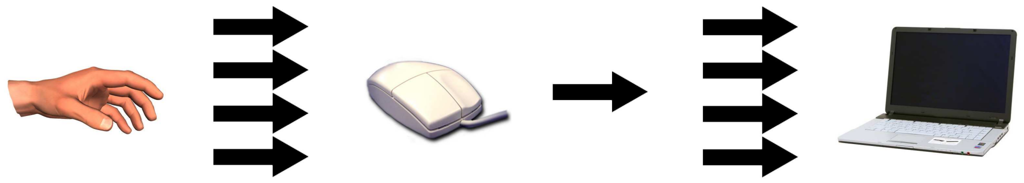

Figure 1.2: While our minds and bodies can control many streams of information, and computers can accept many streams of input, our interaction with the software must trickle through a single-point input device.

Real-world tasks often entail specifying multiple parameters.[*] For example, when setting a table one must control at least two spatial dimensions and one orientation for each utensil. Similarly, multi-parameter setting is also implicit in many tasks performed on a computer. Common examples include window manipulation, map

[*]In this dissertation, we use the term *parameter* to refer to measurable properties that a user may want to control. These may be properties of digital or physical objects (*e.g.,* object orientation) or properties of the user's body (*e.g.,* distance between finger and thumb).

navigation, control of multiple objects, image editing operations such as cropping or tone mapping, and drawing tasks such as curve editing, color selection, or object layout and scaling. But adjusting more than two parameters on a computer is often complicated. Let us take, as an example, the task of positioning, orienting, and scaling an object in a drawing program (see figure 1.3). Standard techniques found in programs such as Microsoft PowerPoint and Adobe Illustrator break up the control of these parameters into a sequence of steps. Graphical widgets represent separate modes, each controlling a single software parameter (like object size) or a pair of related parameters (like X and Y positions). However, as anyone who has used a measuring tape knows, in the real world one can position, orient, and stretch an object in a single fluid motion. The complexity of the software technique derives from the fact that input devices like the mouse can only control two parameters at once. While our minds and bodies can control many streams of information, and computers can accept many streams of input, our interaction with the software must trickle through a single-point input device.

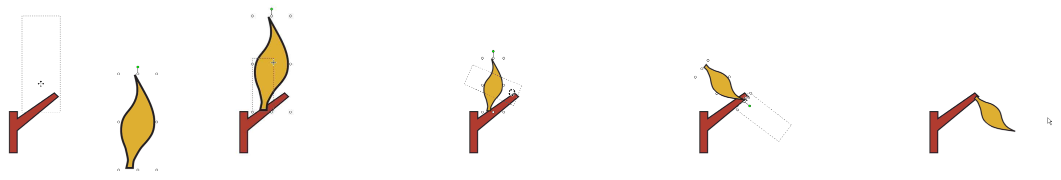
Figure 1.3: To set the leaf at the top of the branch using a standard drawing paradigm, the user selects the leaf and moves it near its destination. He then selects the scale widget and scales the leaf. Next, he selects the rotate widget and orients the leaf. Finally, he selects the leaf again and moves it to its goal position. This common operation requires four selections and three separate modes.

Multi-point input allows for more parallel interaction, thus reducing the necessary complexity of the UI. This parallelism can lead to faster performance, because sub-task execution can overlap in time. Larger sub-tasks may improve expert skill acquisition through *chunking* [98], as experts become adept at performing increasingly large sub-tasks [28]. Parallelism has other cognitive benefits [78]. For example, to draw an axis-aligned ellipse a user must specify the top-left and bottom-right points of of its bounding rectangle. When working sequentially, it can be difficult to determine where to place the first point, as the user must mentally visualize the eventual position of the ellipse. When working in parallel the user can simply monitor the rendered ellipse while adjusting its bounding rectangle.

While there are many potential ways of providing high-degree-of-freedom input, there are several reasons to believe that using the fingers for input would be most effective. As noted by Card, Mackinlay, and Robertson [29] the muscles groups in the fingers are controlled by a disproportionally large area of the motor cortex. A large control area for a muscle group may be correlated with high bandwidth. This idea is supported by the experimental results of Zhai *et al.* [130], which indicate that better performance is achieved by assigning a high-degree-of-freedom task to the small muscle groups of the fingers rather than to the muscle groups of the wrist and arm. Studies by Balakrishnan and MacKenzie [14] conclude that this effect does not result from a higher bandwidth in the muscle groups of individual fingers, but rather from the cooperative interaction of several fingers working in concert.

One way to acquire input from several fingers is to record the points where they touch the surface of a digitizing tablet or touchscreen. Several technologies now exist for creating multi-point touchpads (see section 2.1). These touchpads make for excellent general-purpose input devices, as they are well-suited for a wide variety of input tasks. Since multi-point touchpads can also act as a single-point touchpads they are easy to integrate into prevailing GUI platforms. Thus users gain the benefits of multi-point input while maintaining full compatibility with mature, standardized one-point interaction techniques. A touchpad can also be specialized for specific interaction tasks by placing overlays or stencils over its surface. These overlays divide the surface into regions to emulate control-panels composed of multiple input devices [26]. For example, a touchpad can serve as an array of sliders [21] or a set of programmable buttons. In a similar fashion, a touchpad can be used for text-input by emulating a soft-keyboard [39]. Multi-point touchpads also provide good support for gestural interfaces, as they allow for whole-hand and multi-finger gestures that closely resemble real-world manipulative gestures [39, 93, 122]. Finally, multi-point touchpads support bimanual interaction techniques. Two-handed interaction has been studied extensively, and is well understood [12, 53, 59]. Researchers have developed many bimanual interaction techniques that outperform their unimanual counterparts and reduce task complexity (see section 2.2.3).

One use of multi-point touchpads that has not been well studied previously is continuous input of multiple parameters. As discussed above, this type of input is highly desirable. A few initial explorations [85, 93], including some discussed here, show this to be a promising research direction. However, it is difficult to draw general conclusions from these experiments that can be applied to future development of interaction techniques. How can we guide interface designers to the most promising areas of a vast design space? This dissertation works to improve our understanding of the human factors involved in continuous multi-touch interaction, and the design implications thereof.

1.2 Thesis statement

This dissertation works to support the following thesis:

In a wide variety of tasks, continuous graphical interaction using several fingers allows users to communicate information to a computer faster and more fluently than single-point graphical interaction techniques.

1.3 Dissertation Overview

The goal of this dissertation is to show how interaction techniques that use multi-point touchpads can improve the quality of human-computer interaction through coordinated high-DOF input. We begin by discussing other attempts at improving interaction by using high-DOF input (Chapter 2). This includes development of high-DOF input technology such as touchpads, vision tracking, and instrumented gloves, as well as techniques such as bimanual interaction, gestural interaction, and tangible interaction. We describe a series of novel multi-touch interaction ideas, and show specific examples of such techniques (Chapter 3). These include a system for performance animation, and a family of techniques which illustrate how to create effective high-degree-of-freedom tools. To help interface designers develop future multi-touch interaction techniques, we study key human factors involved in multi-point input. The results of our investigation increase the understanding of such methods, and allow us to formulate guidelines for their design (Chapter 4). The contributions of this work are discussed in Chapter 6.

1.4 Contributions

This dissertation contributes to the field of human computer interaction in several ways. It establishes continuous multi-touch interaction as a valuable method of high-degree-of-freedom input. The techniques described in this work act as landmarks in the design space of multi-touch interaction. The techniques themselves offer more fluid graphical interaction in multiple domains. They allow novices to easily create simple, yet expressive, animations, a difficult task using traditional methods. They simplify graphical manipulations tasks, such as object translation and orientation, by unifying the control of related parameters, and they enhance the selection and grouping of multiple objects.

This work also increases our understanding of interaction using multiple fingers on one hand, and how it compares to related methods such as bimanual input. Our experiments show that people can effectively coordinate the motion of multiple fingers to accomplish common graphical interaction tasks. They also assess the role of stimulus-response compatibility and of the separability of input dimensions in creating effective mappings between fingers and software parameters. This knowledge allows us to formulate a set of design-principles for multi-touch interaction techniques which will guide the development of future interfaces.

1.5 Terminology

Bimanual interaction technique Interaction technique involving the use of both of a user's hands.

Control structure Organization of controllable parameters in an input device or a user's arms, hands, and fingers. See Section 4.3.

Coordination Control of two or more parameters simultaneously so as to attain a goal more effectively than by independent control.

Gesture/hand-gesture A hand posture possibly combined with hand motion used to initiate a command and optionally command parameters to a software system.

High-degree-of-freedom, high-DOF Involving more than two parameters.

Instrument A mechanical or *logical* device which transforms human action so as to make a task easier to perform or to enhance human performance of a task.

Integral control structure Structure of input parameters which makes it easy to control one without affecting the others. An Etch-A-Sketch exhibits such a structure. (Contrast with separable control structure.)

Multi-touch interaction technique Unless otherwise stated, this term refers to methods that use multiple contact points on a multi-point touchpad for *continuous* control of multiple parameters in a software system.

Multi-touch, multi-point, multi-finger Relating to the use of multiple finger contacts on a multi-point touchpad.

Parallelism Simultaneous control of two or more parameters.

Parameter We use the term *parameter* to refer to measurable properties that a user may want to control. These may be properties of digital or physical objects (*e.g.,* object orientation) or properties of the user's body (*e.g.,* distance between finger and thumb).

Posture/hand-posture Static configuration of the hand, generally used to signal a command to a software system. See *gesture*.

Separable control structure Structure of input parameters which makes it easy to control them together in a coordinated manner. A computer mouse exhibits such a structure. (Contrast with integral control structure.)

Touch-pad, touch-surface, interactive surface Any surface that is capable of reporting information regarding hand contact with itself. Information may range from a binary touch/no-touch, to a proximity map or traction field.

Whole-hand interaction Interaction methods that use more parameters of the hand than its position. Examples include hand posture, finger configuration, joint angles, etc.

Chapter 2

Review of Literature and Technology

Many efforts have been made at harnessing our hands' ability to manipulate real-world objects to control digital objects. The results of these efforts have yielded devices such as touchpads and instrumented gloves for measuring the shape and pose of our hands, as well as many techniques for using these parameters. However, only a few methods have addressed the focus of this dissertation: continuous multi-parameter control (see section 2.2.5). The majority of techniques that rely on direct measurement of hand parameters use a symbolic, "gestural," interaction style, in which additional hand measurements are used to specify commands or interaction modes rather than for parallel input. Meanwhile, most high-DOF input techniques rely on special-purpous hardware for measuring specific sets of parameters (*e.g.*, rigid-body position and orientation) rather than on direct hand measurements. The one major exception to this state of affairs is bimanual interaction, where direct measurements of the motion of the user's hands are used for parallel high-DOF control. Continuous multi-touch interaction can be seen as a generalization of the bimanual interaction style. It inherits the benefits of two-handed interaction, while offering the possibility of even more parallelism, and a greatly expanded design space. This chapter places the emerging field of continuous multi-touch interaction within the context of earlier high-DOF and whole-hand interaction techniques.

2.1 Multi-touch and Whole-hand Input Technology

2.1.1 Touchpads

Touch tablets that can sense more than a single point of contact were first proposed by Lee, Buxton, and Smith in 1985 [77]. Their digitizer is composed of an array of capacitive proximity sensors where the finger and sensor act as two plates of a capacitor. Since capacitance is inversely proportional to the distance between the plates, robust contact detection can be accomplished by simply selecting an appropriate threshold. The resolution of the digitizer can be enhanced beyond the physical resolution of the sensor matrix by interpolating data from neighboring groups of sensors. The touchpad could also approximate pressure sensing by monitoring the increase in capacitance as the fingertip flattens against its surfaces.

Another touch-surface based on capacitive coupling is Dietz and Leigh's DiamondTouch system [35]. The digitizer is composed of a grid of row and column antennas which capacitively couple with users when they touch the surface. Users in turn, are capacitively coupled through their chairs to a receiver. By driving each antenna with a unique signal, the system can tell which antenna is being touched by which user. A key advantage of this technique over other methods, is that it can identify which user is touching the surface. The DiamondTouch system uses time-division multiplexing to cycle through the row and column antennas. This scheme only yields the margins of the capacitively coupled area, which limits the system's ability to identify multiple points of contact. A user touching the surface at two points can produce two possible interpretations, and so the system is limited to providing an axis-aligned bounding rectangles of the area touched by each user.

Rekimoto's SmartSkin [93] can provide an image of a hand's proximity to each point on its surface (see figure 2.1). The digitizer consists of a grid of capacitively coupled transmitting and receiving antennas. As a finger approaches an intersection point, the strength of the signal drops. By measuring this drop, the system can determine how close a finger is to the receiving antenna. Through time-division multiplexing the transmitting antenna is identified as well. By thresholding the proximity

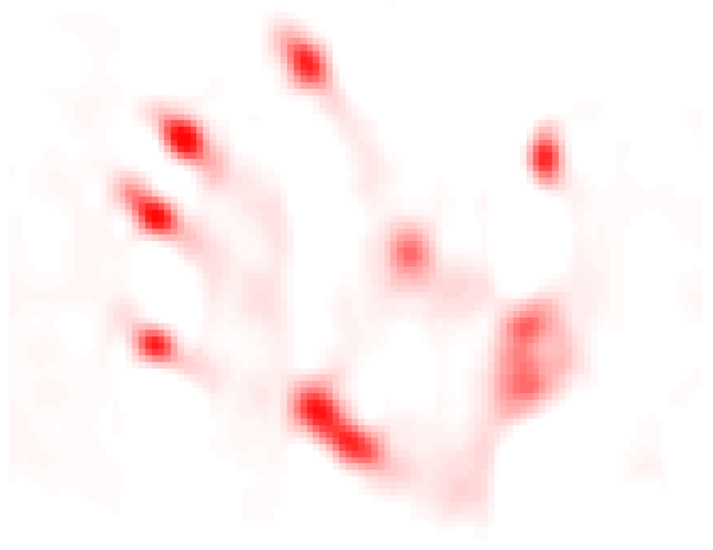

Figure 2.1: The SmartSkin touchpad returns an image of the distance from the touchpad to the hand.

map, multiple points and complex contact regions can be identified.

Several multi-point touchpads have been produced commercially, although their mechanisms of operation are not always known. The TouchStream and iGesture touchpads by FingerWorks [39] appear to be an array of capacitive sensors, and can report the position, contact area, and eccentricity of multiple fingers. They are likely to be decedent from the multi-touch surface described by Westerman [115]. The Tactex MTC Express uses a series of fiber-optic strain gages to measure pressure on multiple points of its surface. Other multi-point touchpads which appear to respond to pressure are the Tactiva TactaPad [101], and JazzMutant's Lemur.

2.1.2 Vision-based Systems

Vision based systems can be roughly classified as “direct” systems, where cameras are aimed directly at the users hands, and “indirect” systems, where the cameras are aimed at a touch-surface that produces a change in the image when touched by a finger or object. As the body of research on direct vision systems for hand and finger tracking is very large, we only describe a few representative systems. For a more complete review of the literature see [32, 79].

One of the earliest direct vision system for whole-hand interaction is Krueger's VIDEOPLACE [69]. The system captures an image of the user, who stands in front of a plain background. It segments the image, and displays it as a silhouette in real-time. The moving silhouette can then be used to interact with digital objects and animated characters. Wellner's DigitalDesk system [114] segments an image of a pointing finger against a cluttered background by calculating a difference image between two consecutive frames. Contact with the desk is determined by using a microphone to listen to the sound of a finger tapping the desk. The Visual Touchpad system of Malik and Laszlo [85] uses the disparity between the images of two cameras to determine the height of fingers above the touchpad. The system reports that a finger is in contact with the touchpad if it is bellow a threshold height above the touchpad. By tracking the position of the hand, the Visual Touchpad can make an informed guess as to the identity of each finger, and also calculate its orientation.

Rekimoto and Matsushita's HoloWall is a typical example of an "indirect" vision system. An infrared illuminant and a camera equipped with an infrared filter are placed behind a diffusive projection panel. As objects begin to approach the panel they are faintly lit by the infrared light. The illumination rises dramatically when object touch the diffusive panel, which allows the system to unambiguously determine the contact areas by simply thresholding the infrared image. A similar system by Han [56] uses frustrated total internal reflection to highlight the touched area.

In contrast to most touch-systems, which can only determine the position, shape, or area of contact, the GelForce system of Vlack *et al.* [108] measures the traction force applied to the touchpad. The system tracks a dense array of colored markers embedded in a block of clear silicone to determine the traction field. Since the system detects deformation due to both pressure on the surface as well as lateral forces, it can be used for both isotonic and isometric input. This ability makes the device appropriate for both position and rate control.

A different approach to vision-based touchpads is described by Wilson [117, 119], who instead of relying tracking finger positions for input calculates an optical flow-field. This technique uses the motion of the entire hand to move and rotate objects.

2.1.3 Whole Hand Input

A large body of work has been dedicated to accurate estimation of whole-hand posture and movement. This has generally been done either through vision-based methods, or by instrumenting the hand with sensors to measure a large number of hand parameters (*e.g.,* extension and adduction of all fingers). While the stated goal of many of these system is to allow users to interact with the digital world using their real-world manipulation skills, successful vision and glove-based interaction systems have been primarily limited to gesture-based interaction (see below). For a detailed survey of the literature see LaViola [75] and Struman [99].

2.1.4 High Degree-of-freedom Input

Several general purpose input devices attempt to make use of our ability to manipulate the many degrees of freedom of physical objects. A large number of these devices are 6 DOF controllers designed to control the three spatial and three angular degrees of freedom of a rigid body in space. These devices can be categorized as free moving *isotonic* controllers such as the "Bat" by Ware [112] and the Fingerball of Zhai *et al.* [130] that allow for isomorphic position and orientation control, and *isometric* controllers such as the Spaceball [1] and Space Mouse [2] that use the forces applied by a user to a static object to control the rate of change of parameters. In between these extremes are *elastic* controllers such as the "poor man's force-feedback unit" of Galyean and Hughes [45] and the EGG of Zhai [127]. These devices provide more kinesthetic feedback than *isometric* devices, but retain the self-centering property needed for rate control.

Hybrid devices also exist. For example, the GlobeFish [44] is an elastic position controller coupled to an isotonic orientation controller. The rockin' mouse [9]is a different sort of hybrid; it uses two spatial dimensions and one angular dimension to control position in three dimensions.

Specialized multi DOF devices have been created for a variety of applications. Motion capture systems are one example [22], as are instrumented armatures [38]. ShapeTape [64] is a length of rubber tape instrumented with 32 bend and twist sensors.

2.2 Multi-touch and Whole-hand Interaction Techniques

2.2.1 Classification of Multi-touch and Whole-hand Interaction Techniques

Struman [99] describes a taxonomy of whole-hand interaction techniques which divides them into a class of discrete and a class of continuous input methods. Within each class a technique is described as a direct interaction, a mapped interaction, or a symbolic interaction. Zhai and Milgram [129] point out that interaction methods form a continuum ranging from the direct to the indirect. Direct interaction methods are based on an isomorphism* between the control space and the display space, while indirect methods or "tools" rely on more complex mappings. In the light of Struman's classification, this continuum can be further extended from isomorphism to tool to symbol. Using a touchscreen to select an object would lie on the isomorphism end of the continuum, while using a gesture to execute a command would lie on the symbol end. Many of the techniques described below are hybrid methods that rely on a gesture or hand posture as a symbolic mode-switch that determines the interpretation of a subsequent mapped or direct manipulation.

2.2.2 Hand Gesture Interfaces

Much research on whole hand interaction techniques has examined the concept of gestures. Hand gestures in the context of HCI are much like hand gestures in everyday communications. They are hand postures and movements that express an idea. These may be simple, iconic symbols used to invoke a command, or, like a pointing index finger, they may also serve to indicate the parameters of an operation [68]. While such gestures may allow a user to select which of several parameters to adjust, the additional degrees-of-freedom are generally used for specifying the gesture, and not for coordinated high DOF control.

*This term is used somewhat loosely in the literature, generally as a reference to an isometry or similarity transformation.

An early example of whole hand gestures is the Charade system [15] which recognizes hand postures as commands to slide a presentation system. This type of gestural interaction can be thought of as simply an implement-free instance of keyboard command-shortcuts, and is found in many gesture-based systems. For example, Wu and Balakrishnan's RoomPlanner application [122] uses a tapping gesture to invoke menus, and a horizontal hand gesture to brink up a private display area. However, the system also uses compound gestures in which a hand posture can be followed by motion to adjust a parameter. Placing two fingers on the touch-surface initiates rotation, a flat hand gesture pans the work area, while a vertical hand gesture lets users sweep items along the table. In a similar vein, Malik *et al.* [84] describes a set of hand gestures for panning, resizing, and zooming a 2D workspace. The posture of the hand is used to select one of several system parameters which can be mapped to continuous parameters of the hand. For example, a two finger touch initiates a mapping from the inter-finger distance to the zoom-scale of the workspace. Gestures do not have to be restricted to one person—Morris *et al.* extends the concept to cooperative multi-user gestures.

A set of 3D multi-finger hand gestures is introduced by Grossman *et al.* [51] for object manipulation on a volumetric display. For example, a thumb "trigger" gesture is used to select an object, and a pinch gesture is used to move it. Vogel *et al.* [109] makes use of 3D hand gestures for pointing on very large displays.

Studies and observation of the usability of the above systems reveal that a well-designed gesture set can lead to fast, fluid interaction in settings, such as table-top collaboration, which are not well served by traditional mouse and keyboard methods. However, gesture-based systems are difficult to design and extend. Gestures must be carefully designed so as to be easy to learn, easy to differentiate from one another, and to accept parameters in a meaningful manner. Adding a single gesture to such a carefully designed system may invalidate the entire design. The design is often *ad hoc*, and few guidelines regarding gesture design and mapping assignment exist. Wu *et al.* [121] offer some thoughts on how to design usable systems through gesture reuse.

2.2.3 Bimanual Interaction

Two-handed interaction techniques have much in common with multi-touch interfaces, as both attempt to increase parallelism in continuous parameter input by measuring multiple hand parameters. A 1986 study by Buxton and Myers [24] reveals that parallel two-handed continuous input can reduce task completion time for scrolling and graphical manipulation tasks. Numerous studies have since increased our understanding of bimanual interaction, and many techniques have been proposed for making use of our two-handed interaction abilities.

Depending on the task, bimanual interaction may have several advantages over unimanual techniques. The most obvious advantage is parallelism. If users can successfully control parameters using two hands simultaneously, they can accomplish a multi-parameter manipulation task in less time. However, some researchers have found that the benefits of bimanual interaction are not limited to mechanical efficiency, but that using two hands changes the way users think about a task [59]. An experiment by Leganchuk [78] provides a good example of this. Participants were given a figure enclosing task which required significant visual planning to accomplish with one hand (see figure 2.2). The performance advantage of the bimanual condition over the unimanual condition increased with the difficulty of planning required to accomplish the task.

