Thesis Proposal:
Principles and Applications of
Multi-touch Interaction

Tomer Moscovich

July 15, 2006

Abstract

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 parameters 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 which illustrate the benefits of multi-touch interaction. The techniques allow users to work faster and more fluently than do traditional single-point interaction methods. We describe a performance animation technique that allows users to 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).

Contents

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

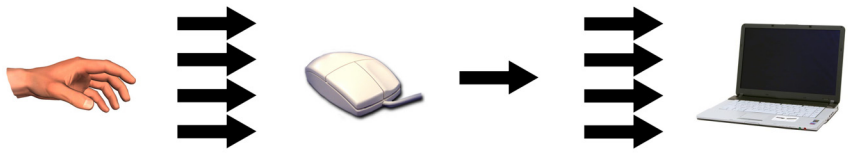

Figure 1.2: While our minds and bodies can control many streams of information, and computers can accept many streams of input, our interaction with the software must trickle through a single-point input device.

Real-world tasks often entail specifying multiple parameters.[1] 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 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.

[1] 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).

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.

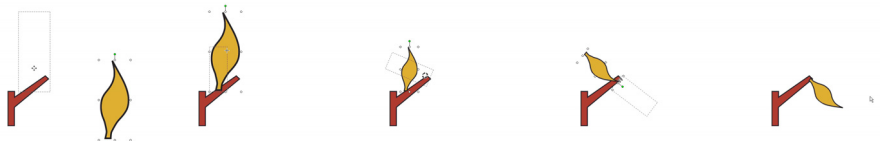

Figure 1.3: To set the leaf at the top of the branch using a standard drawing paradigm, the user selects the leaf and moves it near its destination. He then selects the scale widget and scales the leaf. Next, he selects the rotate widget and orients the leaf. Finally, he selects the leaf again and moves 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* [84], as experts become adept at performing increasingly large sub-tasks [25]. Parallelism has other cognitive benefits [68]. 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 [26] 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. [111], 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 [12] 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 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 [23]. For example, a touchpad can serve as an array of sliders [18] or a set of programmable buttons. In a similar fashion, a touchpad can be used for text-input by emulating a soft-keyboard [35]. 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 [104, 81, 35]. Finally, multi-point touchpads support bimanual interaction techniques. Two-handed input has been studied extensively, and is well understood [46, 52, 10]. 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 [81, 75], 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. As discussed above, the use of multi-point touchpads has many benefits. We therefore study properties of interaction using these devices to provide design guidelines for creating interfaces that use them (Chapter 3). Finally, we describe a series of novel multi-touch interaction ideas, and show specific examples of such techniques (Chapter 4). The contributions of this work are summarized in Chapter 5.

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 expressive animations, a difficult task using traditional methods. They simplify graphical manipulations tasks, such as object transportation 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 bi-

manual and tangible input. Our proposed experiments will reveal how well people can coordinate the motion of multiple fingers. They will also assess the role of stimulus-response compatibility, as well as control integrality and separability on the assignment of parameters in a manipulation task to the degrees-of-freedom of the hand. Using this knowledge we can 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 3.1.

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.

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.

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*.

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 several techniques for using these parameters.

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 [67]. 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 [31]. 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.

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

Rekimoto's SmartSkin [81] 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 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 [35] appear to be an array of

capacitive sensors, and can report the position, contact area, and eccentricity of multiple fingers. 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 [87], 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 [28, 69].

One of the earliest direct vision system for whole-hand interaction is Krueger's VIDEOPLACE [60]. 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 [98] 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 [75] 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 hight 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 [49] 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.* [92]

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 control.

A different approach to vision-based touchpads is described by Wilson [99, 101], who instead of relying tracking finger positions for input calculates an optical flow-filed. 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 [66] and Struman [85].

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 [96] and the Fingerball of Zhai *et al.* [111] 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 [40] and the EGG of Zhai [108]. 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 [39] is an elastic position controller coupled to an isotonic orientation controller. The rockin' mouse [8]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 [19], as are instrumented armatures [34]. ShapeTape [55] 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 [85] 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 [110] point out that interaction methods form a continuum ranging from the direct to the indirect. Direct interaction methods are based on an isomorphism [1] 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 [59].