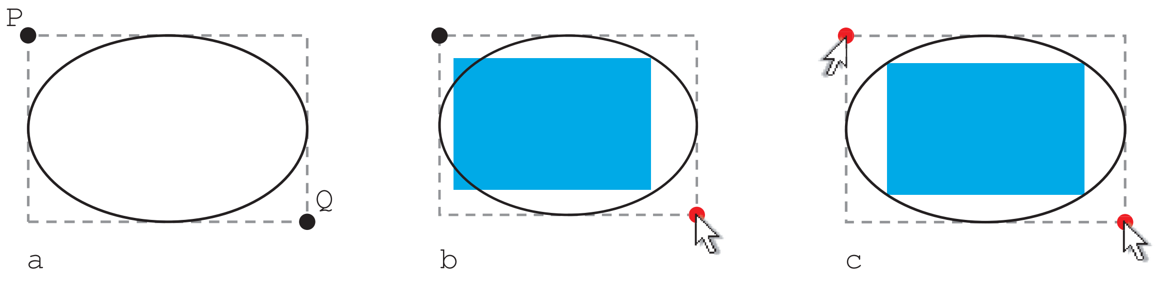

Figure 2.2: (a) To draw an axis-aligned ellipse, a user specifies opposing corner points P and Q of its bounding rectangle. (b) When using one hand, a user must specify these points one at at time. However, determining the location of the first point is a challenging mental visualization task. For example, to tightly enclose a rectangle within an ellipse, the user must visualize the bounding rectangle and its associated ellipse to determine a valid position for the initial point. (c) Using two hands allows the user to externalize the visualization task by rapidly exploring the solution space.

Bimanual interaction methods can be categorized as techniques where the hands are used symmetrically, such as steering a bicycle, and techniques where they are used asymmetrically, such as peeling a potato. Guiard puts forward an influential model of cooperative, asymmetric bimanual interaction [53] which attempts to explain the advantage of manual specialization. According to the model, the hands are coupled through the arms and body to form a kinematic chain, with the non-dominant hand as the base link. The model predicts many properties observed in asymmetric bimanual interaction. The first, is that the non-dominant hand serves to set a dynamic reference frame for the dominant hand's operation. Handwriting, where the non-dominant hand keeps the paper in the dominant hand's most effective work-area is a good example of this. The second, is a scale differences in motion where the dominant hand acts on a finer scale both spatially and temporally than the non-dominant hand. The third is non-dominant hand precedence in action, as dominant hand action is not sensible before its reference frame is set. Hinckley *et al.* [59] confirms the reference frame roles of the hands in cooperative action. The model is widely used as a guideline for designing bimanual interaction (for example, Kurtenbach *et al.*'s T3 system [70]), and also explains why the benefits of two handed interactions do not extend to task that fail to meet Guiard's description. For example, a study by Dillon *et al.* [36] found only a nominal benefit in using two mice for distinct tasks.

In contrast to asymmetric interaction, where the hands play different but complementary roles, in symmetric bimanual interaction both hands serve the same manipulative function. Experiments by Balakrishnan *et al.* [12] indicate that symmetric bimanual interaction requires an integrated task with a single focus of attention to be successful (in terms of low error and high parallelism). Latulipe *et al.* have shown that symmetric mappings can be more effective than asymmetric mappings for certain tasks [72–74].

Researchers and designers have developed a large number of bimanual interaction techniques. For example, the toolglass and magic lenses techniques let users click through a pallet held in the non-dominant hand [18, 70]. 2D navigation methods take advantage of two hands for unified zooming and panning [57]. Various techniques for figure drawing [70, 71] and curve editing [10, 74, 91] have also been proposed. Since 3D navigation and manipulation tasks require the user to control a large number of parameters, bimanual interaction methods seem to be a promising solution. Techniques have been devised for object visualization and manipulation [33, 58] as well as for camera control and navigation [13,125,126]. More bimanual techniques are described in sections 2.2.4 and 2.2.5 in the context of tangible and multi-touch interaction.

2.2.4 Tangible Interfaces

Another way of using multiple fingers for input is to manipulate physical tools and props who's properties (*e.g.,* orientation) are mapped to parameters of the software. The idea, known as tangible or graspable interface, is to make use of our natural prehensile abilities and the affordances provided by physical objects. Fitzmaurice *et al.* point out some advantages of graspable UIs [41]. They include parallel and bimanual input, spatial multiplexing of input rather than temporal multiplexing, support for collaboration, and making use of our spatial reasoning and manipulation skills.

To illustrate this concept Fitzmaurice *et al.* introduce "bricks," tracked physical blocks that serve as handles to digital objects. Users can associate a brick with a digital object by placing it on its display image. Moving the brick produces a corresponding movement in the object. By attaching bricks to the control points of an object (*e.g.,* a spline curve) users can perform more complex manipulations. The metaDESK of Ullmer and Ishii extends this idea by creating physical icons that serve as specialized handles and tools who's physical constraints translate to digital constraints in the software. Hinkley's system for neurosurgical visualization used tracked physical props including a head model and a rectangular plate. These props serve to do more than provide 6 DOF input—their shape gives the user a tangible cue as to the state of the system.

2.2.5 Continuous Multi-touch Interaction

This dissertation is particularly concerned with continuous, coordinated multi-touch control of multiple parameters. Several systems show examples of this type of control. Most of the techniques fall into one of three categories: 1D valuators or sliders, object transport and scaling, and specification of rectangles.

Buxton first introduced the idea of partitioning a multi-touch digitizer into strips to emulate a bank of sliders such as those found in audio mixers and studio light control panels [27]. A similar technique is described by Blaskó and Feiner, although in their system multiple contacts are used to increase the effective number of strips rather than for parallel control [21]. Benko *et al.* [17] take a different tack, and use the distance between two contact points to adjust the control-display ratio between a touchscreen and a cursor. A similar idea is used by Morris *et al.* [87] where the distance between one user's fingers on a table controls the width of a stroke drawn by another user.

Rekimoto [93] first introduced a technique for using two or more fingers to simultaneously translate, rotate, and scale a 2D object. The system finds a similarity transformation that is most similar, in a least-squares sense, to the transformation of the fingers, and applies it to the selected item. A similar two-finger technique is used by Wu [122]. Wilson [118] takes a related approach, by finding a rigid transformation that matches the optical flow of the user's hand. Malik *et al.* [85] take a slightly different approach to translating and rotating items, by measuring the change in a single finger's position and orientation, and applying it to the object.

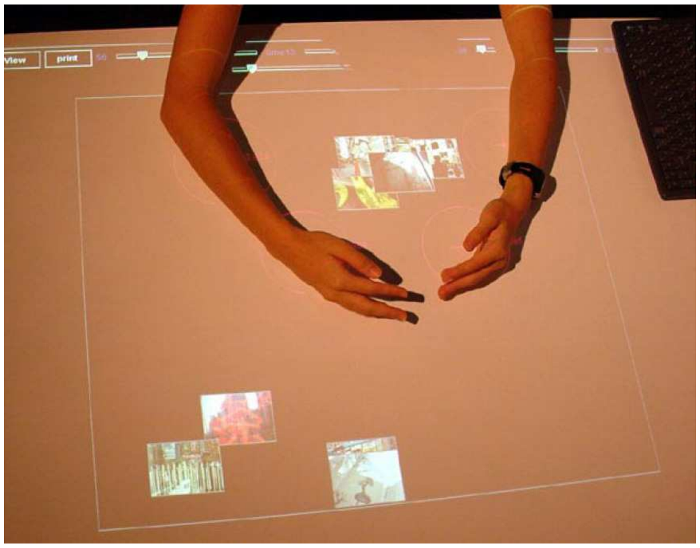

Figure 2.3: Rekimoto *et al.*'s "potential field" interaction: Objects slide away from the hand on the surface.

Dietz and Leigh [35] show how two contact points can determine an axis aligned rectangle. This technique was later used by Forlines and Shen [42] to specify regions-of-interest for fish-eye-lenses. Benko *et al.* [17] uses a similar technique for zoom-pointing. One finger is used to specify the initial center of a rectangle to magnify, while the other stretches the rectangle. After the center has been specified, the first finger is used to precisely point at a small target.

A few multi-touch techniques do not fall into the above classes. Rekimoto [93] shows a curve manipulation technique where four finger contact points are mapped to the control-points of a Bézier curve. The work also shows an example of "potential field" manipulation, where objects slide down the gradient of a distance field from the touch-surface to the hand (see figure 2.3). A different type of interaction is described by Malik *et al.* [84] where one hand positions the works space of the other to access a large display.

Chapter 3

Case Studies

In this chapter we discuss several novel interaction techniques that illustrate the expressive power of multi-touch interaction. Some of these techniques show how multi-touch methods simplify tasks that are currently controlled using a single point. For example, the similarity cursor eliminates the need for separate translation, rotation, and scaling modes. Other techniques, like multi-touch performance animation, let users do things they could not have done before.

These techniques also serve to ground future research into multi-touch methods by showing the kinds of tasks that can benefit from this type of interaction. They point out new directions to explore, while serving as guideposts in a vast design-space. The insight gained by observing the strengths and weaknesses of these techniques helps us formulate design recommendations, and increases our understanding of multi-touch manipulation.

3.1 Deformation and Animation

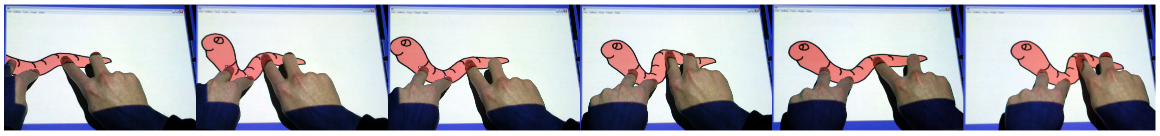

Figure 3.1: Animation sequence of a crawling worm. The animator moves and bends the drawing with his fingers as though manipulating a real-world flexible object.

3.1.1 Motivation

How many words express the meaning of a smile? We spend much of our waking hours communicating with others using our words, our voices, our facial expressions, and our bodies. Some ideas are most easily expressed with words, while others are better suited for pictorial expression. Ideas about time or sequence are often easiest to explain using motion. In face-to-face conversation we commonly use gestures and props to express such ideas. A dinner roll embodies the car that swerved on the highway. A key-chain illustrates the finer points of a basketball play. Unfortunately, our digital communication are commonly limited to static text and images, as animated explanations are difficult to produce. The difficulty lies in the large number of parameters required to describe an animation.

To produce computer animations today, animators must specify a large number of *keyframes*. Each frame is a slightly modified instance of the previous frame. Animation software interpolates the motion to fill in the in-between frames and produce a smooth animation. Animating in this manner is a tedious process that takes years to master [103]. The transformation between time and frames is especially difficult to learn.

While the traditional process can produce very high quality animation, expository animation can be effective even at a very low fidelity. We use this fact, along with two observations, to make 2D animation accessible to novice users, and to simplify the task for professional animators. The first observation, noted by Hämäläinen *et al.* [55] is that given easily deformable characters made of clay, and a simple stop-motion animation system, novice users were able to produce rich expressive animations with no instruction or training. Using clay characters makes it very easy to transform one frame into the next. Animators can use their real-world manipulation skills to bend the clay in their hands, and control many parameters at once. In contrast, traditional animators must completely redraw the character in each frame, while computer animators can only control one or two parameters at a time. Claymation created by novices, however, exhibits odd timing artifacts due to its creators' difficulty in transforming their conception of time into a sequence of frames. This brings us to the second observation: In real-world communication using gestures or props, people rely on their natural sense of timing to perform explanations of temporal ideas. A number of performance animation techniques have been proposed as means of simplifying the animation process [8, 37, 88], but as they are generally limited to controlling a single point at a time, the user must record and synchronize several layers of motion in order to control multiple animation parameters. We introduce a performance animation technique that allows users to bend and manipulate drawings as though they were physical rubber props. By using several fingers to control characters with many degrees-of-freedom, novice animators can create lively and expressive animations (see figures 3.1 and 3.2).

Our deformation method has uses beyond simple performance animation. Animators today often give a plain appearance to their characters because redrawing elaborate surface decoration, such as stripes on a tiger, is very time consuming [103]. Since our method produces plausible deformations for the interior of shapes, it can significantly reduce the need to redraw decorations.

3.1.2 User Experience

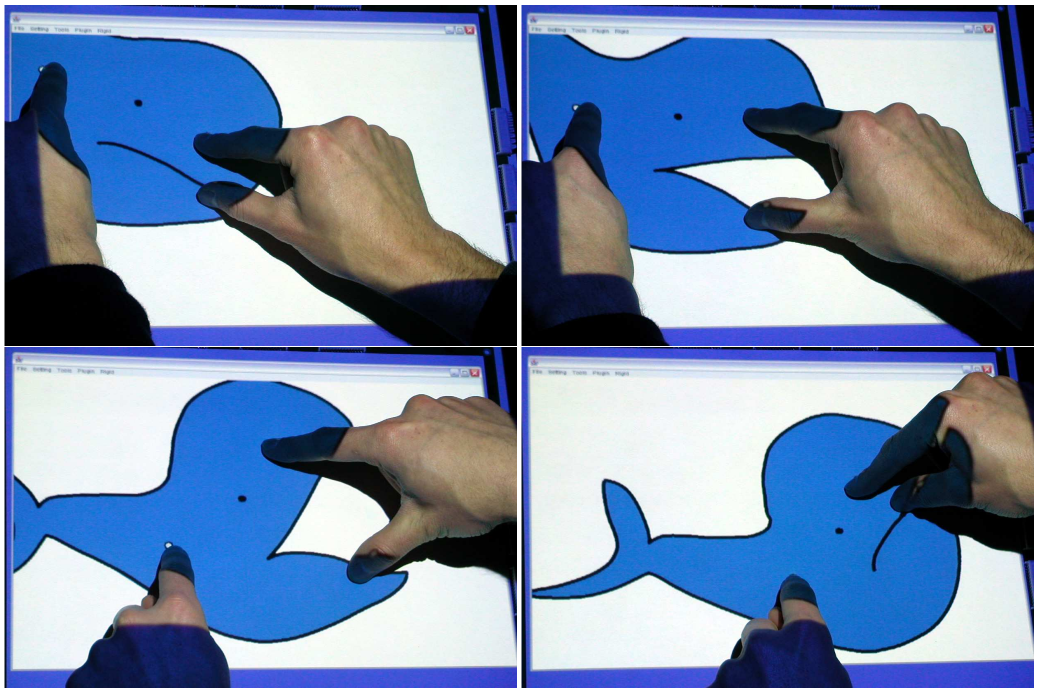
Figure 3.2: A user makes a whale open and close its mouth as it swims.

Our system presents the animator with a drawing of a 2D character projected onto a table. Touching a point on the character pins that point to the animator's finger. The system bends and stretches the character so as to maintain the pinned points fixed to the animator's fingers as they move on the table. Since this interaction is similar to interaction with real-world objects, users grasp the underlying concept right away. In informal demonstrations we found that as soon as participants realized that the drawings respond to their touch, they began to move and stretch them to see how they behave. Participants were able to produce simple motions immediately, and with a bit of practice were able to make more complex and interesting animations.

Our system places no constraints on how users may grasp or move the drawings. This encourages exploration of different types of manipulation in order to discover what works well with each particular drawing. To create a more complex motion, two people may work together, and attempt to coordinate different parts of the drawing. While novice animators using traditional key-framing techniques tend to produce stilted, robotic motion, users of our system produce rich, life-like results, as their animations inherit all of the nuances and imperfections of the users' hand motion. The animations are simple, but they preserve much of the expressive power of hand gestures and body language.

Conversation facilitated by this type of animation is suitable for real-time explanations or collaborative storytelling when all participants are co-present. These animations can also be used for remote or archival exposition by being broadcast or recorded.

3.1.3 Implementation Details

Our setup consists of a digital projector mounted on a platform above a multi-finger touchpad (see figure 3.3). Images are projected onto the touchpad so that points touched by the user are registered with the corresponding points in image-space.

For multi-finger input, we use Rekimoto *et al.*'s SmartSkin [93]. The device produces an image of the proximity of a hand or finger to each point on the touchpad. Points where a finger is in contact with the SmartSkin show up as "hot-spots" in the distance image. To maintain the identity of each contact point, we track them

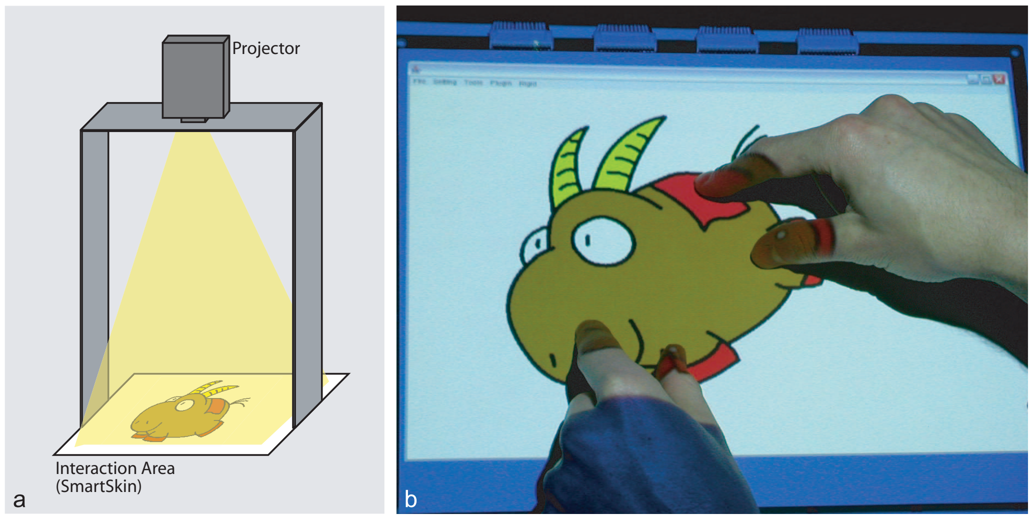

Figure 3.3: (a) Overview of our system setup. (b) Close-up of interaction area.

through a sequence of frames using a greedy exchange algorithm described by Veenman *et al.* [107]. We modify Veenman's algorithm to accommodate changes in the number of tracked points. The algorithm iteratively swaps the inter-frame point correspondences so as to minimize the total distance between corresponding points. While more elaborate error functions are possible (for example, by modeling hand dynamics or structure) we find this technique works sufficiently well. By avoiding assumptions about hand structure, we can handle cases that do not fit such assumptions (such as multiple users).

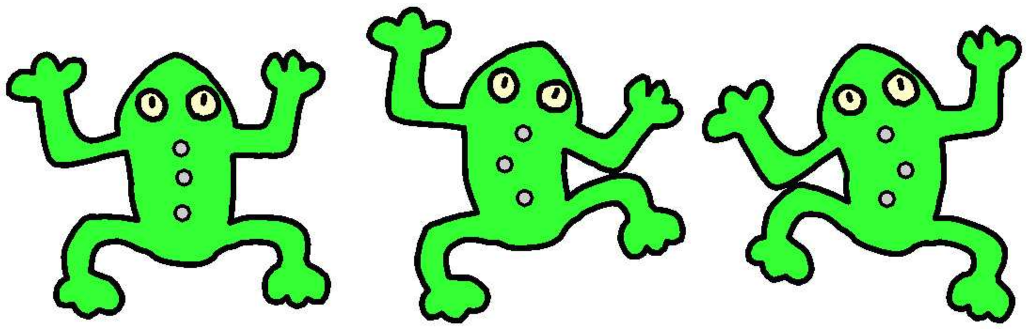

Figure 3.4: Moving an internal constraint point instantly affects the entire shape.

The shape of a drawing in our system is represented by a triangle mesh. This mesh represents the "rest-state" of the drawing. The points where the user touches the drawing serve as constraints for a 2D mesh deformation algorithm developed for this system [63]. The algorithm attempts to maintain the local rigidity of the mesh while meeting the given constraints. It poses the problem as the minimization of a quadratic error metric that measures the distortion associated with each triangle in the mesh. It provides a closed-form solution which gives immediate results. The effect of moving constraint points is global and instantaneous, so small changes may affect the motion of the entire model (see figure 3.4). The object deforms as though made from a sheet of rubber-like material, making for very lively movement that is predictable and easy to control.

3.1.4 Related Work on Accessible Animation

While previous systems have used direct whole-hand manipulation to accomplish a variety tasks [93, 122], none have used the multiple degrees of freedom available from multi-point touchpads to control the many degrees of freedom of an animated character. Real-time performance animation through direct manipulation has been previously accomplished for very simple motions [8, 34]. More complex animation is possible through digital puppetry [100], in which a custom set of controls is mapped to the various parts of each character. The mapping, however, is often arbitrary and requires much practice to master. Motion capture uses a more natural mapping between the movements of the actor and the character. However, it is limited to characters that have real-world counterparts (such as people and animals) and cannot be used to animate arbitrary 2D drawings.

Other attempts at making animation more accessible have used layered recordings of motion to iteratively add motion to an animation [37]. While this approach can yield more detailed motion than our technique, it cannot be used for real-time performance, and it may be difficult to synchronize the separately recorded motions. Another way to control many degrees of freedom at once is to use pre-recorded motion [104], or pre-specified configurations [61, 90]. These approaches produce pleasing animations, but they restrict the type of motion that the user can produce to previously designed animation.

3.1.5 Discussion and Future Work

The interaction technique embodied by our system has important implications to the field of accessible graphics, as well as to multi-touch interaction in general. Previous multi-touch methods have been limited to fairly simple manipulations, such as controlling sliders, specifying bounding rectangles, and 2D positioning (see section 2). Most were restricted to simultaneous control of only three or four parameters using two input points. Our results indicate that a greater level of control is achievable. Users were often observed controlling high-degree-of-freedom characters using four or more touch-points (generally the thumb and index finger of each hand). They were able to bend the characters into a much larger variety of configurations than could be achieved using only two input points. The richness of the resulting animations demonstrates that, if properly used, an increase in the number of input points can lead to a qualitative enhancement in the expressiveness of an interface.

Many everyday activities require the coordination of many degrees-of-freedom. Such activities include transporting multiple objects, or objects that bend or have moving parts, and manipulating a variety of tools. While computer users have the ability to handle these kinds of tasks effectively, their skills are rarely used for digital manipulation. Our animation system shows that multi-touch interaction can take advantage of these manipulation skills to allow for greater throughput, fluency, and parallelism in user input.

The designer of an animation interface must strike a balance between expressiveness and complexity. Expert animators require great freedom of expression, and so are willing to expend the effort required to learn and use a complex interface. The reduced complexity of animation interfaces aimed at novices comes at the cost of limits on expression. These limits are generally set by greatly reducing the number of controllable degrees-of-freedom. The DOFs may be elemental ones, such as object position and orientation, or they may be parameters that control pre-authored state changes, such as time in a walk cycle. However, the low number of available DOFs is a limitation of traditional input devices, not of human motor control. Previous techniques fail to tap the full extent of users' manipulation abilities. Our system was designed specifically to demonstrate how multi-touch input can increase expressiveness while maintaining the simplicity of previous, more constrained, performance animation systems [8, 34, 61]. As such, we have limited the system to real-time performance using a single character. However, the system can be extended in a number of ways that can increase the richness of resulting animations while remaining far simpler than traditional keyframe-based techniques.

Two key issues in increasing the expressiveness of the system are the control of multiple characters, and the coordination of more animation parameters. Simultaneous manipulation of two or more objects can reduce user performance and coordination due to divided attention and a lack of visual integration [12]. Our experience with the system also leads us to believe that controlling more than eight degrees-of-freedom (which appear to roughly correspond to the position, orientation, and span of the two hands) becomes increasingly difficult. One way to increase the number of controllable characters or parameters is through cooperative multi-user control. Given enough space, two or more users can animate separate characters or character parts. This approach builds on the idea of spatially multiplexed input, an important property of multi-touch interaction. However, multi-user input is not always desirable. Instead, we may turn to temporally multiplexed input that can be implemented using sequential recording, also known as layered animation [37, 88].

Just as a musician can record several tracks of music separately and then play them back as a single composition, an animator can sequentially record separate tracks of motion for a single animation. For example, the animator may first record the motion of one character, and then, while this motion is being played back, record the motion of a second character. It is also possible to sequentially control many parameters on a single character. The simplest way of implementing this in our system would be to simply record and play back the motion of the control-points. As a user records a new layer over existing motion, both the current and the recorded constraints are used to deform the character. More complex compositions may also be created. Hierarchical composition would be useful for separating the recording of large-scale motions from the recording of fine details. This may be achieved by embedding the fine-detail control-points within the warped space defined by the large-scale controls. For example, a user may first animate a character's facial expressions, and then record the motion of the character's body at a higher level of the control hierarchy. The face control-points would then be moved along with the larger body deformations.

The creation of richer animations increases the need for animation editing. This would allow the animator to fine-tune motion, and to correct error in position or timing. Since performing animation is fairly quick, one straightforward editing method is to simply re-record short segments of the animation. If a greater level of control is desired, it is possible to create a hybrid between performance animation and traditional key-frame techniques. In such a system, users would define key poses within the performed animation. The position of the control points within this pose may be edited, and the changes propagated within a user-specified time window so as to mantain continuity with the rest of the animation. Similar approaches have been described for motion-capture data using various inverse-kinematics constraint solvers [48, 76]. However, implementing such a system for deformable drawings is trivial, since one must only adjust the input constraints to our deformation solver, and the shape will deform accordingly. Key pose editing is appropriate for changing the location of objects at a given time, however it is often desirable to change the timing of a sequence of animation. One method for editing timing, proposed by Terra and Metoyer [102], is to allow users to perform the timing of a motion path while keeping the path fixed. Features of the performed motion path (*e.g.*, rapid directional changes) are then matched to the original path, and the timing information is transfered. More precise time manipulations can also be performed: time may be warped so as to satisfy constraints on simultaneity or order. For example, a user performing an animation of two colliding objects may find it difficult to get them to bounce off each other at the same time. The software can satisfy a simultaneity constraint by slightly speeding up one object, and slowing down the other [88].

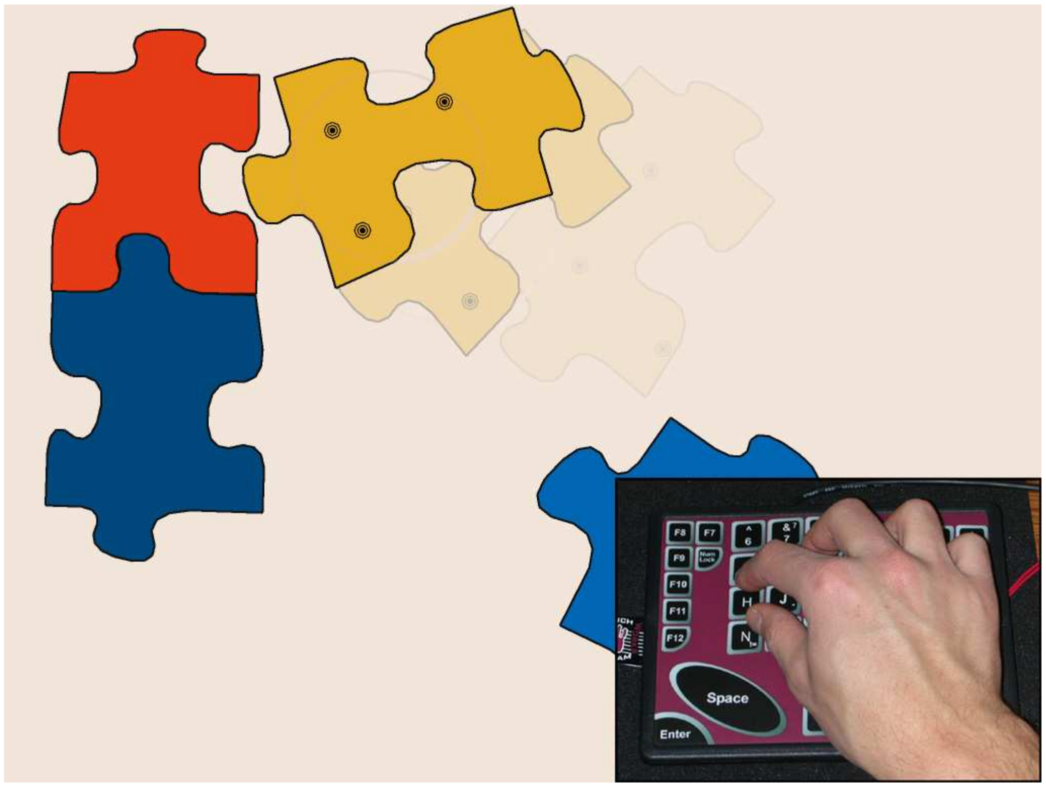

Figure 3.5: The hand cursor lets the user move the puzzle pieces as though sliding objects on a physical table.

3.2 Multi-finger Cursors

3.2.1 Motivation—Cursors as Tools

Our hands are capable of deftly manipulating a remarkable array of objects. Yet when a watchmaker slides a gear into place, he does not touch it directly with his fingers, but holds it, instead, in a pair of tweezers. Direct touch interaction is appealing since it is a simple and familiar concept, but it is not always the best way to accomplish a task. Several interaction techniques have illustrated how multi-point touchpads can be used to control several degrees of freedom in parallel [63, 93, 122], but they all use the devices as touchscreens, superimposing the display directly over the touch-surface. Although touchscreens provide a simple, obvious mapping that is easy for users to understand, they also have a few shortcomings.

These shortcomings are mainly related to physical constraints of our arms, hands, and fingers. Like the gears of a watch, many common UI widgets are too small to be precisely manipulated by the average finger [6]. On a touchscreen, users' hands often occlude the very objects they are manipulating. When the screen is large many areas are out of reach, but holding an arm unsupported for extended periods of time is fatiguing even if everything is within an arm's length [5]. In most home and office environments, people do not interact with the computer by touching the screen directly. Instead, their actions are mediated by a cursor.

The indirection provided by the mouse cursor can have many benefits. For example, by making use of a non-linear mapping between mouse motion and the motion of the cursor, users can achieve precise, sub-pixel control at low speeds, yet still move rapidly across the screen with very little hand motion [23, 66]. By dynamically adjusting the control-display ratio, it is possible to decouple the motor-space from the display-space to make small screen-widgets larger in motor-space, or to make long distances over empty space shorter [20]. Similarly, virtual forces can be applied to the cursor motion to guide users towards likely targets [4]. Other benefits may be gained by adjusting the shape of the cursor. For example, area cursors use a large activation region in lieu of the standard single pixel "hot-spot." This increases the effective width of targets, making small items easier to select. In this light the cursor can be viewed not as a digital proxy for our finger, but as an instrument that enhances our human abilities. Like the watchmaker's tweezers, the artist's paintbrush, or the fisherman's net, cursors let us perform tasks that would be difficult to accomplish unaided.

Still, instruments embodied by traditional 2D cursors are somewhat limited, in that they can be controlled by at most two parameters (the motion of a mouse in x and y). As discussed earlier, limiting input to a single point reduces opportunities for parallel input, and complicates interaction by introducing superfluous modes and control-widgets. In the following sections we introduce a family of interaction techniques that use multi-touch input to control multi-degree-of-freedom cursors. These techniques result in a more fluid and coordinated interaction style than found in their single-point counterparts. They demonstrate the expressive power of simultaneous multi-parameter cursor control, and show how indirection through a cursor helps users overcome the physical constraints that limit similar direct-touch techniques. It is important to note that we have chosen these techniques as representative points in the design space of multi-touch cursors; they may be combined, modified, or extended to suit a variety of applications.

3.2.2 Design Principles

We have attempted to design our cursor techniques so that they would be easy for an experienced mouse user to use and understand. Doing so not only makes the techniques easy to learn, but also makes them simple to integrate with existing single-point user interface frameworks. We do this by maintaining, whenever possible, certain key attributes of the mouse cursor. The first property is a continuous zero-order mapping. That is to say, we use the touchpad as a position control device rather than a rate-control device. Research has shown that position control leads to faster performance than other types of mappings [128]. The limited range of the controller can be addressed by making the mapping relative instead of absolute (users can clutch by lifting and repositioning), and by applying a speed-dependent gain function that allows access to a large screen from a small footprint while still providing sub-pixel accuracy. Note that this is particularly important for multi-finger cursors, as the space taken up by several fingers decreases the effective size of the touchpad.

When using relative positioning devices, users must be able to differentiate between the tracking and dragging states [25]. Our touchpad supports a single press on the interaction-surface that corresponds to a mouse-button press (see section 3.2.4).

3.2.3 Techniques

Hand Cursor

Our first technique is a multi-finger hand cursor. It is a general-purpose cursor which emulates use of a touchscreen. For each finger contact on the touchpad the cursor displays a corresponding point on the screen. The contact points are not used as absolute positions, but rather as positions relative to the hand. The reference frame of the hand cursor, indicated by an enclosing circle, is controlled by the relative motion of the user's hand on the touchpad.

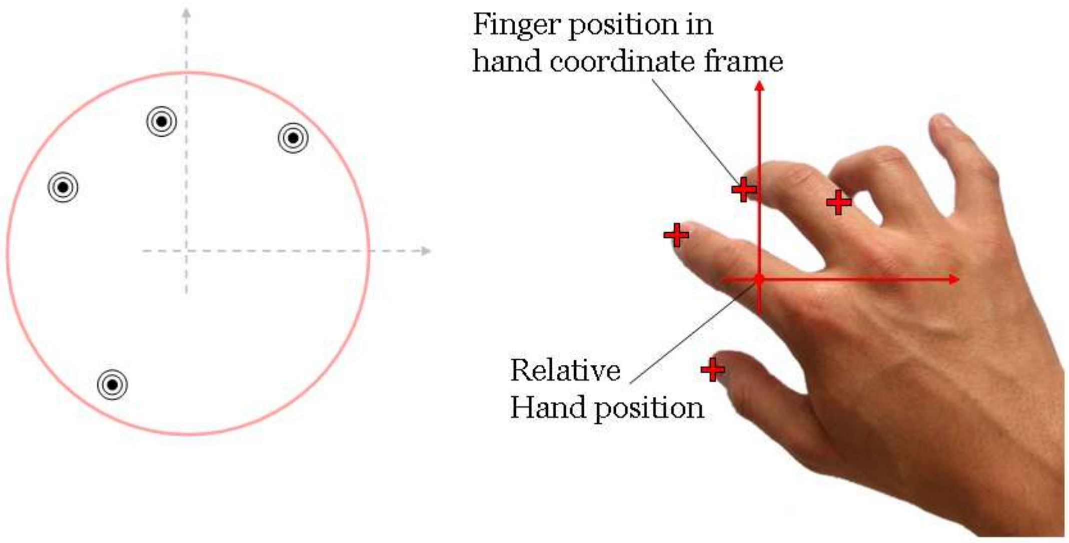

Figure 3.6: The hand cursor maps each finger contact on the touchpad (right) to a point on the screen (left). The position of the fingers is determined within a reference frame set by the position of the hand. Hand position is estimated as the mean finger position. The motion of the hand controls the relative position of the cursor on the screen, the extent and position of which are indicated by an enclosing circle.

The hand cursor allows users to control graphical elements as though manipulating rigid real-world objects. For example, figure 3.5 shows a user moving and rotating a puzzle piece just as one would maneuver a rigid object lying on a table. Multiple fingers also allow the user to grasp several objects at once, which is useful whenever it is necessary to control multiple parameters concurrently. For example, it may be used to control an array of sliders [27] or for modifying the control points of a curve (see figure 3.7). Multi-point input is also useful for performance animation as discussed in section 3.1 (see figure 3.7).

While in theory the movement of five fingers on a surface can describe up to ten parameters, in practice a finger's motion is highly correlated with the motion of the hand and the other fingers [54, 95]. A reasonable way of increasing the bandwidth of interaction is to use two hands. Two-hand cursors are especially well suited for high-degree-of-freedom symmetric bimanual interactions such as shape deformation

[63, 81]. They can also be useful in asymmetric interaction tasks such as controlling the orientation of a toolglass ruler [19, 53, 70].

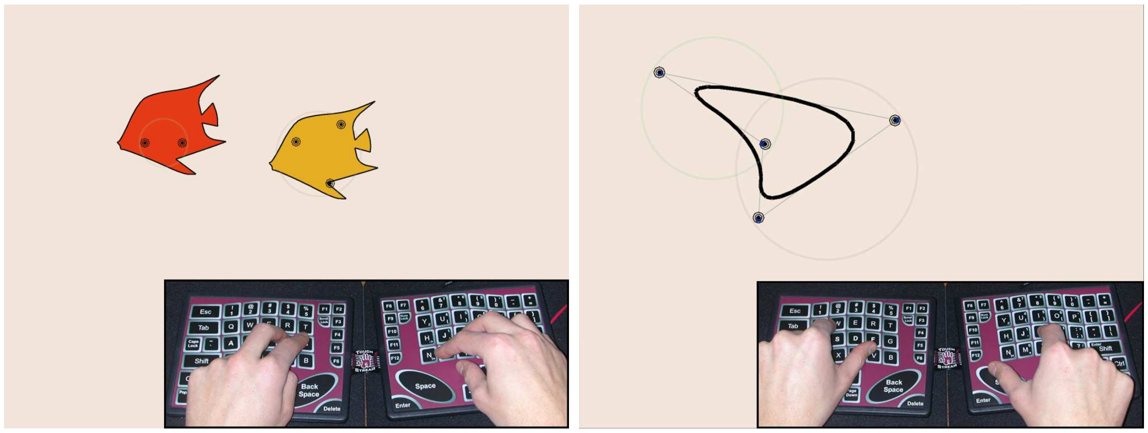

Figure 3.7: (Left) Controlling many parameters at once is useful for performance animation. Here the user animates two swimming fish by using two hand cursors. (Right) Using several fingers allows users to control objects with many degrees of freedom, like the control points of this subdivision curve.

Note that the relative mapping from the touchpad to the cursor is not a homography. While such a mapping is the most straightforward, it has several limitations. For one, touchpads are generally smaller than the display they control, which means that small errors on the touchpad are magnified on the display. There is also a physical constraint on the minimum distance between fingers; contact-points can never be closer than the width of a finger, and this minimum distance may be greatly magnified on a display. A one-to-one mapping also presents problems when using two touchpads for two-handed input. Unless the touchpads were several times larger than the span of each hand, the working space for the fingers of each hand would overlap in a confusing manner that is rarely experienced in the real world.

We solve these problems by scaling the coordinate frame of the cursor so that the finger distances are appropriate for the manipulation task (*i.e.,* small enough to comfortably fit on the interface while maintaining a reasonable reach). We then translate this coordinate frame by the motion of the hand. By applying mouse acceleration to this motion, we give the hand access to the entire screen, while maintaining high precision at low speeds. This relative mapping also allows users to reposition their hand on the touchpad without affecting the cursor. Having finger motion occur at a different scale than hand motion may seem strange, but we have found this technique to be completely transparent. No one who has used our system even commented on it. We believe that this mapping works because it reflects the natural relationship of the hand to the fingers, where the fingers work in a frame of reference relative to the hand. However, there is an important shortcoming to this method that must be considered: Since relative finger motion occurs at a different scale than the global hand motion, it is difficult to independently control the movement of each finger. Moving the hand will move all of the fingers on the cursor, even if a finger remains fixed on the touchpad. We find that in practice, the technique works as long as the fingers operate in concert. For example, moving the control segments of the curve in figure 3.7 is easy, but placing fingers on the control points is difficult.

Implementation Details Since we apply a gain function to the overall translation of the hand, but not to the motion of the fingers relative to one another, we must first separate hand movement from finger movement. By using vision-based hand tracking, or touchpads that detect a hover state, it may be possible to determine the actual hand motion. The TouchStream, however, only provides contact information. We approximate the hand motion using the movement of the centroid of the contact points. Since the finger positions are considered relative to the position of the hand on the touchpad, but the centroid position relative to the fingers changes whenever a finger is added or removed, we cannot use the centroid as the origin of the cursor coordinate frame. Instead, we determine the origin of the cursor by the fingers' initial contact with the touchpad. The coordinate frame is then translated by the displacement of the centroid of the fingers *currently on the touchpad* (*i.e.,* any fingers that are not present in both the current and previous frames are discounted from the centroid calculation).

This method for determining hand position has a few limitations: The motion of any finger may displace the screen-space points of other fingers. Even if all fingers move away from the centroid at the same rate, the detected hand position will change since the fingers are not evenly distributed around the center. Another issue arises when users reposition their hand on the touchpad. Since in general not all fingers touch down simultaneously, yet the cursor's coordinate frame is determined by the initial contact, the origin relative to the fingers may not be where it was during the previous movement. This may be remedied by dropping the first few contact frames, at the cost of a slight delay.

In the real world we commonly use both hands to manipulate a single object. When two hand cursors come close together, it becomes difficult to judge which finger (on screen) belongs to which hand. The circle drawn around the finger-points of each hand cursor serves not just to indicate the reference frame of each hand, but also to help the user perceive the cursors as two separate hands. Note that this indicator only shows grouping. We do not fill in the "body" of the hand. Test users of an earlier implementation which used a translucent disc to indicate the hand attempted to select objects with the disk rather than the fingers. (This type of selection may actually be appropriate for some tasks; see Section 3.2.3.)

Finding best-approximation rigid motions Since human finger motion is not constrained to rigid translations and rotations, we often need to solve problems of the form "Given the original finger positions $P_0, P_1, \ldots$ and their new positions, $S_0, S_1, \ldots$, which transformation T in some class C of transformations has the property that $T(P_i) \approx S_i$ for each i?" where the approximation is in the least-squares sense, *i.e.*, we want to minimize $\sum_i \|T(P_i) - S_i\|^2$. For translations, this is easy: we translate the centroid of the P_is to the centroid of the S_is. For rotations, it is more subtle. Rekimoto *et al.* [93] describes using multiple fingers to move objects rigidly in 2D, but does not present the implementation. The analogous problem, in 3D, has been solved in the vision community [50, 106]. We repeat the solution here for the reader's convenience: Letting P denote a matrix whose columns are the $P'_i = P_i - Q$, where Q is the centroid of the P_i, and S denote a similar matrix for the centroid-adjusted S_i, we seek a rotation X such that $XP \approx S$. To find X, we compute $H = SP^T$, and its singular-value decomposition $H = UDV^T$. In general, we then get $X = UV^T$, provided both $\det U$ and $\det V$ have the same sign, and H has full rank. If the determinants' signs differ, we negate the last column of V^T before computing the product $X = UV^T$. If the matrix H has rank less than two (*e.g.,* if the fingertips all lie along a line segment both before and after moving), then we must add points that lie off that segment before the rotation is uniquely determined.

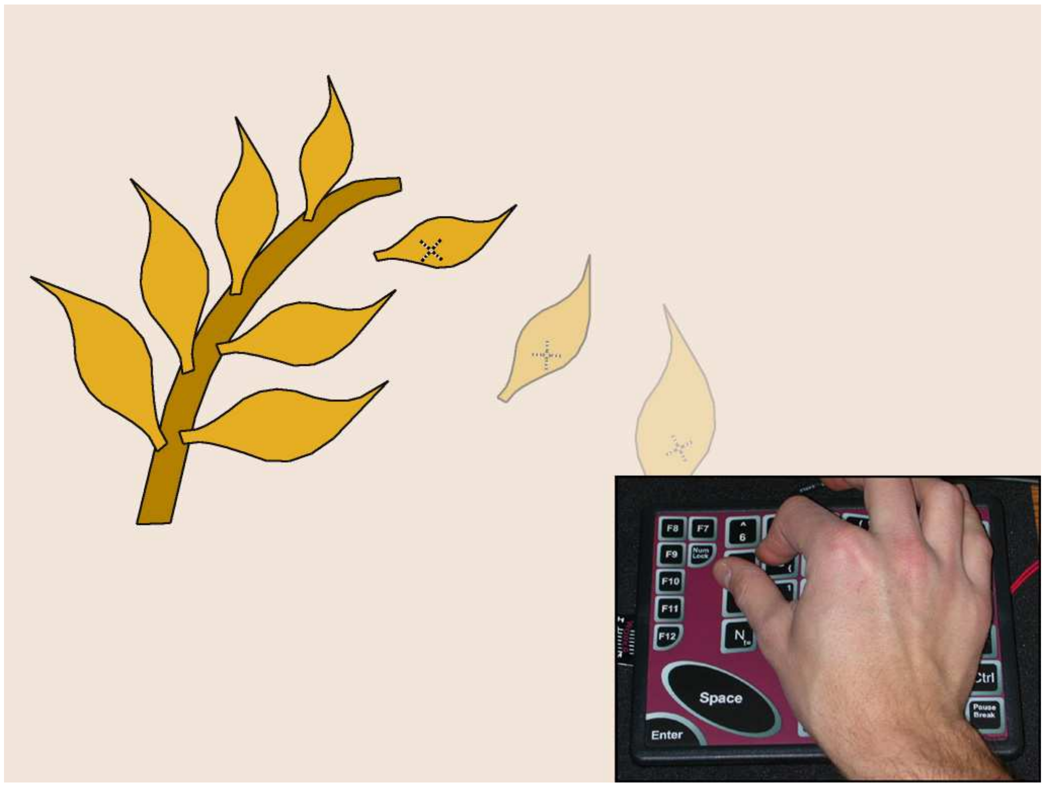

Figure 3.8: A user simultaneously translates, rotates, and scales a leaf using the similarity cursor. Parallel control of an object's properties allows for more fluid interaction, and may decrease task completion time.