An early example of whole hand gestures is the Charade system [13] which recognizes hand postures as commands to a presentation system. This

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

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 [104] 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.* [74] 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.* [44] 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.* [93] 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.* [103] 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 [21] 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 [52]. An experiment by Leganchuk [68] 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.

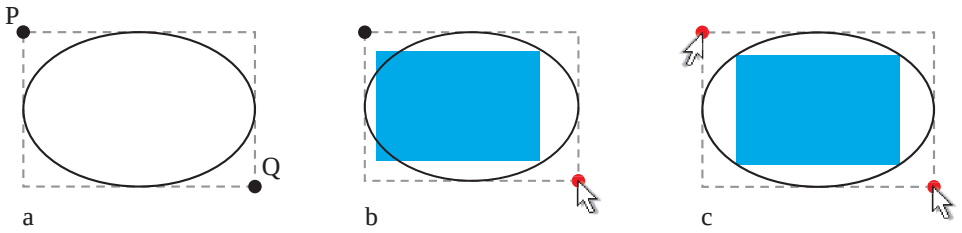

Figure 2.2: (a) To draw an axis-aligned ellipse, a user specifies opposing corner points P and Q of its bounding rectangle. (b) When using one hand, a user must specify these points one at at time. However, determining the location of the first point is a challenging mental visualization task. For example, 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 [46] 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 in-

teraction. 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.* [52] 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 citeKurtenbach97), 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.* [32] 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.* [10] 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 [64, 65, 63].

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 [61, 15]. 2D navigation methods take advantage of two hands for unified zooming and panning [50]. Various techniques for figure drawing [62, 61] and curve editing [65, 9, 80] 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 [51, 29] as well as for camera control and navigation [106, 11, 107]. 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 [37]. 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 [24]. 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 [18]. Benko *et al.* [14] 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.* [77] where the distance between one user's fingers on a table controls the width of a stroke drawn by another user.

Rekimoto [81] 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 [104]. Wilson [100] takes a related approach, by finding a rigid transformation that matches the optical

flow of the user's hand. Malik *et al.* [75] 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.

Dietz and Leigh [31] show how two contact points can determine an axis aligned rectangle. This technique was later used by Forlines and Shen [38] to specify regions-of-interest for fish-eye-lenses. Benko *et al.* [14] 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 [81] 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.* [74] where one hand positions the works space of the other to access a large display.

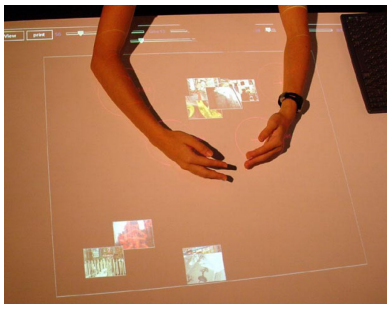

Figure 2.3: Rekimoto *et al.*'s "potential field" interaction: Objects slide away from the hand on the surface.

Chapter 3

Principles of Multi-touch Interaction

When developing a multi-touch interaction technique, the UI designer must make several decisions. In this chapter we examine the human factors that pertain to two key decisions. The first is the mapping between fingers and their motion on the touch-surface to parameters in the controlled entity. Many parameters of the hands and fingers can be used. These include the position of individual fingers or of the hand as a whole, relative positions and angles of individual fingers, the orientation of the hand, and so forth. The assignment of different sub-tasks to these parameters effects how well users can control the tasks, and how well they can coordinate or isolate them. We pay particularly close attention to the effects of assigning subtasks to one and two hands in order to relate the work to the well-studied topic of bimanual interaction. The second design decision we address is the visual feedback the software give users during manipulation tasks. The compatibility between this feedback, the user's mental model, and the motion of the user's hands may have a powerful effect on task performance [102].