Similarity Cursor

Users often control only one object at a time, so it is useful to abstract the parameters of the hand into a single point cursor. Positioning a single point over an object is easier than placing several points, especially when the object is small relative to the width of the fingers. The similarity cursor allows users to focus on a single target while simultaneously controlling its position, rotation, and scale (*i.e.,* determining an orientation-preserving *similarity* transformation).

Users control the cursor using two fingers by a simple mapping from the hand's position and orientation, and the span of the fingers (see figure 3.8). We provide feedback regarding the cursor state by render the cursor as rotating cross-hairs. These show the translating and rotating motion of the hand. Note that we do not resize the cursor to indicate scaling, since it is undesirable to have a cursor that is too large or too small for the task [110]. Instead, we indicate scaling by animating stripes which slide toward and away from the center at a rate proportional to the scale rate.

Rotations, scaling, and translation are very common in illustration and 2D animation software, and in most commercial systems must be performed separately. This is usually accomplished either by switching modes, or by using a different control widget for each operation. With the similarity cursor all three operations may be accomplished in a single smooth motion. Research on symmetric bimanual interaction suggests that, even discounting mode-switching time, increased parallel input correlates with shorter completion times for alignment tasks involving positioning, rotation, and scaling [73]. This is particularly evident at the final stage of alignment, where separately adjusting each property may undo the previous adjustment. We expect that this will hold for parallel input using one hand.

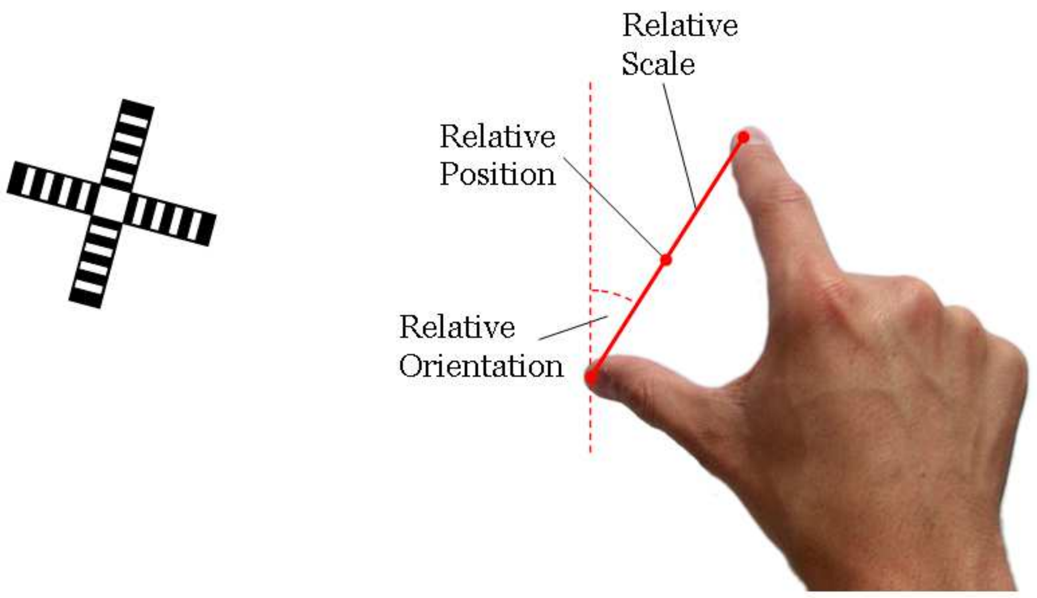

Figure 3.9: The similarity cursor is controlled using two fingers on the touchpad. Change in the position of their centroid controls the relative position of the cursor. Change in the slope of the line they define controls the relative orientation of the cursor. The changing distance between the points adjusts the size of the object manipulated by the cursor, and is indicated by animated stripes.

Implementation Details In our implementation, this cursor is controlled by one or two fingers, but could easily be extended to use the entire hand. Cursor position is controlled by the relative motion of the centroid of the two fingers. We first apply the Windows XP mouse-gain function [86] to this motion to reduce the control footprint and increase precision.

Rotation To determine rotation, we look at the angle of the segment connecting the two touch points. The change in this angle between drag events is mapped to cursor rotation. Due to physical limitations of finger and wrist movement, it is difficult to make large rotations without placing the hand in an awkward position. We remedy this situation by taking advantage of the indirection provided by the cursor, and applying a speed gain function to cursor rotation.

We use the same gain function for rotation as we do for translation. However, since the function is defined in terms of distance, we must first convert the rotation angle to a distance. Given the vector C from the first finger to the second, the analogous vector P for the previous frame, and the rotation matrix R which rotates by the angle between P and C, we calculate the gain distance as: $d = ||RP - P||$.

The system rotates objects about the cursor center after it has been translated by the change in position of the fingers' centroid. Because of this, it appears to the user that the object rotates about the centroid of the fingers. If the user chooses to hold one finger fixed and rotate the other about it, both rotation and translation result. Our informal experience shows that users quickly grasp this idea, and can easily adjust for any unintended translation.

Scaling We set the initial scale factor $s = 1$ whenever an object is selected. If the current and previous lengths of the segment connecting the touch points are l_c and l_p then the new scale factor after each drag event is $s' = s + (l_c - l_p)/d$ where d is the change in length which will increment the scale factor by one. We set d to the height of the touchpad (11.34cm). An alternate design would multiply the scale factor by the ratio of the current and previous segment lengths. While this may be a reasonable choice for some applications, it leads to exponential growth which is rarely useful in drawing applications.

Since it is common for items in digital illustrations and animations to have real-world analogues, it is likely that for many tasks translation and rotation would be more common operations than scaling. However, due to physiological constraints on finger motion it is difficult to rotate the fingers while keeping them at a precisely fixed distance. While the similarity cursor makes it easy for the user to correct scale errors, for many tasks it may be helpful to avoid them altogether. This may be done by using a modifier key or gesture (*e.g.*, two fingers for rotation/translation, three fingers for simultaneous scaling.) Alternatively, a gain function can be designed that attenuates small variations in scale.

Adjustable Area Cursor

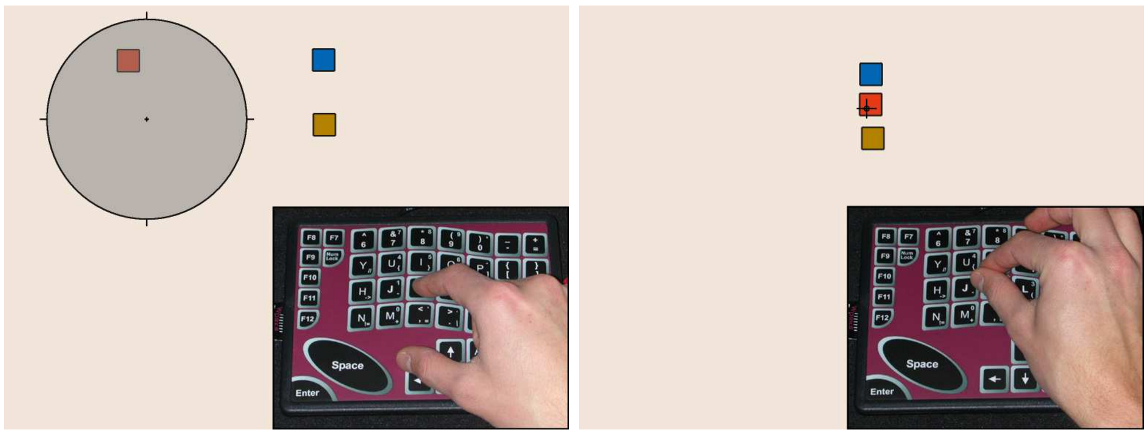

Figure 3.10: (Left) The large activation area of the adjustable area cursor reduces the time and precision needed to acquire isolated targets. (Right) The selection of fixed-radius area cursors is ambiguous in crowded regions. This ambiguity is resolved with the adjustable area cursor by minimizing the activation area.

Our third technique extends the idea of area cursors [67], by allowing the user to control the size of the cursor's activation area. As with a real hand, the size of the cursor's activation area is proportional to the span of the fingers on the touchpad. Users can easily select small isolated objects by simply spreading their fingers and roughly moving the cursor to the vicinity of the object (Figure 3.10 (Left)). The object is selected as long as it lies within the activation area, so precise positioning is unnecessary. To select a specific object from a crowded area users bring their fingers together to minimize the area of the cursor, making it behave like an ordinary point cursor (Figure 3.10 (Right)). For very small targets (such as control-points in a drawing program) it is plausible that users may benefit from using a small or medium activation area even in the presence of clutter. However, since the added cognitive load of selecting an appropriate cursor size may negate the benefits of a larger selection area, this is difficult to judge without a formal study.

An important feature of the adjustable area cursor is that it can distinguish the intentional selection of a single object from the intentional selection of many. This means that users can easily grab ad-hoc groups of adjacent objects (Figure 3.11 (Right)). These groups are simply determined by the radius of the cursor, so they may be quickly created or modified. To control a group of objects current interfaces require an initial grouping step. The adjustable area cursor unifies the grouping and selection steps. Of course, for this to work the group must be sufficiently separated from any objects that are not to be selected. However, even if such “distracter” objects are present the cursor can potentially speed up existing group selection techniques (for example, by using a modifier key to add or remove objects to the selection).

Previous area cursor techniques [52,67] share a problem which makes them difficult to integrate into existing interfaces: They make it difficult or impossible to click on the empty space between selectable objects or interface elements (Figure 3.11 (Left)). This is frequently a valid operation. For example, a user may want to position a text cursor in a word processor, or to create a new shape in a drawing program. The adjustable area cursor solves this problem by letting the user minimize the activation area. This does not require a mode switch, or a change in the user’s conception of the cursor. It is simply a consequence of the adjustable radius.

Implementation Details The adjustable area cursor is controlled by one or more fingers. The cursor is moved by the gain-adjusted translation of the centroid of the contact points, while the diameter of the cursor is set to a multiple of the maximum distance between contact-points. Note that the latter is an absolute mapping, which makes it easy for the user to instantly specify a large or small diameter with the initial touch of the fingers. (A diameter greater than the height of the screen or smaller than one pixel is not very useful, so there is no need for clutching to extend the range of

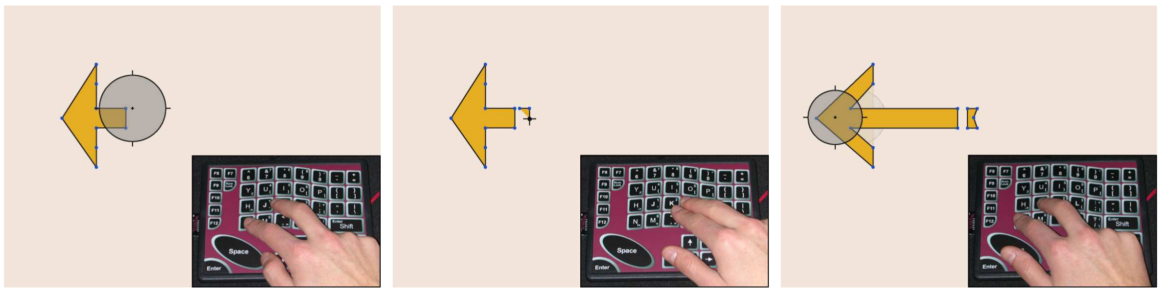

Figure 3.11: (Left) Traditional area cursors make it impossible to click on empty space near selectable objects. (Center) The adjustable area cursor can emulate a point cursor without requiring the user to switch modes. (Right) Since the adjustable area cursor does not suffer from the ambiguity of fixed-area cursors, it can be used to control groups of nearby objects.

reachable diameters.) The control/display ratio for the diameter is set so that a fully extended hand will cover most of the screen. To ensure that a point cursor can be achieved it is important to subtract the maximum width of a finger (if the result is negative, the diameter is simply set to zero).*

When all but one finger is lifted off the touchpad our implementation maintains the last specified diameter. An alternative is to minimize the diameter to create a point cursor, but we believe that for most tasks our choice is preferable. Creating a point cursor by placing two fingers together is quick, and not much more difficult than lifting a finger, and maintaining the last specified diameter has several advantages: The size of the area cursor is likely to be correlated to the density of elements on the screen. If the user continues to operate in the same region, it is likely that the cursor size will remain suitable. When the user manipulates a group of objects maintaining a constant size will be useful if the group needs further adjustment.

We render the cursor as a translucent gray circle (Figures 3.10 and 3.11). Short radial segments extending along the main axes become a cross-hair indicator when the radius is zero. This indicator may be enhanced to disambiguate the selection of partially overlapping targets by modifying its boundary to include all selected objects (as in the Bubble Cursor [52]). A simpler alternative is to highlight all selected targets.

*Of course, the variations in hand-sizes among users must be taken into account; our current implementation uses the author's hand and finger sizes. Since our informal tests have been with people of similar size, this has worked reasonably well.

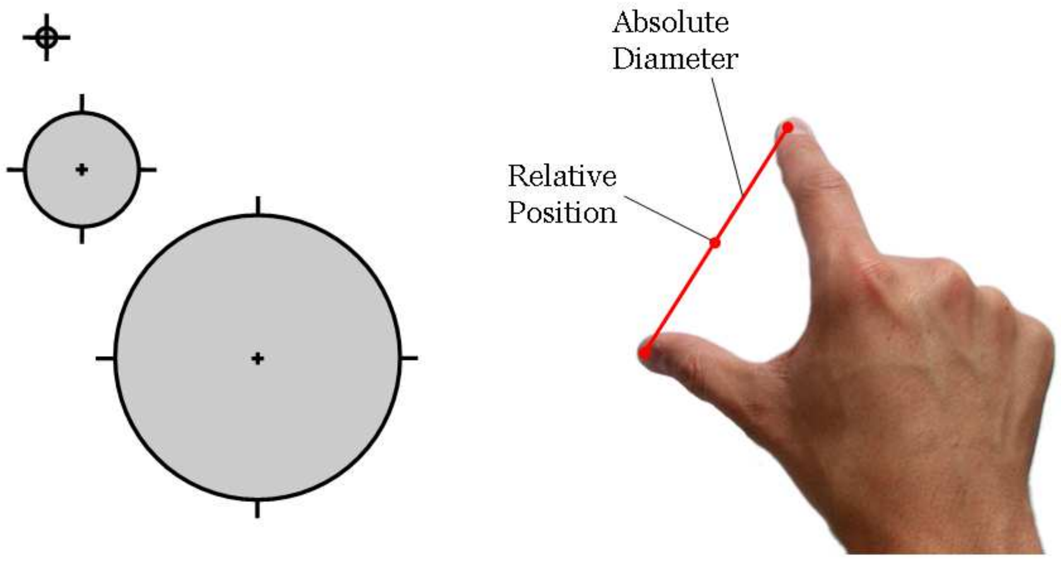

Figure 3.12: The motion of the centroid of the fingers on the touchpad controls the relative position of the adjustable area cursor. The greatest distance between contact points is mapped to the radius of the cursor.

3.2.4 A Relative Multi-point Touchpad

For multi-point input we use a FingerWorks TouchStream touchpad [39]. The touchpad provides a 16.2cm×11.34cm work surface for each hand. It measures the position, velocity, and contact area of each finger at about 100 Hz. In its default mode, the touchpad acts as a relative positioning device, and distinguishes tracking and dragging states using multi-finger gestures. This approach conflicts with our use of multiple fingers for cursor control, so we must use an alternate method to separate the states. Instead, we have the user use a light touch for tracking, and press down on the touchpad to initiate dragging. This technique was described by Buxton *et al.* [27], and enhanced by MacKenzie [83], who showed that tactile feedback indicating a pressure-triggered state change provides greater throughput and accuracy than a lift-and-tap technique or pressing a separate button. MacKenzie also noted that using touch area as a proxy for pressure is suboptimal, and that area thresholds must be determined on a per-user basis. The problem is compounded on a large surface touchpad, where a finger's posture relative to the touchpad is more variable, since changes in posture correlate with changes in contact area. Benko *et al.* [17] uses this fact to create a postural clicking gesture. Unfortunately, this type of gesture limits finger mobility, so it is only suitable for tasks where independent finger motion is not very important (*e.g.*, pointing on a large display).

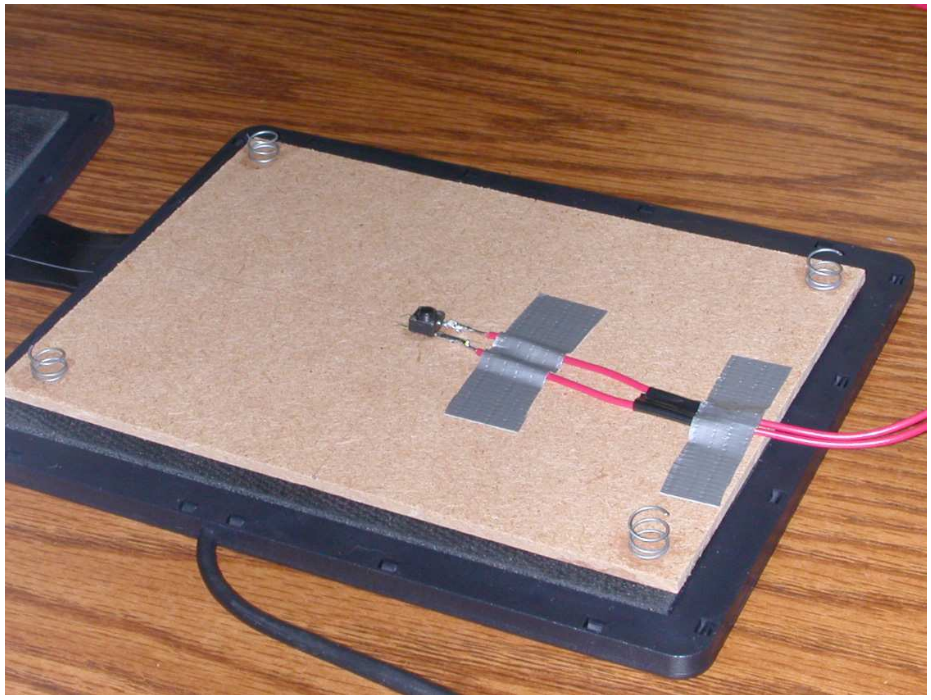

Figure 3.13: Our prototype touchpad uses a tactile switch to allow users to distinguish tracking from dragging motion.

To overcome these problems, we place a single tactile button beneath the touchpad (Figure 3.13), and support the touchpad with a spring at each corner. The stiffness of the button and springs must be chosen so that users do not inadvertently press the button, and to minimize fatigue during drag operations. The button provides a crisp "click", like a mouse button, making the distinction between tracking and dragging clear. Note that this precludes independent drag states for each finger. But since finger movements are not completely independent [54] it is likely that such a level of control would be difficult for most users. This technique appeared to work fairly well in informal tests—one user did not even notice that he was using a button. However, some users had trouble finding the right level of pressure to keep the button pressed while dragging, and consequently pressed harder on the touchpad, increasing their fatigue. This problem may be addressed either by further adjustment of the button stiffness, or by only using the button's down event to initiate dragging, and terminating dragging when the hand leaves the touchpad.

For most cursors, using the touchpad as a relative input device is simple. We just add the gain-transformed change in hand position to the current cursor position. For the hand cursor there is an extra complication that is discussed in section 3.2.3. Note that current and previous hand positions must be calculated only from fingers that are *currently on the touchpad*. Otherwise the estimated position will change dramatically whenever the user adds or removes a finger from the touchpad. Since using multiple fingers decreases the effective size of the touchpad, adjusting the gain on cursor motion is essential to minimize clutching [66]. Our cursors use the Windows XP gain function [86] which in practice yielded a Control/Display ratio ranging between 0.8 and 3.6.

3.2.5 Evaluation

We have incorporated two of our cursor techniques into a simple vector-based drawing program in order to evaluate the methods within the context of a real-world application. Two important tools in the Adobe Illustrator program are the object control tool and the control-point manipulation tool. The first is used to select, move, resize, and rotate shapes, while the second is used to edit Bezier curves by selecting and moving their control points. We have created two new drawing tools that are modeled after these Adobe Illustrator tools, but replace the standard single-point cursor with a similarity cursor, and an adjustable area cursor.

Our prototype drawing program is shown in figure 3.14. It provides tools for curve drawing and editing and for shape manipulation, which are selected using toolbar buttons or keyboard shortcuts. Our three-state touchpad is used to drive both the cursors for the drawing tools and the system cursor. This illustrates a nice feature of multi touch cursors: users can control multi-touch cursors within the drawing area, but as the cursor exits this area, it seamlessly switches to the system cursor, allowing users to interact with standard graphical interface elements such as buttons, sliders, or a color-picker. There is no need to explicitly select a mode, or to switch input

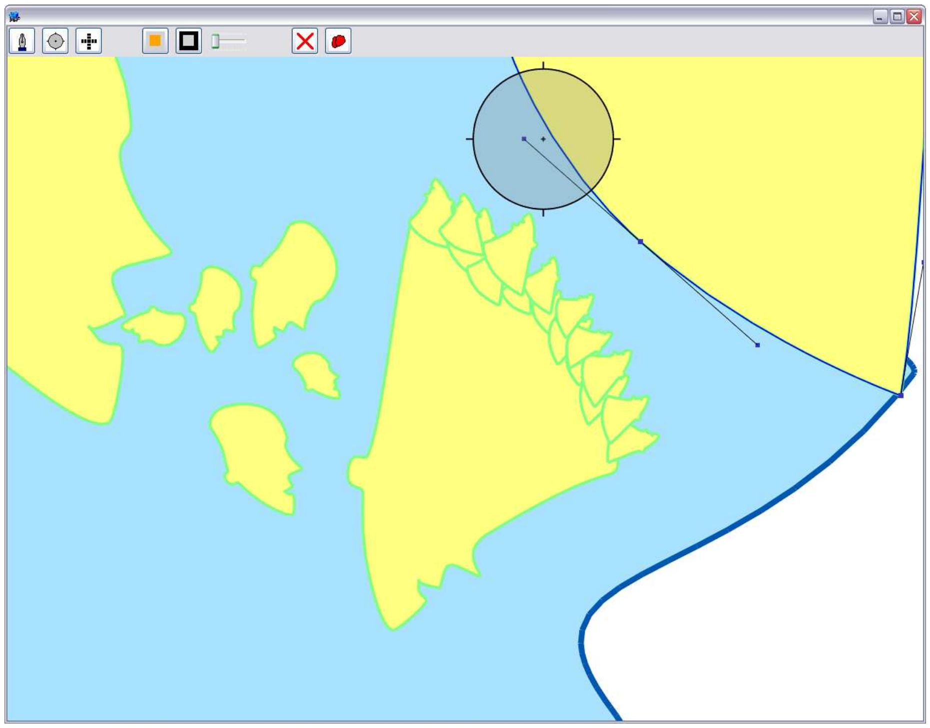

Figure 3.14: Our prototype illustration system uses the similarity cursor and the adjustable area cursor for shape and curve manipulation. Drawing is provided by courtesy of one of our artist participants.

device.

The object-control tool acts just like the similarity cursor, as described in section 3.2.3. It allows for object selection and for simultaneous translation, scaling, and rotation. Unlike its Illustrator counterpart, it does not allow for axis-aligned non-uniform stretching. The tool allows users to constrain motion to exclude rotation, by holding down the Control modifier key, and to exclude scaling by holding down the Shift key. The curve-editing tool acts like the adjustable area cursor. It can select and move one or more control points that lie within its selection radius. Moving a control point that lies on the curve also moves the corresponding tangent control points on either side. Moving a tangent control point rotates the point opposite the corresponding curve control point so as to maintain continuity. By holding the Control key, a user can disable continuity preservation in order to create a cusp. Unlike its Illustrator counterpart, the point-control tool does not allow for points to be added or removed from the selection; each click defines a new set of selected points.

To evaluate our system, we asked three art students from the Rhode Island School of Design, who were experienced users of Adobe Illustrator, to try out our system. The artists were told that we were developing several novel digital illustration tools, and required their critique and feedback to help us improve them. Each artist used the system for a minimum of 30 minutes. This was followed by a 15 minute interview regarding the touchpad and illustration tools. The overall response was very positive, but the artists did help up identify some important design considerations. Some of these are particular to illustrators, others are apply more generally.

The Touchpad While all of the artists were comfortable using the touchpad within a few minutes, the greatest barrier in learning to use it was unlearning the skills and habits of using a traditional touchpad. Initially, all of the artists attempted to use the touchpad by controlling the cursor with their index finger, and clicking by pressing down with their thumbs. All required multiple reminders to use both fingers together. Another issue identified was the sensitivity of the switch. The artists liked having a switch integrated into the touchpad, but some found it to be too sensitive, a state which resulted in unintended clicks. While one artist did say that he found the touchpad less fatiguing to use than a standard laptop touchpad, all agreed that they greatly prefer using a Wacom tablet and stylus for illustration purposes, citing the tablet's superior accuracy, and the feel of control and precision provided by the stylus. This indicates that it may be desirable to create a device that supports both stylus and touch input [123].

The Similarity Cursor The artists all found the similarity cursor easy to learn, saying that if felt very "natural." However responses were mixed regarding its utility. One artist said that he preferred the technique to the Adobe Illustrator method, since it was "more intuitive, and you don't have to switch tools." Another preferred the Illustrator method, saying that having separate tool makes it "more stable" and precise. This artist suggested adding a modifier key to constrain translation during rotation and scaling.

The Adjustable Area Cursor The artists found it more difficult to learn to use the adjustable area cursor than the similarity cursor. The trouble appeared to lie in the absolute mapping between hand span and size. One artist “had trouble controlling size at first,” saying that the it seemed to jump. It may be that a relative mapping would be easier to learn (as no one had trouble scaling using the similarity cursor). But due to the limited useful range for the radius the ability to instantly create a very small or large area should be more useful. All artists were able to use the cursor effectively during the session, although not all felt completely comfortable with it. All did appreciate the ability to control multiple points simultaneously, and used this feature when drawing. Although artists frequently used the cursor with a large area, none noticed it being any easier to select individual points than with a standard cursor. One artist preferred the area cursor to the traditional Adobe Illustrator tool, saying that it was fun to use, “sort of natural,” and that the transparency “lets you see what you are getting at.”

3.2.6 Discussion of Multi-finger Cursors

When an artist, in traditional media, selects a pen or a brush in lieu of finger-painting, she overcomes the limited resolution and shape of her fingers. Likewise, by using an intermediary *cursor* instead of direct-touch manipulation, we can provide users with increased precision, greater reach, and a more capable grasp. Using multiple fingers to control such cursors allows for increased parallelism, which may simplify the phrasing of interaction tasks [78].

Our initial experiments with multi-finger cursor control have produced three graphical interaction techniques that offer several benefits over traditional cursor techniques. The clear benefits include more fluid interaction through parallel input, lightweight grouping, and resolution of outstanding issues with area cursors. The techniques are immediately applicable, as they fit easily into current GUI frameworks.

Other potential benefits of these methods require further study before they can be ascertained. For example, while it is clear that the area cursor would work well for isolated targets, the additional cognitive load it imposes may reduce its performance in dense areas. Similarly, it is not clear how quickly this cursor’s usability for group selection deteriorates as the complexity of a group's structure increases. Studies of these properties exist in the literature for other techniques. Grossman and Balakrishnan's experiments on the bubble cursor [52] may be used to determine the area cursor's sensitivity to "distracter targets," while the work of Mizobuchi *et al.* may be used to determine the effects of cohesiveness and shape complexity on selection performance.

It is also important to study the effects of using the same muscle groups to control a cursor parameters while simultaneously indicating the dragging state by maintaining pressure on the touchpad. The method may reduce accuracy and increase fatigue. Clicking and dragging using the three-state touchpad should be compared to alternatives such as using a mode-switch controlled by the other hand or a foot-pedal.

There are also some problems with our techniques that remain to be solved. Our implementation of the hand cursor makes it difficult to move fingers independently—the position of each finger point is so dependent on the position of the other fingers, that it may move even if the corresponding physical finger remains stationary on the touchpad. Additionally, while our techniques make it easy to control parameters simultaneously, they sometimes make it difficult to control parameters independently (*e.g.,* rotating without translating or scaling). Techniques for constraining cursor motion need to be investigated.

These techniques also suggest further study on the limits of multi-finger control. How many parameters can comfortably be controlled? How do physiological constraints limit the types of viable interaction? Some of these questions are addressed in Chapter 4.

We have observed an interesting phenomenon in our informal tests of the system: When using multiple fingers to control cursors, certain behaviors appear that resemble gestures. For example, a user will place two fingers together to turn the area cursor into a point cursor, or lift all but one finger to restrict motion to translation. These hand gestures are not arbitrarily assigned to these meanings—they are a consequence of the properties of the cursors themselves, and can be learned by the user without referring to a manual.

The design space for multi-finger cursors is still largely unexplored, and contains

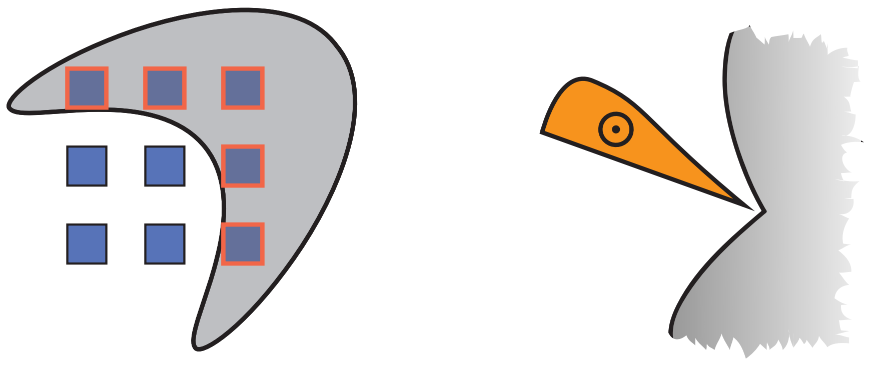

Figure 3.15: More cursor possibilities: A shaped area cursor for more precise selections (left) and a parametrized 2D sculpting tool (right).

many enticing possibilities. For example, just as our we can shape our grasp to accommodate the form of a manipulated object [82], we may be able to adjust the shape of the area cursor to allow for more precise grouping (see figure 3.15(left)). Snapping the cursor's area to select an integer number of targets may improve performance, while an area cursor that can rotate and scale its selection may be useful for drawing applications. Other types of cursor instruments may also be designed. For example, a parametrized cursor whose size, shape, or orientation can be adjusted may be used as a 2D "sculpting" tool to shape drawings much like a sculptor shapes a piece of clay (figure 3.15(right)).

Multi-finger cursors may also have applications in 3D manipulation. For example, positioning and orienting an object in 3D requires control of twice as many parameters as are needed in 2D. A number of specialized three- and six-degree of freedom input devices have been created [2, 9, 44, 58, 130] to address this problem, but they do not perform well in standard 2D interaction tasks. Since the supporting interface in 3D software is often 2D (menus, sliders, etc.) users are forced to alternate between input devices. Multi-point cursors may allow users to operate traditional 2D UIs, while providing the larger number of control parameters necessary for 3D manipulation.

3.3 Discussion and Lessons Learned

Our experience with the multi-touch techniques described in this chapter has lead us to make a number of observations which we believe are important to multi-touch interaction design. First, it appears that multi-touch interaction is very well suited to object manipulation tasks. By using the analogy of physical manipulation as a model for mappings from hand measurements to object parameters, one can construct easy-to-learn techniques that leverage our real-world interaction experience. Specifically, such mappings appear to allow users to coordinate the control of object position, rotation, and scale using fingers of one hand, while using two hands allows for both rigid-body control, as well as for stretching and twisting operations.

It is important to note that people cannot independently control each of their ten fingers in continuous, coordinated manipulation; the motion of the fingers are highly correlated. However, while the manipulations described above can generally be performed with just two fingers on each hand, one should not conclude that using more than two fingers is without utility. More fingers may be used to specify scope (as with the hand cursor's control of multiple objects) and to increase the user's reach and range of motion. Different finger pairings also have different strengths and weaknesses. For example, pointing and selection tasks (*e.g.,* using the adjustable area cursor) appear to be easier to perform with the index and middle fingers, while rotation and scaling is best supported by the thumb and an opposing finger (particularly the index and middle fingers). This is partially due to the greater range of motion available using the thumb and opposing fingers, but other structural and cognitive factors may also be at play.

Perhaps one of the greatest advantages of multi-point input over single-point input, is that the removal of the single-point constraint on design opens up many possibilities for novel solutions. However, exploring this new design space may be difficult. The advantage of a new technique over an old one is not always clear. We do not yet know under what conditions people can effectively coordinate multiple degrees of freedom. It is also unknown when such coordination leads to better performance than serial parameter manipulation, nor how multi-touch interaction using one hand compares to similar solutions using two hands. Answers to these questions could greatly help designers of future multi-touch interaction. The following chapter makes a first attempt at addressing them.

Chapter 4

Principles of Multi-touch Interaction

4.1 Introduction

The techniques described in Chapter 3 illustrate that in order for multi-touch interaction to advance beyond simply mimicking physical manipulation, designers must construct higher level mappings, or "instruments," that transform touchpad input into software parameters. In this chapter we discuss concepts and design guidelines for ensuring that these mappings are effective. We present experimental results which validate these guidelines, and provide further insight into user perception and control of multi-touch interaction.

As discussed in Chapter 2, researchers have developed many multi-touch input technologies, and a variety of interaction techniques that rely on such input. Researchers have also studied the design of two-handed interfaces, as well as multi-touch gestures, but *continuous* multi-touch interaction is not yet well understood. Designers of multi-touch interaction techniques currently rely only on guesswork and intuition in their work. Unlike a standard mouse or touchpad, a multi-touch device offers a large number of input variables which can be represented in various ways. Making usable mappings between these measurements and software parameters can be difficult. The design space is large, and inappropriate mappings can lead to poor user performance and confusion.

A number of biomechanical, neural, and cognitive factors have bearing on the effectiveness of a mapping. For example, physical constraints limit the motion of the joints in the hand and wrist [82], while force-enslaving of neighboring fingers reduce their ability to move independently [54]. Similarly, the divergence of the output of neurons in the primary motor cortex yield coordination patterns in finger motion [95], which limit the number of effective degrees of freedom.

We focus our study on two factors which we believe are critical to the interaction design process, but which have not yet been studied in the context of multi-touch interaction. The first is the relationship between the structure of the degrees of freedom of the hands and fingers to the control structure of the task. We show how designers can ensure a match between the two structures, and that providing such a match yields better performance than an unmatched mapping. The second factor is the effect of perceptual-motor compatibility on task performance. We uncover what visual mappings are compatible with multi-touch interaction using one and two hands, and show that compatible mappings lead to better performance and less confusion than incompatible mappings.

4.2 Motivation

Most existing multi-touch interaction techniques use the interaction surface as a touchscreen. The display and motor spaces are aligned, and users interact with interface components by touching their image with their fingers [47,63,69,85,93,118,122]. This leaves the choice of mapping between fingers and parameters up to the user. Users are well prepared to make this choice since the interaction is based on the analogy of touching and manipulating physical objects.

The advantage of these touchscreen techniques is that they are easy to learn and understand. The interface designer can meet the user's expectations by maintaining an analogy to the physical world. However, this analogy imposes rigid constraints and limitations on possible interactions, which reflect the physical constraints of the hand [6]. Fingers are wider than many UI widgets common today, making precise selection difficult. The hands and fingers often obscure the very object the user is manipulating. On large displays many objects are simply out of reach. These limitations can be overcome by creating more complex, indirect mappings between the control and display space [129]. These indirect mappings are analogous to our use of instruments [16]. Such indirection has been used to enhance pointing precision on single-point touch-screens [6], and to increase the user's reach on a large display [43, 84]. Benko *et al.* [17] enhance pointing precision on a multi-touch display by using a second finger to dynamically adjust the control-display ratio, or to scale portions of the screen. The problem of increasing range and precision is also addressed by the multi-touch cursor techniques discussed in chapter 3. The hand cursor applies cursor acceleration to the coordinate frame of the user's fingers, while the similarity cursor increases the range and precision of rotation control by applying a gain function to hand rotation.

These indirect methods represent powerful tools for high-degree-of-freedom control, but selecting a mapping between the user's fingers and parameters of the software is now up to the interface designer rather than the user. An appropriate choice is not always obvious, since there is no clear physical analogy. Even if a clear mapping exists, its effectiveness is difficult to predict, as it is governed by a large number of physiological and cognitive factors. Some of these factors have been explored in the area of bimanual interaction. For example, several researchers have noted that in two-handed manipulation using two cursors, users become disoriented when the right-hand cursor crosses to the left of the left-hand cursor [57,74]. Balakrishnan and Hinckley have shown that transformation of the hands' relative frames of reference can reduce performance. It has also been demonstrated that bimanual coupling is affected by visuomotor scale transformations [113].

On what basis, then, can a designer choose an effective mapping? We suggest that appropriate mappings can be selected by examining two types of relationships between the degrees-of-freedom of the hands and the control task. The first relationship is the degree to which the user's physical actions are similar to the visual feedback provided by the system (*e.g.,* are they in the same direction?). The second relationship is the match between the *structure* of the degrees of freedom of the control and the perceptual structure of the task (*e.g.,* does a pair of parameters vary together or independently?). These two relationships are discussed in detail in section 4.3 where we explain why ensuring a compatible perceptual mapping, and matching control structure are good guidelines for multi-touch interaction design. However, these guidelines cannot be applied without an understanding of the perception and structure of multi-touch interaction. In section 4.4 we describe an experimental analysis of these aspects of parameter mapping, and demonstrate how these principles can be applied to the design of an interaction task.

4.3 Parameter Mapping in Multi-finger and Bimanual Interaction

The degrees-of-freedom provided by a multi-touch digitizer have properties and structure imposed by the form of the touchpad, and by the nature of human hands and cognition. These properties and their relation to the properties of software parameters determine the quality of a mapping between points on the touch-surface and the software.

The most obvious properties are the physical ones. Points reported by the touchpad are two dimensional, where the range of the X dimension is limited by the width of the surface, and the range of the Y dimension is limited by the height. The effective resolution along each dimension is constrained by the resolving power of the digitizer, and by the limits of human precision. The points form one or two groups determined by the hand of their corresponding fingers. The three dominant parameters of each hand (its X and Y position and its orientation) establish a reference frame for the points in its group. Since these parameters affect the position of entire groups of points, it is useful to separate the degrees of freedom into hand parameters and finger parameters withing the hand's reference frame. The hand position is limited by the width and height of the touch-surface, while its orientation is constrained by physical limits on wrist rotation and arm movement [80]. Similarly, the position of the finger points is constrained by limits on each finger's extension, adduction, and abduction. In contrast to these fixed limits and structure, the range and resolution of software parameters are bounded only by the amount of machine memory allotted to them, and they may be organized in a variety of ways. In practice the bounds and structure of these parameters are specific to the task in which they are involved.

Another important attribute of control dimensions is the degree to which groups of dimensions have an *integral* structure or a *separable* structure. Integrality of dimensions refers to the degree to which these dimensions vary together, or are perceived to form a unified whole. Measures along integral dimensions form a Euclidean metric space, while distances in separable spaces are measured by the Manhattan distance. The importance of matching the input control structure of a system to the structure of the task was first put forward by Jacob *et al.* [65] with regard to Garner's theory of the perceptual structure of visual information [46]. Experiments by Garner and others show that distances in perceptual spaces between objects possessing multiple attributes can follow either a Euclidean or a Manhattan metric depending on the type of attribute varied. For example, the X and Y dimensions of an object's position have an integral structure, while its color and shape are separable. Jacob *et al.* put forward the hypothesis that user performance can be enhanced by allowing users to control separable dimensions with an input device that controls the dimensions separably (*i.e.,* without changing other parameters) and to control integral dimensions with a device that can control them integrally (*i.e.,* makes it easy to move along several dimensions). An experiment by Jacob *et al.* using integral and separable matching tasks controlled by integral and separable input devices appears to support his hypothesis.

Later work by Wang *et al.* [111] points out that the visual pathways responsible for object perception are separate from those guiding action. Since the way people perceive an object may be unrelated to how they act when grasping or manipulating it, perceptual integrality does not necessarily play a role in forming the structure of a manipulation task. The authors put forth the hypothesis that human ability to control properties of an object separately or concurrently depends on the existence of independent visuomotor channels. For example, one hand controls position and another controls orientation. They show, experimentally, that human transportation and orientation behavior exhibits a parallel structure for part of the duration of object transport. That is, transportation occurs separately from orientation for part of the time, and concurrently for part of the time. However, the onset of orientation varied with the target distance, implying that the two processes are not independent. In the language of Jacob *et al.*, the task would be characterized as both integral and separable. A perceptual structure cannot be both simultaneously, hence the authors conclude that the structure of a manipulation task is a consequence of the human visuomotor control structure rather than of the perceptual structure of the task.

In this dissertation we will use the term *integral control structure* to refer to methods of input that make it difficult for the user to move along a single dimension without affecting another. For example the X and Y dimensions of a finger's position have an integral structure. We will use the term *separable control structure* to indicate methods of input that allow the user to keep one parameter fixed while varying another. For example, the extension of a finger can be kept fairly constant during hand motion, since it is controlled by a separate muscle group, and possibly by an independent visuomotor channel.

A third important relationship between control parameters and software parameters is the degree to which the user's physical actions are similar to the visual feedback provided by the system. *Stimulus-response compatibility* is a well-studied principle [40] which states that matching properties of a control to properties of a visual stimulus leads to superior performance over a mismatched control. In an experiment matching the direction of motion of a joystick-control to the direction of cursor motion in the visual field a compatible mapping yielded significantly shorter movement and reaction times, as well as a significant decrease in errors [120]. Issues with stimulus-response compatibility are familiar in the field of human-computer interaction. For example, when the left-hand cursor in a bimanual control task moves to the right of the right-hand cursor, users have trouble identifying which cursor is controlled by which hand.

4.3.1 Multi-touch Interaction Using One and Two Hands

In the following investigation we limit our focus to interaction using two fingers on a touchpad. We believe that two point interaction makes a good "base-case" for the more general problem of multi-touch input. Furthermore, since using one finger on each hand is essentially two-handed interaction, this choice relates multi-touch interaction to bimanual interaction, which has been extensively studied (see section 2.2.3).

The literature on the subject contains many examples of useful interaction techniques that outperform their one-handed counterparts. These include two-handed menu systems [18], illustration techniques [70, 74] methods for 2D and 3D manipulation [73, 125], and more. Since all of these techniques simply take the motion of two points as input, it is reasonable to ask if these points could just as easily come from two fingers on the same hand instead of two different hands. Using only one hand for input has several advantages. It leaves the user's other hand free to control other aspects of the software (such as selecting modes or executing keyboard shortcuts) or to interact with other people or the environment. A single hand also requires a smaller workspace, and may provide for better integration of parameters.

There is another reason to focus on two-finger interaction: It is likely that the major degrees of freedom of the hand can be represented by only the thumb and finger. A study by Santello *et al.* [94] reveals that more than 80% of the variance in static hand postures can be accounted for by only two principle components. Both components describe the opening and closing of the grasp via flexion of the finger joints and rotation of the thumb. The motion of fingers is linked both mechanically and neurologically [54,95], yielding highly correlated movements. Thus we expect that use of more than two fingers on the same hand would yield only minor advantages (especially when constrained to the surface of a touchpad). This does not preclude the use of more than two fingers, it simply means that groups of fingers should be considered together. For example, fingers forming part of a grasp act as a single highly coordinated group referred to as a "virtual finger." In many common grips the thumb acts as one virtual finger, while the rest of the fingers act as another [82].

This high correlation in finger movements highlights the different control structures of one and two-handed input. We would expect that the motion of fingers on separate hands may be more easily decoupled than of fingers on the same hand. Moreover, it may be easier to coordinate the motion of the fingers of one hand than of two. We hypothesize that user perception of touchpad interaction using one finger on each hand is compatible with control of two positions, and perception of interaction using the index finger and thumb of one hand is compatible with controlling a position, an orientation, and a scale or distance related to the span of their hand. The differing control structures affect how well users can coordinate or separate the control of different parameters. The way each of these degrees-of-freedom is perceived determines the compatibility between users' motor actions and the visual feedback they receive.

4.3.2 Measuring Coordination

An important aspect of high-degree-of-freedom control is the user's ability to manipulate multiple parameters in parallel. Coordinated control, is essential for certain applications (*e.g.,* performance animation) while in others a high degree of parallelism speeds up performance. In order to compare the effect of different mappings on coordination ability we require a metric for measuring parallelism. One of the simplest is *simultaneity*, which is defined as the percent of time that the user simultaneously adjusts all parameters under consideration. This measure has been commonly used due to its simplicity [24, 73, 78]. However, this measure does not account for the magnitude of the adjustment of each parameter, or for whether the adjustment is relevant to the task. For parallelism to be beneficial, parameters must be adjusted in a coordinated fashion.