A number of biomechanical, neural, and cognitive factors have bearing on these design decisions. For example, physical constraints limit the motion of the joints in the hand and wrist [72], while force-enslaving of neighboring fingers reduce their ability to move individually [47]. Similarly, the divergence of the output of neurons in the primary motor cortex yield coordination patters in finger motion [83]. We focus our study on the following questions: How well can people coordinate the simultaneous control of multiple hand parameters, and what factors affect this ability? What is the influence of a match or mismatch between the physical structure of the

task, and the cognitive structure of the control model? What are the effects of the perceptual-motor compatibility of the mapping on task performance?

3.1 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, 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 constrained 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 [70]. 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 unitary 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 Robert *et al.* [56] with regard to Garner's theory of the

perceptual structure of visual information [41]. 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. Robert *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 Robert *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.* [95] 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 Robert *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 most possibly by an independent visuomotor channel.

An 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 [36] 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 [102]. 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.

3.2 Multi-finger Control Vs. Bimanual Control

An important case of multi-finger interaction in the field of HCI, is the use of a single finger position from each of a user's two hands. This is the bimanual interaction condition, 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 [15], illustration techniques [61, 65] methods for 2D and 3D manipulation [64, 106], and more. Since all of these techniques simply 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. Then again, two hands may make separate parameter control easier, and may map better to many tasks.

Bimanual interaction techniques are generally classified as either symmetric techniques, where both hands play the same role [10], or asymmetric techniques, where the roles of the hands follow the principle of Guiard's Kinematic Chain [46]. A real world example of symmetric interaction is steering a bicycle, while peeling a potato is an asymmetric interaction. Can either group of techniques transfer to two fingers on one hand? In an asymmetric task, the non-dominant holds and positions an object to provide a

reference frame for the dominant hand to work in. This is very different from the typical role played by fingers of one hand, which must work in opposition about an object in order to maintain a stable grasp [72]; it seems unlikely that two fingers of one hand could play this asymmetric role. In a symmetric bimanual task, such as moving a large box, the role of the opposing hands is very similar to the role played by grasping fingers. Thus, it is reasonable to suppose that two fingers on one hand could be used instead of two hands in symmetric interaction techniques.

However the control structure of two fingers on one hand is very different from the control structure of two hands, so the two techniques may not be interchangeable. Two hand positions are structured as two separable pairs of integral spatial dimensions, while two finger positions are produced by a pair of integral spatial dimensions (the hand position) a possibly separable orientation, and the span of the hand. In the following sections we propose a series of experiments that explore the differences between these two techniques, so as to allow interface designers to choose the mapping most appropriate to the task.

Choosing between one and two-handed interaction methods is not the only reason to study the use of two fingers on one hand. Interaction using two fingers on one hand serves a good base-case for one-handed multi-finger interaction. In the case of the thumb and one opposing finger, it may be possible to generalize experimental results to multiple fingers. 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 [72]. There is also evidence which suggests that the major degrees of freedom of the hand can be represented by the thumb and finger. A study by Santello *et al.* [82] 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.

3.2.1 Measuring Coordination

An important aspect of high-degree-of-freedom control is the user's ability to manipulate multiple parameters in parallel. In order to compare the effect of different mappings on this 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 [68, 64, 21]. 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 [110] 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. To measure the amount of extra work users perform due to imperfect coordination Zhai and Milgram define a metric for *inefficiency* 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 [10]. 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 work together to simultaneous reduce tracking error. Given a distance d that a hand must move towards a target so as to achieve perfect tracking, and the actual distance a that hand moved towards the target, the instantaneous fractional error reduction for that hand is given by $E = a/d$. The instantaneous parallelism between the two hands is then $\min(\frac{E_l}{E_r}, \frac{E_r}{E_l})$. A mean parallelism of 0 results from sequential use of the hands during a session, while a value of 1 results from both hands simultaneously reducing their fractional error by the same amount.

3.2.2 Target Matching Task

Our first experiment centers around a type of target-matching task that has been commonly studied in the literature [68, 27, 61]. We ask participants to tightly enclose an ellipse within the bounds of a rectangle (see figure 3.1). This task is similar to the common marquee selection technique, and to many common drawing operations. Two-handed rectangle specification has even been used for image tone manipulation [63].