What does it mean for a user to coordinate the control of several parameters? Zhai and Milgram [129] propose efficiency as a measure for coordination. Efficiency relates the actual distance d users traverse through parameter space to the length s of the shortest path. Just as walking diagonally across a square lawn takes fewer steps than walking along its border, a user who is able to coordinate the control of the required parameters will traverse a shorter path in parameter space. This measure assumes that any extra work users perform is due to imperfect coordination. To measure this extra work, Zhai and Milgram define the *inefficiency* metric as a percent of the minimum necessary work: $(d - s)/s$. This defines perfect coordination as zero inefficiency, while less coordinated action has a greater inefficiency. Note that unless d is linearly dependent on s, this measure is scale-dependent and can only be used for comparisons on tasks of the same scale.

The inefficiency metric is well suited for tasks where users aim to accomplish a fixed goal, or to reach a static target. However, it does not apply to pursuit tasks, where users track a moving target, or path-following tasks such as segmenting an image. Balakrishnan and Hinckley define a measure of coordinated parallelism for bimanual pursuit tasks [12]. The measure can be applied to any pair of parameters, or to two logical groups of parameters (such as hand position). The metric measures how much two hands or fingers work together to simultaneously reduce tracking error. Given a point $\mathbf{p}$ traveling through $\mathbf{p}'$ on the way to goal point $\mathbf{g}$, the fractional error reduction for that point is $q = \|(\mathbf{p}' - \mathbf{p})^{\mathrm{T}}(\mathbf{g} - \mathbf{p})\| / \|\mathbf{g} - \mathbf{p}\|$ clamped between 0 and 1. The instantaneous parallelism for two points is then $\min(q_0, q_1) / \max(q_0, q_1)$ if both fractional reductions are positive, and 0 otherwise. This metric yields a mean parallelism of 0 when each finger's error is reduced sequentially, and a mean value of 1 when the fractional errors of both fingers are simultaneously reduced by the same amount.

4.4 Perception and Control of Multi-touch Input Using One and Two Hands

The following investigation aims at improving the understanding of continuous high degree-of-freedom input using multi-point touchpads. We hypothesize that the structure of the degrees of freedom of the hands and fingers, and their relationship to the visual nature of the task, strongly influence the effectiveness of a mapping between hand measurements and software parameters. We show that this effect is visible both in overall user performance, and in the user's ability to coordinate the control of these parameters.

We describe two experiments on user perception and control of multi-touch interaction using one and two hands. The first experiment addresses how to maintain perceptual-motor compatibility in multi-touch interaction. The second measures separable and parallel control of degrees of freedom in the hands and fingers. Results indicate that user perception of a bimanual interaction task is compatible with the control of two points, but can be incompatible with the control of orientation and scale. An analogous unimanual task is compatible with the control of a position, orientation, and hand-span. Furthermore, we find a slight advantage in using two hands for separate control of two positions, while one and two-handed control exhibit the same degree of parallelism in an object manipulation task. We show how these results can be applied to the design of multi-touch interaction.

4.4.1 Experiment One

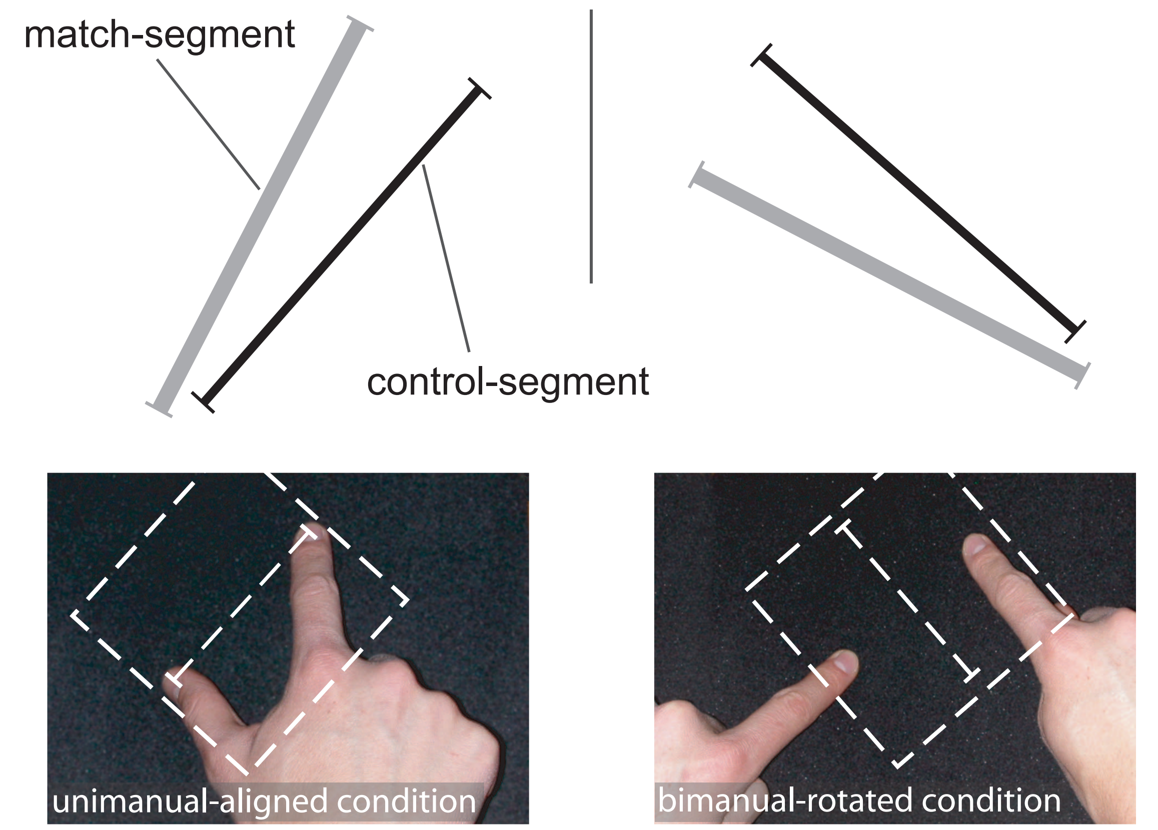

Figure 4.1: Visual display and mappings for the tracking task. Users were asked to track a moving segment using two fingers on one and two hands. For each hand condition, the segment was manipulated using both an aligned display (left) and a rotated display (right).

The goal of this experiment is to establish mappings that ensure compatibility between the user's finger movements and the visual feedback presented by the system. In particular we examine mappings for an object transportation and orientation task. We use a two-point object manipulation technique known as two handed "stretchies" that has appeared frequently in the literature [33, 70, 73, 105]. A one-handed equivalent has also been described [89, 93]. The technique allows a user to simultaneously translate, rotate, and scale an object. In the case of two fingers on a touchpad, each of the user's fingers controls the position of a point on the object. The transformation of the line segment connecting the user's fingers is applied to the manipulated object. Change in its length scales the object, change in its angle rotates the object, and change in its position moves the object.

We present participants with a segment tracking task similar to one previously used to study bimanual parallelism [12] (see Figure 4.1). Participants are asked to pursue a short line segment as it randomly moves and rotates on the screen by controlling a "match-segment" so that its position and orientation match the target-segment as closely as possible. This continuous pursuit task forces participants to coordinate their control of parameters as much as possible, allowing us to measure their coordination ability.

Participants manipulate the control-segment (using the two handed stretchies technique) as though it were a stripe drawn on a transparent sheet of rubber. They can manipulate this sheet using either one finger on each hand (*bimanual* condition), or the thumb and index finger of their right hand (*unimanual* condition). As discussed above, the movements of the fingers on one hand are highly correlated. Therefore we hypothesize the following:

H1 The unimanual manipulation condition will exhibit greater parallelism than the bimanual condition.

The manipulation is performed under two visual conditions: *aligned* and *rotated*. In the aligned condition, the control-segment is drawn so that its endpoints are aligned with the points controlled by the user's fingers (Figure 4.1 left). For the rotated condition the segment is drawn rotated 90° about the center of the aligned segment (Figure 4.1 right). In both visual conditions the motor control task is identical. Any finger motion would result in the same visual transformation under both conditions. However, we predict that alignment or lack of alignment with the user's fingers will have different effects in the one and two-handed conditions. If users perceive the task as control of position, orientation, and scale, then the alignment of the segment should have no effect on performance. We predict that this is the case in unimanual multi-touch interaction: Motor rotation is compatible with visual rotation, and motor extension of the fingers is compatible visual expansion. However, if users perceive the task as control of two points, then only the aligned condition will maintain perceptuomotor compatibility. In the rotated case, moving the left finger up will result in the leftmost endpoint moving to the right. Attempting to control points instead of orientation and scale makes the task more difficult. In light of this analysis, we make the following hypothesis:

H2 Presenting a rotated display of the match-segment will have no effect under the unimanual condition, but will significantly reduce performance in the bimanual condition (*i.e.,* increase tracking error).

Apparatus and Task Design

Participants interacted with the system using a FingerWorks iGesture Pad [39]. The touchpad measures 15.5 $\times$ 11.5 cm, and tracks finger contacts at approximately 100 Hz. The system made an absolute mapping between points on the touchpad and a 1024 $\times$ 746 pixel region on the screen at a control/display ratio of 0.55. The display was placed approximately 45 cm from the subject. For the unimanual condition the touchpad was placed in front of or in front and slightly to the right of the subject's right shoulder, while in the bimanual condition it was placed directly in front of the subject and screen. The display was updated at over 100 frames per second.

The match-segment was maintained at a length of 3 cm in touchpad space. The center of the segment was constrained to a 8.6 $\times$ 12.6 cm region of the touchpad, the angle of the aligned segment was constrained to lie between 0 and 86° from the horizontal. This range is accessible within the joint limits of both the bimanual and unimanual condition, and ensures that the left and right endpoints never cross. The path of the center of the segment was interpolated using a cubic interpolant through random points in the constrained region of the touchpad at a rate of 5 seconds per point. The angle was interpolated through random angles in the constrained range at a rate of 6.25 seconds per angle.

The match-segment was drawn with a gray seven pixel wide stroke with 14 pixel long tick marks at its endpoints. The control-segment was drawn with a two pixel wide black stroke. If more or less than two fingers were detected on the touchpad, tracking temporarily stopped, and the control-segment turned red to alert the subject. In the aligned condition, the control segment was drawn so that its endpoints corresponded to the mapped positions of the contact points on the touchpad. In the rotated condition, both segments were drawn rotated 90°about the center of the corresponding aligned segment.

Participants

Twelve right handed university students (6 women, 6 men) participated in Experiment One. All were regular computer users, but had no previous experience using a multipoint touchpad. They spent approximately 25 minutes performing the task and filling out a short questionnaire. Participants were paid $5.

Design and Measures

A within-subject full factorial design was used. The independent variables were the hand condition (unimanual and bimanual), and the visual presentation (aligned and rotated). Participants completed four 30 second tracking trials under each of the four condition for a total 8 minutes of tracking. The first trial in each condition was for practice. For the later three data-collection trials participants were asked to track the match-segment as closely as possible. The order of presentation of the four conditions was balanced according to a Latin square. The series of transformations used to generate the animation path for the match-segment was identical under all conditions.

Dependent variables were mean error and mean parallelism in each 30 second trial. Mean error was calculated as the mean sum of the distances between the endpoints of the control-segment and the endpoints of the match-segment. Note that this error is preserved under rotation, so we can use the segment endpoints in the rotated condition as well. Parallelism was calculated as a ratio of error-reduction as described above. As the touchpad sampling rate is somewhat variable, the data was resampled at 50 Hz. Segments where the user had too few or too many fingers in contact with the touchpad for more than 0.5 seconds were removed from the analysis, while shorter segments were linearly interpolated.

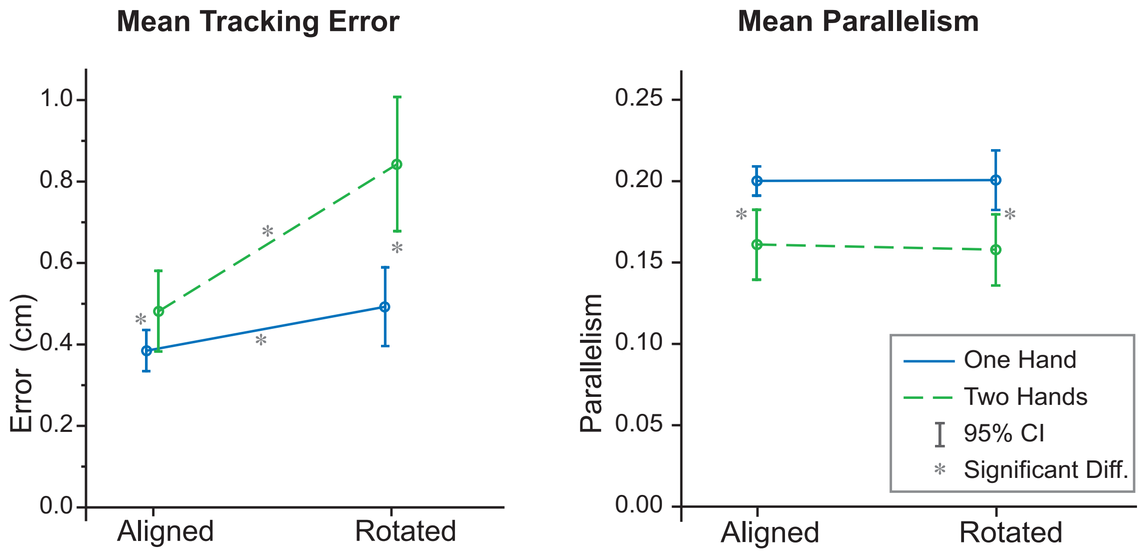

Figure 4.2: Results for Experiment One. Note that rotating the visual stimulus resulted in a large increase in error in the bimanual condition, but only a minor increase in the unimanual condition.

Results and Analyses for Experiment One

Results for parallelism can be seen in Figure 4.2 (right). An analysis of variance revealed a significant main effect for hand condition ($F_{1,11} = 34.16, p < 0.05$) but no effect for, or interaction with, visual presentation. The unimanual condition showed significantly more parallelism in both the aligned condition ($t_{11} = 3.69, p < 0.0125^*$) and the rotated condition ($t_{11} = 4.72, p < 0.0125^*$). This supports hypothesis *H1*, that one hand exhibits more parallel control than two. However, the difference is small, and the overall parallelism observed is low. The low parallelism value may indicate that an equal-rate reduction in percent error is not a strategy employed by our motor control system. We will explore this issue further in Experiment Two.

Results for tracking error are shown in Figure 4.2 (left). An analysis of variance revealed a significant main effect for hand condition ($F_{1,11} = 21.13, p < 0.05$) and for visual presentation ($F_{1,11} = 49.10, p < 0.05$), as well as an interaction between the two factors ($F_{1,11} = 40.96, p < 0.05$). Rotating the visual presentation of the segment resulted in a significant difference in error in both the bimanual condition

*Bonferonni correction for four comparisons at $\alpha = .05$.

($t_{11} = 7.78, p < 0.0125^{\dagger}$) and the unimanual condition ($t_{11} = 3.66, p < 0.025^{\dagger}$). While this does not meet our *H2* prediction that rotating the visual presentation will have no effect on the unimanual condition, the relative magnitudes of the changes in error do provide support for our hypothesis. In the unimanual condition, rotating the segment increased error by 28%, while in the bimanual condition it increased error by 75%. Thus, it is reasonable to surmise that, to a first approximation, control of a position, orientation, and span is perceptually compatible with unimanual manipulation, but is not compatible with bimanual manipulation in the absence of a clear finger-to-point correspondence. When such a correspondence exists, bimanual manipulation is compatible with the control of two positions.

Participant feedback appears to corroborate this view. Participants were asked if they found any aspect of the task particularly difficult. Commenting on the rotated bimanual condition one participant said that "it was as if the controls were reversed when rotating." Another said that this condition was the most difficult, and that she "found it hard for the two sides of my body to work together," and difficult to "fix my sight on the two invisible spots on the screen where my fingers 'were'." No such comments were made about the unimanual rotated condition.

We hypothesize that the small increase in error in the unimanual condition may be due to the fact that changing the span of the hand is an *oriented* expansion, rather than a uniform one. The interaction between hand span and orientation is important in grasping behavior. This relation to grasping was visible in one variation of our pilot study. When the system ignored the inter-finger distance and kept the segment length constant, we observed that participants brought their fingers much closer together in the rotated condition than in the aligned condition, as if they were attempting to hold the segment between their fingers.

A significant difference in error between one and two hands was seen in both the aligned condition ($t_{11} = 2.45, p < 0.05^{\dagger}$), and rotated condition ($t_{11} = 5.58, p < 0.0167^{\dagger}$). In the aligned conditions this represented a 25% increase in error. This appears to suggest that unimanual manipulation may be better suited for manipulation tasks that requires a high degree of coordination than bimanual manipulation. However, the fitness of one or two handed multi-touch techniques for a given task

†This test used Holm's sequential Bonferonni procedure [60] for four comparisons at $\alpha = .05$.

may have more to do with the structure and nature of the manipulation task and the particular degrees of freedom that require coordination. Our next experiment explores this issue further.

4.4.2 Experiment Two

The goal of this experiment is to assess the structure of one and two handed multi-touch interaction. In particular, we propose that in an object manipulation task, two hands are better able to isolate the control of two individual points than one hand. Furthermore, in the light of Experiment One, we expect that one hand would be better able to coordinate control of an object's position, orientation, and size.

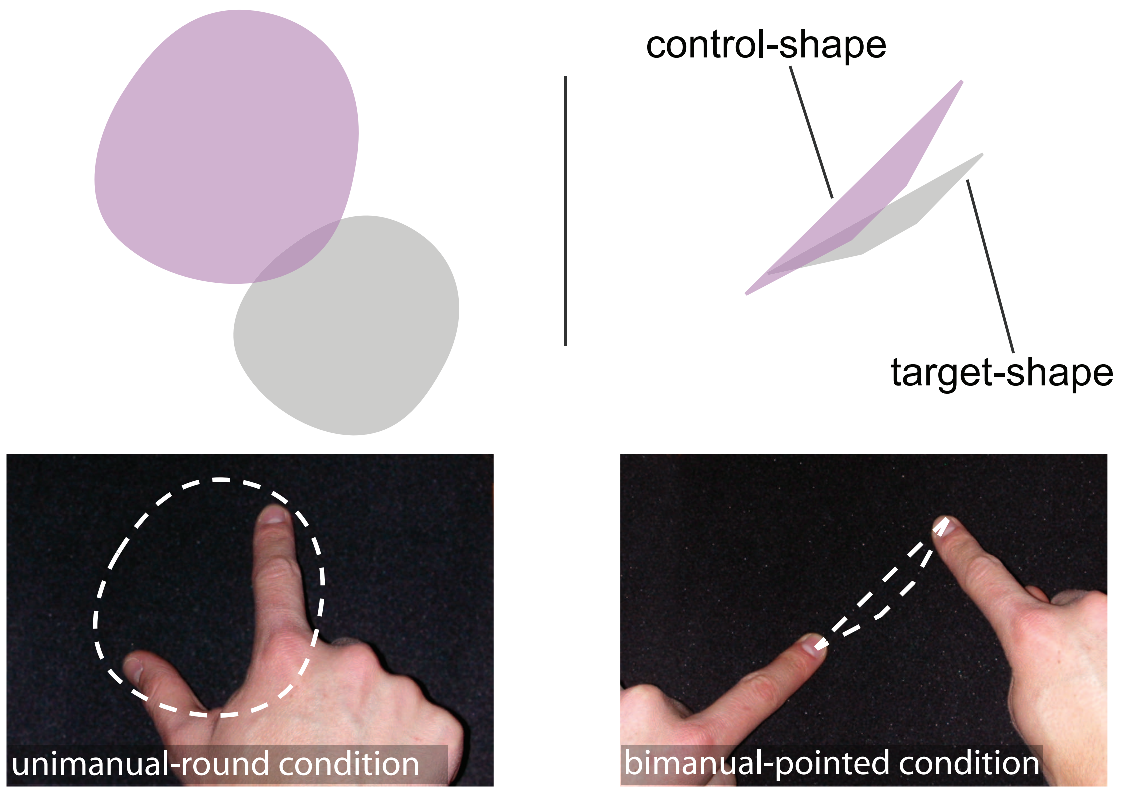

Figure 4.3: Visual display and mappings for the alignment task. Subjects were asked to align a control-shape to a congruent target-shape using two fingers on one and two hands. For each hand condition users manipulated both a round, featureless shape (left) and a thin, pointed shape who's key features were aligned with the subject's fingers (right).

Participants are presented with an object alignment task. Using the same two-point "stretchies" technique used in Experiment One, participants used two fingers on one or two hands to move, orient, and scale a control-shape so that it is aligned with a target-shape (see Figure 4.3). The experiment uses two types of shapes. The first is a thin, pointed shape with two prominent, sharp "features" at opposite ends (Figure 4.3 right). We believe that a clear alignment strategy for this shape is to align each prominent feature on the control-shape with the corresponding feature on the target-shape. We align the mapped position of the user's fingers on the touchpad so that they lie directly on the two feature points. This ensures that moving a single finger will only move its corresponding point, while leaving the opposite feature point fixed. Since we expect that separate control of two points is easier with two hands than one, we predict the following:

H3 Bimanual alignment of the pointed shape will be quicker than unimanual alignment.

H4 Bimanual alignment of the pointed shape will be more efficient than unimanual alignment (as measured by Zhai and Milgram's inefficiency metric).

The second shape is a smooth, round shape with no obvious features (Figure 4.3 left). Lacking such features a reasonable alignment strategy is to attempt to align the entire boundary of the shape. We expect that this strategy would benefit from a high degree of coordination between the adjusted dimensions, since separately adjusting the scale, position, or orientation, would throw-off previous adjustments. Thus, we make the following hypotheses:

H5 Unimanual alignment of the round shape will be quicker than bimanual alignment.

H6 Unimanual alignment of the round shape will be more efficient than bimanual alignment.

Apparatus and Task Design

The hardware and display setup were identical to those in Experiment One.

The control-shape was drawn in a translucent violet, keeping the target-shape (gray) always visible. If more or fewer than two fingers were in contact with the touchpad, tracking was temporarily stopped, and a red border was drawn about the control-shape to alert the subject. If every point on the boundary of the control-shape was within 1 mm (in touchpad coordination) of a point on the boundary of the target-shape, the control shape was considered aligned and was drawn in green. Maintaining alignment for 0.5 seconds ended the trial.