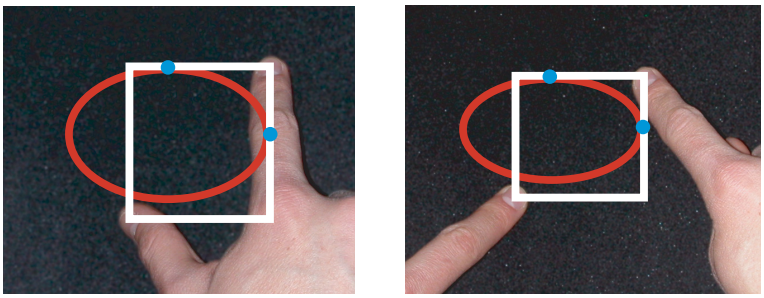

Figure 3.1: The study participant must tightly enclose the ellipse within the rectangle. A mark appears wherever a side of the rectangle is sufficiently aligned with the corresponding side of the ellipse. Two points on the touchpad control the upper right corner and the lower left corner of the rectangle. In the first mapping (left) the corners are controlled by the left and right hand. In the second (right) the corners are controlled by the thumb and finger.

Defining a rectangle using two hands proves to be more effective than using one hand. Leganchuk *et al.* [68] has shown that, for more cognitively demanding tasks, the performance enhancement cannot be explained by the greater physical efficiency of using two hands. The added benefit seems to stem from the externalization of the visual planning part of task, and greater ease of exploration. These benefits should also hold for unimanual two-finger manipulation. However, as discussed above the different structure of the control space may affect performance.

To measure differences in user performance we break the task down into two phases. The first we call the *approach* phase, in which the user brings the rectangle to the general vicinity of the target, and the second is an *adjustment* phase, in which the user fine-tunes the parameters to match the target precisely. The efficient execution of the approach phase requires good coordination between the degrees of freedom. Since this part of the task requires rough transportation and sizing of the rectangle, it may map better to the unimanual method, which has a single pair of spatial dimensions, than to the bimanual method, which has two. The adjustment phase, however, requires a precise match along each dimension. This may be easier to achieve by focusing on one parameter at a time while keeping the others fixed. An inability to fix parameter values may result in multiple acquisitions and overshoots along each dimension. Thus, the greater separability of the bimanual control space may prove superior in this part of the task.

This analysis leads us to the following hypotheses regarding user performance on this task:

H1 In both phases of the task, the unimanual mapping exhibits the greatest coordination as measured by Zhai *et al.*'s inefficiency metric.

H2 Inefficiency is positively correlated with completion time of the approach phase. (i.e. more inefficiency leads to longer completion times.)

Corollary to H1 and H2 The unimanual mapping leads to shorter approach completion times than the bimanual mapping.

H3 The adjustment phase is less coordinated than the approach phase under both mappings (more inefficiency).

H4 Inefficiency is negatively correlated with completion time of the adjustment phase. (i.e. more inefficiency leads to shorter completion times.)

Corollary to H1 and H4 The bimanual mapping leads to shorter adjustment phase completion times than the unimanual mapping.

H5 The bimanual mapping results in fewer target overshoots than the unimanual mapping.

H6 Overall trial completion time will be the same for both bimanual and unimanual conditions, i.e. due to a balance of advantages for each condition.

3.2.3 Continuous Tracking Task

One of the primary functions of our hands is to transport and orient objects in our environment. Whether moving a fork between a plate and our lips, or maneuvering a couch up a narrow staircase, our hands let us deftly control the position and orientation of objects in space. Computer interaction techniques that rely on direct manipulation are grounded in a metaphor between our handling of real-world objects and of digital ones. Because of this, these techniques often require the user to position and orient objects on the screen.

In the real-world, people move and orient objects in parallel [95]. This may be done using either one or two hands. When manipulating physical objects on a table people use one hand for small light objects, and two hands for large or heavy objects [64]. While two hands can more easily counteract the wight and moment of large physical objects, this may not be such an

important criteria in digital manipulation. We propose a study comparing symmetric bimanual manipulation with two-finger unimanual manipulation for a parallel transportation and orientation task. The study examines user performance and parameter coordination under two visual conditions. A better understanding of how different mappings affect user performance will help establish design guidelines for multi-touch systems.

We present participants with a segment tracking task similar to one previously used to study bimanual parallelism [10] (see Figure 3.2). 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. Unlike static target acquisition, a continuous pursuit task forces participants to coordinate their control of parameters as much as possible. Since serial control strategies become infeasible, it allows us to focus our measurements on participants' coordination abilities under the two conditions.

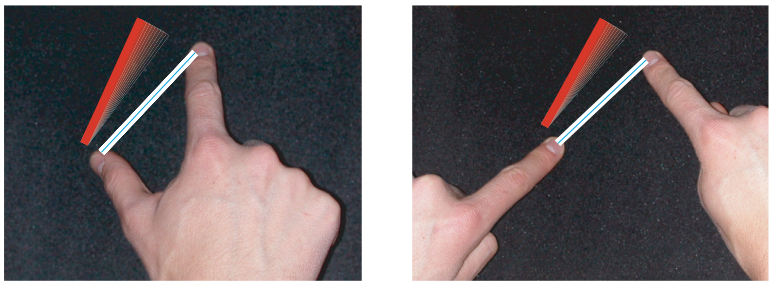

Figure 3.2: Study participants pursue a short segment as it moves and rotates on the screen. Two input points from either one or two hands control a "match segment" by moving it rigidly as though it were a stripe painted on a piece of glass. The segment is oriented parallel to the line through the input points.

Symmetric bimanual techniques for simultaneous translation and rotation have been successfully demonstrated by several researchers [62, 61]. However, the control structure of the two-finger unimanual method appears to match the task better than the bimanual method. Like the task, one hand has a single pair of spatial dimensions, and an orientation parameter. These may be easy to separate from the adduction and extension of the fingers relative to the hand. In contrast, the two pairs of spatial dimensions of two hands do not map as readily to the task, and encode a redundant fourth parameter which is not easily separated. This leads us to the following

hypotheses:

H1 Unimanual transportation and orientation exhibits a greater degree of coordination than the bimanual method (as measured by Balakrishnan *et al.*'s parallelism metric).

H2 Unimanual transportation and orientation exhibits a smaller tracking error than the bimanual method.

Perhaps the most striking difference between the one- and two-handed conditions is that in one, orientation is linked to hand rotation, and in the other orientation is a consequence of relative hand position. The latter has an additional level of indirection, but it does not seem to cause us any trouble in the real world. However, this transparency is dependent on a certain set of conditions. For example, it is known that bimanual coupling is affected by visuomotor transformations [97]. Since these types of transformations may be introduced when mapping the control to the task, it is important to study their effects. We do this by modifying the pursuit task so that the match segment is perpendicular to the line passing through the input points. It is as though we turned the piece of glass on which it is drawn 90 degrees (see figure 3.3). By applying the same transformation to the target segment we present participants with a motor task identical to the first experiment. In the unimanual condition, a clockwise hand rotation still rotates the match segment clockwise. In the bimanual condition rotating the hands clockwise about their midpoint still rotates the match segment clockwise, however, moving the left hand up moves the leftmost point of the segment to the right and similarly moving the right hand down moves the rightmost point left. Since this does not follow the perceptual-motor compatibility principles discussed in section 3.1, it may significantly impact user performance. Thus, we propose the following hypotheses:

H3 In the rotated condition, unimanual transportation and orientation exhibits more coordination and a smaller tracking error than the bimanual method.

H4 Unimanual transportation and orientation is unaffected by rotation of the visual stimulus.

H5 Bimanual transportation and orientation in the rotated condition exhibits worse coordination and greater error than bimanual method in the unrotated condition.