The center of the line segment connecting the target positions of the subject's fingers was randomly placed within a 3×2.5 cm rectangle in the center of the touchpad. The segment was oriented at a random angle between 0 and 80°from the horizontal, and was assigned a random length between 2.5 and 4 cm. The end position of each trial constituted the start position for the next trial.

Participants

Twelve right handed university students (5 women, 7 men) participated in Experiment Two. All were regular computer users, but had no previous experience using a multi-point touchpad. They spent approximately 30 minutes performing the task and filling out a short questionnaire. Participants were paid $5.

Design and Measures

A within-subject full factorial design was used. The independent variables were the hand condition (unimanual and bimanual), and the shape (pointed and round). Participants completed three sets of 20 alignment trials under each of the four conditions. The first set of trials in each condition was considered practice, as was the initial trial in each set. In the later two data collection trials participants were asked to work as fast as possible. The order of presentation of the four conditions was balanced according to a Latin square. The ordered series of transformations used to generate the target shapes was identical under all conditions.

Dependent variables were trial completion time and inefficiency (see section on coordination). Inefficiency was measured with respect to the path traveled by the two control points. Trials were discounted from the analysis if more or fewer than two fingers were in contact with the touchpad for more than one second, or for a distance of more than 1 cm. Trials longer than 10 seconds were also removed. (Discounted trials constituted 5% of the data.)

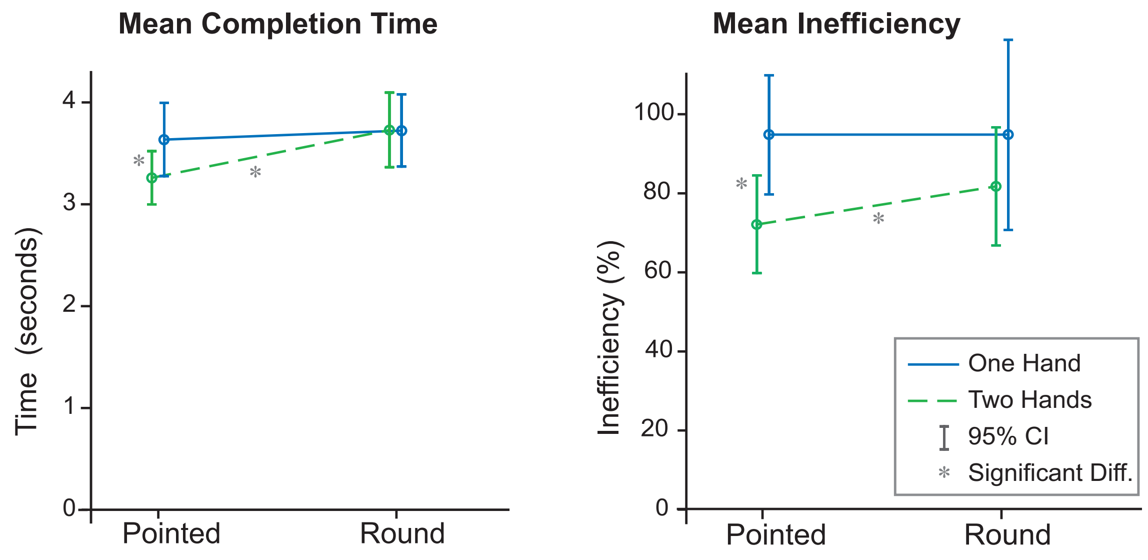

Figure 4.4: Results for Experiment Two. Subjects aligned the pointed shape slightly faster using two hands than one. This result is explained by the greater efficiency of bimanual control of two separate points than unimanual control.

Results and Analyses of Experiment Two

Completion times for Experiment Two are shown in Figure 4.4. An analysis of variance revealed a significant main effect for shape ($F_{1,11} = 5.44, p < 0.05$) as well as a significant interaction between hands and shape ($F_{1,11} = 8.84, p < 0.05$). In the bimanual condition users aligned the pointed shape significantly faster than the round shape ($t_{11} = 3.63, p < 0.0125^{\dagger}$). No such difference was found in the unimanual condition. Furthermore, users aligned the pointed shape significantly faster using two hands than using one ($t_{11} = 2.86, p < 0.0167^{\dagger}$). This confirms hypothesis *H3*. We interpret this to mean that users are better able to separate the control of two points when using fingers on opposing hands than when using fingers of the same hand.

Notably, no significant difference between hand conditions was found for the round shape. This contradicts hypothesis *H5* that one hand would perform faster for this shape. This could be interpreted in two ways. First, it is possible that the strategy participants used for aligning the round shape did not entail the high degree of coordination we expected. Alternatively, it is possible that two hands can coordinate the necessary degrees of freedom just as well as one. We look at the efficiency data to help resolve this issue.

Inefficiency for experiment two is shown in Figure 4.4. A significant main effect was found for hands ($\mathrm{F}_{1,11} = 11.24, p < 0.05$). When manipulating the pointed shape, two hands were significantly more efficient than one ($\mathrm{t}_{11} = 4.87, p < 0.0125^{\dagger}$). Two hands were also more efficient when manipulating the pointed shape than when manipulating the round shape ($\mathrm{t}_{11} = 3.02, p < 0.0125^{\dagger}$). No difference was found between one and two hands on the round shape. This confirms *H4*, but contradicts *H6*. That is, two hands were more efficient than one for the task requiring separation, but one hand was not more efficient for the task requiring coordination. Due to the significant positive correlation between inefficiency and completion times ($r = 0.598, p < 0.05$) we conclude that shorter completions times are due to greater efficiency.

While it is not surprising that two hands show a greater amount of coordination for the separable task, the results for the integral task appear to contradict Experiment One. In the first experiment, one hand displayed slightly more parallelism than two for a task that required a high degree of coordination. In the second experiment, no such difference was found. This may be attributed to several differences between the two experiments. First, Experiment One involved moving the center of the control-shape greater distances than in Experiment Two. This would result in greater parallelism for fingers with a close mechanical link. Furthermore, while in the first experiment, both fingers had to reduce absolute error at an approximately equal rate, the setup of the second experiment yielded a different start-to-goal distance for each finger. This may favor greater parallelism in a separable control structure.

It should also be noted that while both parallelism and efficiency are intended as measures of coordination, they do not measure precisely the same thing. However, analysis of the parallelism in Experiment Two revealed no difference in parallelism for the round shape, and more parallelism in the bimanual condition for the pointed shape ($\mathrm{t}_{11} = 3.63, p < 0.0125$).

4.4.3 Discussion and Future Work

Our experiments yielded two clear conclusions. First, unimanual multi-touch manipulation is compatible with a visual rotation task, even when lacking clear point correspondences. This implies that users perceive the unimanual task as control of a position, rotation, and a (possibly oriented) scale. Two handed multi-touch manipulation is only compatible with an object manipulation task when there is a clear correspondence between the fingers and the manipulated control points. This has a number of design implications. Unimanual orientation control is more robust to visual transformations than bimanual control. Thus, interfaces that involve rotation (*e.g.*, a dial) may be performed with one hand with less reliance on visual feedback. It also suggests that applying a gain function to object rotation [89] could be beneficial for a one-handed interaction technique, but, since it hinders a clear finger-to-point correspondence, it may result in confusion and reduced performance if a two-handed mapping is used.

The second clear result is that two hands perform better than one at tasks that require separate control of two points. Examples of these include window manipulation, marquee selection, image cropping, or control of separate objects. It is also the case that these task are perceptually compatible with bimanual control.

A number of open questions still remain. The cause of the small increase in error in the rotated unimanual condition is not yet clear. While we hypothesize that it is caused by an interaction between the orientation and span components of the manipulation, our experiment did not separate these two components. Further investigation of this issue may provide designers with a better model of user perception of one-handed multi-touch interaction. It is also not yet clear under what conditions one hand can coordinate the degrees-of-freedom of a manipulated object better than two. Our experiments indicate that this may be the case under certain conditions, and hint that the answer may have something to do with the scale or symmetry of the action. From the perspective of an interface designer, however, the question may not be of much practical value. If a task requiring a high degree of coordination is compatible with both one and two handed manipulation, but has no clear separation of control points, our results indicate that performance differences between one and two-handed multi-touch techniques are likely to be small. Under these conditions, a designer may safely interchange the two methods as best suits the task.

Chapter 5

Future Research Directions

Multi-touch interaction is still in the early stages of development. In this work we have shown a number of methods for using multi-touch input to improve the user experience. We have discussed the implications of these techniques, and how they may be extended. We have also began to explore the human factors involved in multi-touch interaction. Many more applications may benefit from multi-touch interaction, and questions remain about multi-touch interaction design. This chapter outlines future research directions along both of these paths.

5.1 Principles

5.1.1 Human Manipulation Abilities

The experiments described in chapter 4 address parameter mapping for two fingers on one hand, or one finger on each of two hands, but we have yet to generalize our findings to multiple fingers on a single hand, and to bimanual interaction using more than two fingers. Different pairs of fingers have differing degrees of independence, which makes them more or less suitable for various tasks. The role each finger plays may also depend on the type of manipulation being performed. While hand specialization is well explained by Guiard's kinematic chain model [53], it is not yet clear how the model can be extended to account for fingers, and whether such an extension would have any useful implications. The model asserts that any link higher up in the chain (and particularly the base link) acts as a reference frame for the lower links. However, the hands and fingers are not connected sequentially in a single chain, but form a branching network of kinematic chains. Do these structures share a single root, or does the base of the chain change to accommodate the task? The answers to these questions may have important design implications. They may help designers understand role assignment for different fingers, and determine how users respond to transformation in kinesthetic reference frames [11]. For example, when using the thumb and index finger, is the frame of reference for both digits always set by the hand, or can one finger act relative to the position of the other? These factors determine whether asymmetric bimanual interaction technique have usable unimanual analogues.

This work has focused on unimanual interaction using only two fingers, as they represent the major degrees-of-freedom of the hand. Work by Santello *et al.* has shown that about 80% of the variance in static hand postures is explained by only two principle components (both relating to opening and closing the grasp) [94], but the other 20% is not just noise. Another four or five components still provide useful, albeit subtle, information. Do these results transfer to finger motion along a surface? If so, is the information useful for interaction tasks? These additional dimensions appear to relate to shaping the grasp to conform to the shape of the target object. As such, they may be useful controlling the shape of regions-of-interest. Fluid control of non-trivial regions-of-interest has applications to manipulation of dense or continuous data, such as images, curves, and surfaces, and to selection and control of multiple objects.

Another way of providing more control for the user is through motor-learning. This work has focused mainly on closed-loop interactions that are limited by the user's attention and working memory. However, people are capable of learning, through practice, intricate motor-control patterns such as handwriting, typing, and the playing of musical instruments. This ability may prove useful for multi-touch interaction techniques designed with expert users in mind. For example, multi-touch fingerspelling can provide implement-free text input that is potentially faster than single-point stroked alphabets. The greater number of motions possible with multiple fingers eliminates the need for complex strokes, or for multi-stroke characters.

5.1.2 Importance of Haptic Feedback in Multi-touch Interaction

Fitzmaurice, Ishii, and Buxton [41] have demonstrated that tangible interfaces, such as their *Bricks* system, have many advantages over traditional interfaces that are mediated through a generic input device such as a mouse. Among these advantages are the encouragement of two-handed interaction, allowance for parallel input and spatially multiplexed input (rather than sequential and temporally multiplexed input), making use of our well developed manipulation skills, taking advantage of our spatial reasoning abilities, and support for multi-user collaboration. A great part of these advantages appears to be a consequence of allowing users to interact with the system using multiple fingers on both hands. If this is the case, then a multi-touch interfaces that digitally simulate tangible UIs may allow for user performance that is on par with their tangible analogues. This is especially the case for the prevailing table-top style of interaction, where physical widgets are often restricted to sliding on a surface. A large number of such simulations could be run on a single multi-touch system without the need to acquire and store a large number of special-purpous input devices. Tangible UI devices may be costly, and may clutter the user's workspace, so it is important to asses the value of their palpability.

Our fingers are not simply manipulators, they are also used to sense properties of the grasped object. Our sense of touch plays a crucial role in our prehensile behavior [82], and without it we become more reliant on our visual perception. Designing experiments to test our ability to compensate for the absence of tactile feedback is fairly straightforward. For example, comparisons could be made of real and simulated tangible interface techniques that are commonly used (*e.g.,* sliding physical objects on a table). Since physical objects may interact with each other, a physical simulation may be needed to accurately emulate the behavior of the system. Performance indicators, such as timing and error, can than be measured for simple sub-tasks such as target acquisition and path-tracking. Attention should also be paid to the effects of inter-object contact. For example, aligning two objects to each other, or sliding an object into a slot or socket could potentially be easier given appropriate haptic feedback.

5.2 Applications

5.2.1 3D Manipulation

A large part of computer interaction development has focused on interaction which takes place on our two dimensional screens. But the world is not flat, and to properly represent it requires interaction in three dimensions. 3D interaction is crucial for animation, computer-aided design, and scientific visualization applications. Moving from two to three dimensions does not add only a single parameter. For example, positioning and orienting an object in 3D requires the control of twice as many parameters as are needed in 2D. As a consequence of this, software for 3D manipulation is often brimming with numerous control widgets and manipulation modes. To address this problem several specialized three- and six-degree of freedom input devices have been created [2,9,44,58,130]. While these may work well for 3D manipulation, they do not perform as well in standard 2D interaction tasks that have been designed with a 2D device in mind. A multi-point touchpad, however, is as flat as the screen, and can easily be used for 2D interaction, yet the large number of degrees-of-freedom it provides makes it a good candidate to control the many parameters of 3D environments.

The problem is finding an appropriate mapping. The structure of 3D manipulation is very different from the control structure provided by multi-point touchpads. A number of systems have explored mappings from 2D points on the screen to 3D manipulation [49, 124], and while their usability has not been formally tested, they show that this may be a promising direction. As the benefit of integrated 2D and 3D input is great, further exploration of multi-touch 3D control may be very fruitful.

Ideally, a 3D manipulation technique would allow a user to simultaneously control the six degrees-of-freedom of rigid-body motion (object position and orientation). Theoretically, three input points should be sufficient for this task. For example, such control could be achieved using a similar approach to Gleicher and Witkin's through-the-lens camera control [49]. A user would touch the screen (or use a hand cursor) so that her fingers touch the screen-image of the object of interest. Constraint optimization could then be used to solve for a rigid transformation that maintains the projected finger positions over the same spots on the manipulated object. Such a mapping, however, may require frequent clutching for rotation, as the user may only manipulate points on the front-facing surface of the object. However, as discussed in chapters 3 and 4, the motions of the fingers are highly correlated, so simultaneous independent control of three fingers on one hand may be not be feasible. If one-handed control is desired, it may be practical to separate the manipulation into control of position and of orientation. The work of Wang *et al.* suggests that there exist separate motor control pathways for object transportation and orientation, so they may be executed either simultaneously or sequentially.

Experiments comparing a three DOF input device to a standard mouse have shown that users perform a 3D docking task 30% to 40% faster when given simultaneous control over all three spatial dimensions (even given a lower resolution and lack of S-R compatibility in the additional dimension) [9]. It is reasonable to expect similar improvements using a multi-touch technique if an appropriate mapping is found. As user feedback regarding the mapping of the similarity cursor (chapter 3) was very positive, we suggest using a similar mapping for 3D positioning. Motion of two fingers on a touchpad controls relative position along the film plane, while the the change in distance between the fingers controls relative position along the look vector. Bringing the fingers together pushes the object away, while pulling them apart draws the object closer. While using hand-span may not be strictly perceptually compatible with distance along the look vector, it is at least mnemonically compatible, since increasing the hand-span increases the size of the object's projection on the screen.

Two fingers can also be used for object rotation. Standard single-point rotation techniques such as the Arcball and Virtual Sphere have a number of limitations [30, 97]. For example, the available axes of rotation depend on where in the interaction area the user first clicks to start the rotation. Thus, a user cannot always achieve the desired rotation without repeated clutching. A simple multi-touch mapping could greatly reduce the need for clutching. For example, using two fingers on a touchpad, motion along X and Y could control rotation about the X and Y axes, while twisting the fingers about their centroid would control rotation about Z. Trackball-style techniques require the user to position the cursor outside the trackball to control rotation about Z, thereby eliminating rotations about the other axis. Using the above multi-touch mapping all degrees-of-freedom are available at any time.

We have implemented a prototype of the above multi-touch interface, and collected some preliminary user feedback. Our prototype used the number of fingers in contact with the touchpad as a mode switch: two fingers controlled translation, while three controlled rotation. In informal trials, users found the position control technique easy to learn, and were able to move an object in three dimensions simultaneously. The rotation technique took a bit longer to learn, and had a number of design issues. For example, it can be difficult to rotate an object about Z without affecting the other dimensions, which users found very confusing. Attenuating X and Y rotations when large rotations were made about Z alleviated this problem. Preliminary testing suggests that users experienced with 3D software performed somewhat faster using the multi-touch interface than using an Arcball technique. However, novice users performed slightly better with the Arcball. While these initial results are promising, there is still much room for improving the design, and for studying its benefits and shortcomings.

Multi-touch 3D interaction has uses beyond simply positioning objects. The ability to select and manipulate objects or control-widgets in three dimensions has applications for surface editing, sculpting, and character posing. A similar interface may also be used for camera control. By following the principles of Guiard's kinematic chain model, a multi-touch UI which assigns camera control to the non-dominant hand and object manipulation to the dominant hand could have large performance gains over a one-handed or modal interface [13, 59].

5.2.2 Multi-touch Instruments

The *Instrumental interaction* model extends the principles of direct manipulation. It is a powerful means for assessing and designing interaction techniques. Beaudouin-Lafon defines an *interaction instrument* as "a mediator or two-way transducer between the user and domain objects." For example, a scrollbar transforms mouse motion into scrolling commands. Instruments can be described in terms of their degree of indirection (*e.g.*, spatial or temporal offset), degree of integration between the control parameters, and degree of compatibility between the actions of the user and the system's reaction.

In chapter 3 we discuss a specific type of interaction instrument: the cursor. A cursor transforms raw measurements of the physical world into software parameters. However, most instruments do not mediate directly between the user and the object of interest; they mediate between the *cursor* and the object. Thus, multi-touch cursors make possible new types of interaction instruments. The similarity cursor allows for interaction widgets that can be twisted or squeezed. Designers can use such widgets to create instruments with a greater degrees of integration or compatibility than their single-point analogs. Multi-touch instruments also allow for greater parallelism in input, which can be helpful in difficult high-DOF control tasks. For example, tone mapping an image, or manipulating its hue, saturation, and brightness, requires a great deal of trial-and-error, since adjusting one parameter affects how the others are perceived. A multi-touch photo-enhancement tool would allow for rapid exploration of the image space.

High-DOF control also makes physically inspired instruments more feasible. Imbuing graphical objects with simple physical properties, such as collision response, can create a rich-set of new affordances for manipulating them [3]. For example, a simulated straightedge in a drawing or presentation program can be used for aligning objects, or for sweeping them aside [92]. Similar tools could be designed for cutting, grouping, or constraining the motion of objects.

5.2.3 Other Applications and Interaction Techniques

Multi-point touchpads are extremely versatile general-purpous input devices. Not only do they allow for coordinated high-DOF input, they can also seamlessly integrate with standard single-point input techniques. However, if we are to replace the mouse with a multi-point touchpad, it is important to study touchpad use for standard pointing. It has been shown that when multiple fingers work in concert to control a physical implement (such as a stylus or mouse) they provide greater throughput than the use of a single finger. Can multiple fingers on a touchpad control a single point more effectively than one? Is it possible to design a virtual stylus?

While multi-point touchpads are far from ubiquitous, many of the techniques described in this dissertation rely only on the position, orientation, and span of the hand. As such, they may be usable with devices other than multi-point digitizers. For example, standard capacitive touchpads can detect a bounding-box of the contact region, which may be used to calculate these properties. Multi-touch input could also be produced by juxtaposing two single-point touchpads in various configurations. One configuration that may be especially well suited for hand-held devices is to place a touchpad on the back of the device and a touch-screen on the front [96, 116]. Our techniques also create space for new hardware designs, such as mice that sense orientation and finger extension.

There are many more applications that may benefits from coordinated high-degree-of-freedom control. For example, curve editing on a computer is still a difficult task that could be simplified through parallel input [74]. Multi-touch interaction could also benefit 2D and 3D navigation. For example, simultaneous control of zooming and panning could improve navigation of maps, documents, and other 2D data presentations [7, 62]. Multi-touch interaction may also be useful for specifying spatial and temporal relationships of multiple agents. Example applications include stage and film direction, dance choreography, sports coaching, and all manner of logistics.

Chapter 6

Conclusions

6.1 Summary of Contributions

This dissertation has presented two types of contributions. First, we showed that multi-touch interaction is useful for a variety of interaction tasks. Novel techniques, such as those presented here, allow people to easily accomplish tasks that were formerly difficult or even impossible. For example, the multi-touch deformation interface lets novice users create expressive animations. Other techniques, such as the multi-touch cursors, make interaction tasks simpler and more fluid by reducing the need for many interaction modes or widgets. Second, this dissertation has increased our understanding of the human factors involved in multi-touch interaction. This knowledge will help interface designers better understand what tasks are suitable for multi-finger interaction, and how to best assign parameter control to the degrees of freedom of the user's hands.

6.1.1 Human Abilities in Multi-touch Interaction

Our experimental results establish multi-touch interaction as a viable means of coordinated multi-parameter input. They show that user performance using multi-touch methods is on a par with symmetric bimanual interaction techniques. These two-handed techniques have been shown to be superior to their one-handed counterparts, as they increase parallelism, reduce mode-switching time [73], and help users reason about a task [58,78]. Unimanual multi-touch techniques share these advantages, while leaving the other hand free to control other parameters, or execute other tasks (*e.g.*, apply keyboard modifiers).

We have made new discoveries regarding the control structures and perceptual compatibility of both bimanual interaction and unimanual multi-touch interaction. Specifically, we have found that user action in a bimanual interaction task is perceptually compatible with the control of two points, while an analogous unimanual task is compatible with control of a position, orientation, and hand-span. We have also discovered that users are slightly better at separating the control of two positions when using two hands than when using two fingers on one. However, when the motion of the two hands or fingers must be highly coordinated, this advantage disappears.

This knowledge is useful for future multi-touch interaction design. Since the relationship of human cognitive processes with the physiology of the arms and hands is complex, it may lead to behavior that contradicts our intuitive understanding of how our minds and bodies function. Because of this, user interfaces often fail to exhibit the benefits their designers expected. A better understanding of human abilities can help eliminate poor design choices. For example, a designer may create a method for two-handed object rotation that fails to provide a clear point correspondence between the fingers and the object. We now know that such a mapping may lead to user confusion and poor performance, since bimanual interaction is only compatible with object rotation when a such a correspondence exits.

6.1.2 Multi-touch Interaction Techniques

We have presented a variety of novel interaction techniques for graphical manipulation. While each technique is useful in itself, together they serve to outline the types of tasks and techniques that are well suited to multi-touch interaction. Similar methods may be useful in other applications as well.

We have introduced the concept of a multi-touch cursor as an instrument for enhancing our natural abilities. We provided three examples of how such cursors can be designed and used, and discussed further directions for cursor design. The cursor techniques we described allow for parallel input rather than serial (*e.g.*, simultaneous translation and rotation) and reduce the need for separate interaction modes (*e.g.,* by integrating object grouping and manipulation into a single step). These and similar techniques have immediate use in illustration and presentation software.

Our multi-touch deformation technique illustrates how increasing the number of input points yields not just a quantitative change, but a qualitative improvement in the expressiveness of the interaction. It not only saves expert animators time in creating key-frames, but allows both expert and novice to perform real-time animation for communicating temporal concepts. It shows that multi-touch interaction is useful not just for human-computer communication, but for human-human communication as well.

6.2 Limitations and Open Issues

Several questions regarding our techniques, and multi-touch interaction in general, remain unanswered. While it is clear the cursor techniques reduce interface complexity and increase fluency, it is not manifest that these effects lead to better performance or user satisfaction. The literature on bimanual interaction suggests that increased parallelism and "chunking" may lead to these effects under the right conditions, but the utility of our techniques begs for empirical verification. In particular, the adjustable area cursor may lead to increased cognitive overhead in cluttered areas, which may lead to poor performance. Another outstanding issue is the usability of the three-state touchpad for precise manipulation tasks. The touchpad switch is activated by the same muscle groups that control the cursor. The added tension to the fingers may reduce precision, or lead to other types of interference.

6.3 Closing Remarks

Many linguists believe that language evolved from manual gestures [31]. Yet despite our aptitude in using spoken language, we still use our hands to communicate. The role of the computer has shifted from number-crunching machine to communication device, but the part played by hand movements in communicating information to a computer and in computer-mediated communication, has largely been limited to that of a single pointing finger. This dissertation has established the thesis that multi-touch interaction allows users to communicate information to a computer faster and more fluently than single-point interaction techniques. We have described a number of novel multi-touch interaction techniques, and provided guidelines for future development. As computing devices permeate an ever-growing portion of our lives, new interaction methodologies must rise to meet challenges that lie beyond the realm of today's mouse-and-keyboard paradigm. We believe that the versatility of multi-touch interaction will make it an integral part of future interfaces.

Bibliography

[1] 3Dconnexion, a Logitech company. Spaceball.

[2] 3Dconnexion, a Logitech company. Spacemouse.

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

[4] David Ahlstroem, Rainer Alexandrowicz, and Martin Hitz. Improving menu interaction: A comparison of standard, force enhanced and jumping menus. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1067–1076, New York, NY, USA, 2006. ACM Press.

[5] Bengt Ahlström, Sören Lenman, and Thomas Marmolin. Overcoming touchscreen user fatigue by workplace design. In *CHI '92: Posters and short talks of the 1992 SIGCHI conference on Human factors in computing systems*, pages 101–102, New York, NY, USA, 1992. ACM Press.

[6] Pär-Anders Albinsson and Shumin Zhai. High precision touch screen interaction. In *CHI '03: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 105–112, New York, NY, USA, 2003. ACM Press.

[7] Caroline Appert and Jean-Daniel Fekete. Orthozoom scroller: 1d multi-scale navigation. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 21–30, New York, NY, USA, 2006. ACM Press.

[8] R. Baecker. Genesys. In *Proceedings of the AFIPS Spring Joint Computer Conference*, 1969.

[9] Ravin Balakrishnan, Thomas Baudel, Gordon Kurtenbach, and George Fitzmaurice. The rockin'mouse: integral 3d manipulation on a plane. In *CHI '97: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 311–318, New York, NY, USA, 1997. ACM Press.

[10] Ravin Balakrishnan, George Fitzmaurice, Gordon Kurtenbach, and William Buxton. Digital tape drawing. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 161–169, New York, NY, USA, 1999. ACM Press.

[11] Ravin Balakrishnan and Ken Hinckley. The role of kinesthetic reference frames in two-handed input performance. In *UIST '99: Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 171–178, New York, NY, USA, 1999. ACM Press.

[12] Ravin Balakrishnan and Ken Hinckley. Symmetric bimanual interaction. In *CHI '00: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 33–40, New York, NY, USA, 2000. ACM Press.

[13] Ravin Balakrishnan and Gordon Kurtenbach. Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In *CHI '99: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 56–62, New York, NY, USA, 1999. ACM Press.

[14] Ravin Balakrishnan and I. Scott MacKenzie. Performance differences in the fingers, wrist, and forearm in computer input control. In *CHI '97: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 303–310, New York, NY, USA, 1997. ACM Press.

[15] Thomas Baudel and Michel Beaudouin-Lafon. Charade: remote control of objects using free-hand gestures. *Commun. ACM*, 36(7):28–35, 1993.

[16] Michel Beaudouin-Lafon. Instrumental interaction: An interaction model for designing post-wimp user interfaces. In *CHI '00: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 446–453, New York, NY, USA, 2000. ACM Press.

[17] Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. Precise selection techniques for multi-touch screens. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1263–1272, New York, NY, USA, 2006. ACM Press.

[18] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings UIST '93*, pages 73–80. ACM Press, 1993.

[19] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings of SIGGRAPH '93*, pages 73–80, New York, NY, USA, 1993. ACM Press.

[20] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *CHI '04: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 519–526, New York, NY, USA, 2004. ACM Press.

[21] Gábor Blaskó and Steven Feiner. Single-handed interaction techniques for multiple pressure-sensitive strips. In *CHI '04: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1461–1464, New York, NY, USA, 2004. ACM Press.

[22] Bodenheimer, B. *et. al.* The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97*, pages 3–18, September 1997.

[23] M. Bohan, S.G. Thompson, D.S. D.S. Scarlett, and A. Chaparro A. Gain and target size effects on cursor-positioning time with a mouse. In *Human Factors and Ergonomics Society 47th Annual Meeting*, pages 737–740, 2003.

[24] W. Buxton and B. Myers. A study in two-handed input. In *Proceedings of CHI '86*, pages 321–326, New York, NY, USA, 1986. ACM Press.

[25] William Buxton. A three-state model of graphical input. In *Proceedings of INTERACT '90*, pages 449–456. North-Holland, 1990.

[26] William Buxton, Ralph Hill, and Peter Rowley. Issues and techniques in touch-sensitive tablet input. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 215–224, New York, NY, USA, 1985. ACM Press.

[27] William Buxton, Ralph Hill, and Peter Rowley. Issues and techniques in touch-sensitive tablet input. In *Proceedings of SIGGRAPH '85*, pages 215–224, New York, NY, USA, 1985. ACM Press.

[28] William A. S. Buxton. Chunking and phrasing and the design of human-computer dialogues. In *Human-computer interaction: toward the year 2000*, pages 494–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[29] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A morphological analysis of the design space of input devices. *ACM Trans. Inf. Syst.*, 9(2):99–122, 1991.

[30] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-d rotation using 2-d control devices. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 121–129, New York, NY, USA, 1988. ACM Press.

[31] Michael C. Corballis. *From Hand to Mouth: The Origins of Language*. Princeton University Press, 2002.

[32] James L. Crowley and Joelle Coutaz. Vision for man machine interaction. In *EHCI*, pages 28–45, 1995.

[33] Lawrence D. Cutler, Bernd Frolich, and Pat Hanrahan. Two-handed direct manipulation on the responsive workbench. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 107–ff. ACM Press, 1997.

[34] Richard C. Davis and James A. Landay. Informal animation sketching: Requirements and design. In *Proceedings of AAAI 2004 Fall Symposium on Making Pen-Based Interaction Intelligent and Natural*, October 2004.

[35] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of UIST 2001*, pages 219–226. ACM Press, 2001.

[36] R. F. Dillon, Jeff D. Edey, and Jo W. Tombaugh. Measuring the true cost of command selection: techniques and results. In *CHI '90: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 19–26, New York, NY, USA, 1990. ACM Press.

[37] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. *ACM Trans. Graph.*, 22(3):409–416, 2003.

[38] Chris Esposito, W. Bradford Paley, and JueyChong Ong. Of mice and monkeys: a specialized input device for virtual body animation. In *SI3D '95: Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 109–ff., New York, NY, USA, 1995. ACM Press.

[39] FingerWorks. iGesture Pad.

[40] P. M. Fitts and C. M. Seeger. S-r compatibility: spatial characteristics of stimulus and response codes. *Journal of Experimental Physiology*, 47:381–391, 1953.

[41] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *Proceedings of CHI '95*, pages 442–449, New York, NY, USA, 1995. ACM Press.

[42] Clifton Forlines and Chia Shen. Dtlens: multi-user tabletop spatial data exploration. In *UIST '05: Proceedings of the 18th annual ACM symposium on User*

interface software and technology, pages 119–122, New York, NY, USA, 2005. ACM Press.

[43] Clifton Forlines, Daniel Vogel, and Ravin Balakrishnan. Hybridpointing: Fluid switching between absolute and relative pointing with a direct input device. In *UIST*, New York, NY, USA, 2006. ACM Press.

[44] Bernd Froehlich, Jan Hochstrate, Verena Skuk, and Anke Huckauf. The globefish and the globemouse: two new six degree of freedom input devices for graphics applications. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 191–199, New York, NY, USA, 2006. ACM Press.

[45] Tinsley A. Galyean and John F. Hughes. Sculpting: an interactive volumetric modeling technique. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 267–274, New York, NY, USA, 1991. ACM Press.

[46] Wendell R. Garner. *The Processing of Information and Structure.* Lawrence Erlbaum, Potomac, MD, USA, 1974.

[47] Yotam Gingold, Philip Davidson, Jefferson Han, and Denis Zorin. A direct texture placement and editing interface. In *Proceedings of UIST*, New York, NY, USA, 2006. ACM Press.

[48] Michael Gleicher. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff., New York, NY, USA, 1997. ACM Press.

[49] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 331–340, New York, NY, USA, 1992. ACM Press.

[50] D. Goryn and S. Hein. On the estimation of rigid body rotation from noisy data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1219–1220, 1995.

[51] T. Grossman, D. Wigdor, and R. Balakrishnan. Multi-finger gestural interaction with 3d volumetric displays. In *Proceedings of UIST '04*, pages 61–70. ACM Press, 2004.

[52] Tovi Grossman and Ravin Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of CHI '05*, pages 281–290, New York, NY, USA, 2005. ACM Press.

[53] Y. Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, pages 485–517, 1987.

[54] C. Hager-Ross and M.H. Schieber. Quantifying the independence of human finger movements: Comparisons of digits, hands, and movement frequencies. *Journal of Neuroscience*, 20(22):8542–8550, 2000.

[55] Perttu Hämäläinen, Mikko Lindholm, Ari Nykänen, and Johanna Höysniemi. Animaatiokone: an installation for creating clay animation. In *CHI '04: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 17–24, New York, NY, USA, 2004. ACM Press.

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

[57] Ken Hinckley, Mary Czerwinski, and Mike Sinclair. Interaction and modeling techniques for desktop two-handed input. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 49–58, New York, NY, USA, 1998. ACM Press.

[58] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. Passive real-world interface props for neurosurgical visualization. In *CHI '94: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 452–458, New York, NY, USA, 1994. ACM Press.

[59] Ken Hinckley, Randy Pausch, Dennis Proffitt, James Patten, and Neal Kassell. Cooperative bimanual action. In *CHI '97: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 27–34, New York, NY, USA, 1997. ACM Press.

[60] S. Holm. A simple sequentially rejective multiple test rocedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

[61] T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial keyframing for performance-driven animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 107–115, New York, NY, USA, 2005. ACM Press.

[62] Takeo Igarashi and Ken Hinckley. Speed-dependent automatic zooming for browsing large documents. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 139–148, New York, NY, USA, 2000. ACM Press.

[63] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.

[64] Measurand Inc. Shapetape.

[65] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and Jr. M. Preston Mullen. Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact.*, 1(1):3–26, 1994.

[66] H.D. Jellinek and S. K. Card. Powermice and user performance. In *Proceedings of CHI '90*, pages 213–220, New York, NY, USA, 1990. ACM Press.

[67] Paul Kabbash and William A. S. Buxton. The "prince" technique: Fitts' law and selection using area cursors. In *Proceedings of CHI '95*, pages 273–279, New York, NY, USA, 1995. ACM Press.

[68] Stelios Kourakis. On gestural interaction and modular gestures. Technical report, Universitat Pompeu Fabra, 2004.

[69] Myron W. Krueger, Thomas Gionfriddo, and Katrin Hinrichsen. Videoplace: An artificial reality. In *Proceedings of CHI '85*, pages 35–40, New York, NY, USA, 1985. ACM Press.

[70] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The design of a GUI paradigm based on tablets, two-hands, and transparency. In *CHI '97: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 35–42. ACM Press, 1997.

[71] Celine Latulipe. Symmetric interaction in the user interface. In *UIST 2004 Companion Proceedings*. ACM Press, 2004.

[72] Celine Latulipe, Ian Bell, Charlie L. A. Clarke, and Craig S. Kaplan. symtone: Two-handed manipulation of tone reproduction curves. In *Proceedings of Graphics Interface 2006*, 2006.

[73] Celine Latulipe, Craig S. Kaplan, and Charles L. A. Clarke. Bimanual and unimanual image alignment: An evaluation of mouse-based techniques. In *Proceedings of UIST '05*, pages 123–131, New York, NY, USA, 2005. ACM Press.

[74] Celine Latulipe, Stephen Mann, Craig S. Kaplan, and Charlie L. A. Clarke. symspline: Symmetric two-handed spline manipulation. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 349–358, New York, NY, USA, 2006. ACM Press.

[75] Joseph J. LaViola. A survey of hand posture and gesture recognition techniques and technology. Technical Report CS99-11, Brown University, Department of Computer Science, 1999.

[76] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[77] SK Lee, William Buxton, and K. C. Smith. A multi-touch three dimensional touch-sensitive tablet. In *Proceedings of CHI '85*, pages 21–25, New York, NY, USA, 1985. ACM Press.

[78] Andrea Leganchuk, Shumin Zhai, and William Buxton. Manual and cognitive benefits of two-handed input: an experimental study. *ACM Transactions on Human Computer Interaction*, 5(4):326–359, 1998.

[79] Julien Letessier and François Bérard. Visual tracking of bare fingers for interactive surfaces. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 119–122, New York, NY, USA, 2004. ACM Press.

[80] Jun Liu, David Pinelle, Samer Sallam, Sriram Subramanian, and Carl Gutwin. Tnt: improved rotation and translation on digital tables. In *GI '06: Proceedings of the 2006 conference on Graphics interface*, pages 25–32, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[81] I. Llamas, B. Kim, J. Gargus, J. Rossignac, and C.D. Shaw. Twister: A space-warp operator for the two-handed editing of 3d shapes. *ACM Trans. Graph.*, 22(3):663–668, 2003.

[82] C. Mackenzie and T. Iberall. *The Grasping Hand*. North Holland, 1994.

[83] I. Scott MacKenzie and Aleks Oniszczak. A comparison of three selection techniques for touchpads. In *Proceedings of CHI '98*, pages 336–343, New York, NY, USA, 1998. ACM Press.

[84] S. Malik, A. Ranjan, and R. Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *Proceedings of UIST '05*, pages 43–52. ACM Press, 2005.

[85] Shahzad Malik and Joe Laszlo. Visual touchpad: A two-handed gestural input device. In *Proceedings of ICMI '04*, pages 289–296. ACM Press, 2004.

[86] Microsoft. Pointer ballistics for windows xp, 2005.

[87] Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. Cooperative gestures: Multi-user gestural interactions for co-located groupware. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1201–1210, New York, NY, USA, 2006. ACM Press.

[88] Tomer Moscovich and John F. Hughes. Animation sketching: An approach to accessible animation. Technical report, Brown University, 2003.

[89] Tomer Moscovich and John F. Hughes. Multi-finger cursor techniques. In *GI '06: Proceedings of the 2006 conference on Graphics interface*, pages 1–7, Quebec, Canada, 2006.

[90] Tom Ngo, Doug Cutrell, Jenny Dana, Bruce Donald, Lorie Loeb, and Shunhui Zhu. Accessible animation and customizable graphics via simplicial configuration modeling. *Proceedings of SIGGRAPH 2000*, pages 403–410, July 2000.

[91] Russell Owen, Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. When it gets more difficult, use both hands: Exploring bimanual curve manipulation. In *GI '05: Proceedings of the 2005 conference on Graphics interface*, pages 17–24, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[92] Roope Raisamo and Kari-Jouko R. Design and evaluation of the alignment stick. *Interacting with Computers*, 12(5):483–506, 2000.

[93] Jun Rekimoto. Smartskin: An infrastructure for freehand manipulation on interactive surfaces. In *Proceedings of CHI 2002*, pages 113–120. ACM Press, 2002.

[94] Marco Santello, Martha Flanders, and John F. Soechting. Postural hand synergies for tool use. *The Journal of Neuroscience*, 18:10105–10115, 1998.

[95] Marc H. Schieber and Marco Santello. Hand function: Peripheral and central constraints on performance. *Journal of Applied Physiology*, 6:2293–2300, 1996.

[96] Carsten Schwesig, Ivan Poupyrev, and Eijiro Mori. Gummi: a bendable computer. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 263–270, New York, NY, USA, 2004. ACM Press.

[97] Ken Shoemake. Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of the conference on Graphics interface '92*, pages 151–156, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[98] H. Simon. How big is a chunk? *Science*, 183:482–488, 1974.

[99] David J. Struman. *Whole-hand Input.* PhD thesis, Massachusetts Institute of Technology, 1992.

[100] David J. Sturman. Computer puppetry. *IEEE Comput. Graph. Appl.*, 18(1):38–45, 1998.

[101] Tactiva Inc. TactaPad.

[102] S. C. L. Terra and R. A. Metoyer. Performance timing for keyframe animation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 253–258, New York, NY, USA, 2004. ACM Press.

[103] F. Thomas and O. Johnson. *Disney Animation: The Illusion of Life.* New York, Abbeville, 1984.

[104] M. Thorne, D. Burke, and M. van de Panne. Motion doodles: An interface for sketching character motion. In *Proceedings of SIGGRAPH 2004*, 2004.

[105] Brygg Ullmer and Hiroshi Ishii. The metadesk: Models and prototypes for tangible user interfaces. In *Proceedings of UIST '97*, pages 223–232. ACM Press, 1997.

[106] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[107] C.J. Veenman, E.A. Hendriks, and J.J.T. Reinders. A fast and robust point tracking algorithm. In *1998 International Conference on Image Processing*, volume 22, pages 653–657, 1998.

[108] Kevin Vlack, Terukazu Mizota, Naoki Kawakami, Kazuto Kamiyama, Hiroyuki Kajimoto, and Susumu Tachi. Gelforce: a vision-based traction field computer interface. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1154–1155, New York, NY, USA, 2005. ACM Press.

[109] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42, New York, NY, USA, 2005. ACM Press.

[110] Yanqing Wang and Christine L. MacKenzie. Object manipulation in virtual environments: relative size matters. In *Proceedings of CHI '99*, pages 48–55, New York, NY, USA, 1999. ACM Press.

[111] Yanqing Wang, Christine L. MacKenzie, Valerie A. Summers, and Kellogg S. Booth. The structure of object transportation and orientation in human-computer interaction. In *CHI '98: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 312–319, New York, NY, USA, 1998. ACM Press.

[112] Colin Ware and Danny R. Jessome. Using the bat: A six-dimensional mouse for object placement. *IEEE Comput. Graph. Appl.*, 8(6):65–70, 1988.

[113] Cornelia Weigelt and Simone Cardoso de Oliveira. Visuomotor transformations affect bimanual coupling. *Experimental Brain Research*, 148, 2003.

[114] Pierre Wellner. Interacting with paper on the digitaldesk. *Commun. ACM*, 36(7):87–96, 1993.

[115] Wayne Westerman. *Hand Tracking, Finger Identification, and Chordic Manipulation on a Multi-touch Surface.* PhD thesis, University of Delaware, 1999.

[116] Daniel Wigdor, Darren Leigh, Clifton Forlines, Samuel Shipman, John Barnwell, Ravin Balakrishnan, and Chia Shen. Under the table interaction. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 259–268, New York, NY, USA, 2006. ACM Press.

[117] Andrew D. Wilson. Touchlight: An imaging touch screen and display for gesture-based interaction. *International Conference on Multimodal Interfaces*, 2004.

[118] Andrew D. Wilson. Flowmouse: A computer vision-based pointing and gesture input device. In *Interact '05*, 2005.

[119] Andrew D. Wilson. Playanywhere: a compact interactive tabletop projection-vision system. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 83–92, New York, NY, USA, 2005. ACM Press.

[120] C. J. Worringham and D. B. Beringer. Directional stimulus-response compatibility: A test of three alternative principles. *Ergonomics*, 41(6):864–880, 1998.

[121] M. Wu, C. Shen, K. Ryall, C. Forlines, and R. Balakrishnan. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems, TableTop 2006*, 2006.

[122] Michael Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *ACM UIST*, pages 193–202, 2003.

[123] Ka-Ping Yee. Two-handed interaction on a tablet display. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1493–1496, New York, NY, USA, 2004. ACM Press.

[124] Robert Zeleznik and Andrew Forsberg. Unicam—2d gestural camera controls for 3d environments. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, pages 169–173. ACM Press, 1999.

[125] Robert C. Zeleznik, Andrew S. Forsberg, and Paul S. Strauss. Two pointer input for 3d interaction. In *SI3D '97: Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 115–ff., New York, NY, USA, 1997. ACM Press.

[126] S. Zhai, E. Kandogan, B. Search, and T. Selker. In search of the "magic carpet", design and experimentation of a 3d navigation interface. *Journal of Visual Languages and Computing*, 10(1), 1999.

[127] Shumin Zhai. *Human Performance in Six Degree of Freedom Input Control.* PhD thesis, University of Toronto, 1995.

[128] Shumin Zhai. User performance in relation to 3d input device design. *SIGGRAPH Comput. Graph.*, 32(4):50–54, 1998.

[129] Shumin Zhai and Paul Milgram. Quantifying coordination in multiple DOF movement and its application to evaluating 6 DOF input devices. In *CHI '98: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 320–327, New York, NY, USA, 1998. ACM Press.

[130] Shumin Zhai, Paul Milgram, and William Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *CHI '96: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 308–315, New York, NY, USA, 1996. ACM Press.