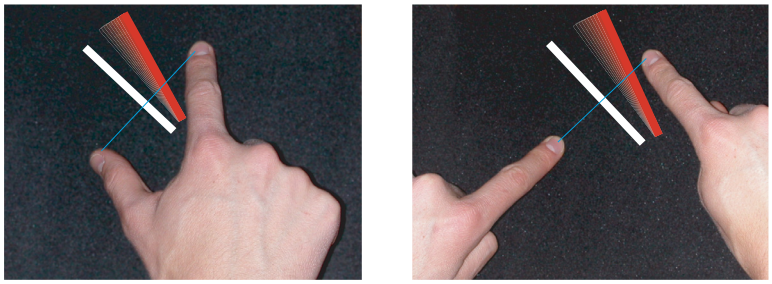

Figure 3.3: By rotating the target and matching segments, while preserving the motor control task, we check for directional compatibility issues in the bimanual method.

3.2.4 Discussion

The results of these experiments will shed light on our ability to coordinate multiple parameters using one hand. We expect they will show that multi-touch interaction is a viable means for coordinated input of multiple continuous variables. The experiments will also answer several questions regarding how hand parameters should be mapped to software parameters. Does it matter if the control structure matches the structure of the task? if so, what is the optimal match? Are our hypotheses about the control structure of fingers on one and two hands valid? Furthermore, while it is reasonable to believe that perceptual-motor compatibility is important in determining these mappings, it is not yet known how this compatibility can be assured. Our experiment will show whether object rotation is compatible with the orbit of two hands about a point, and with the rotation of a single hand.

Chapter 4

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 guide-posts 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.

4.1 Deformation and Animation

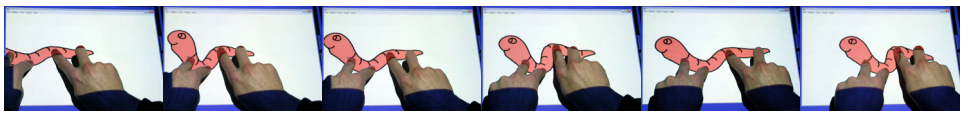

Figure 4.1: Animation sequence of a crawling worm. The animator moves and bends the drawing with his fingers as though manipulating a real-world flexible object.

4.1.1 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 [88]. 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.* [48] 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 [7, 33, 78], 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 4.1 and 4.2).

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 [88]. Since our method produces plausible deformations for the interior of shapes, it can significantly reduce the need to redraw decorations.

4.1.2 User Experience

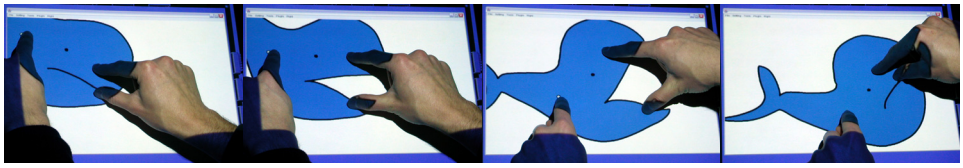

Figure 4.2: A user makes a whale open and close its mouth as it swims.

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.

4.1.3 Implementation Details

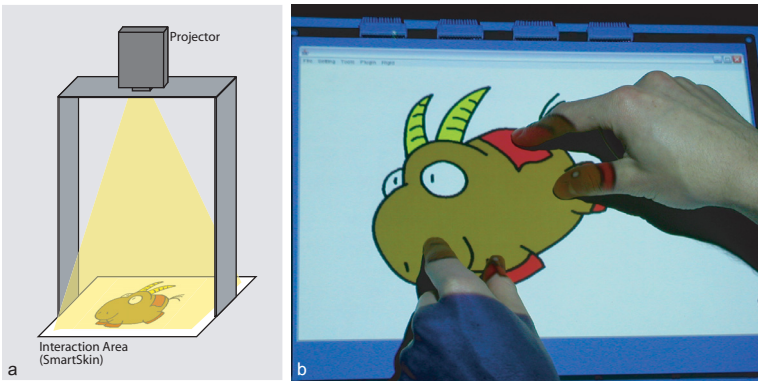

Figure 4.3: (a) Overview of our system setup. (b) Close-up of interaction area.

Our setup consists of a digital projector mounted on a platform above a multi-finger touchpad (see figure 4.3). 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 [81]. 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 through a sequence of frames using a greedy exchange algorithm described by Veenman *et al.* [91]. 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).

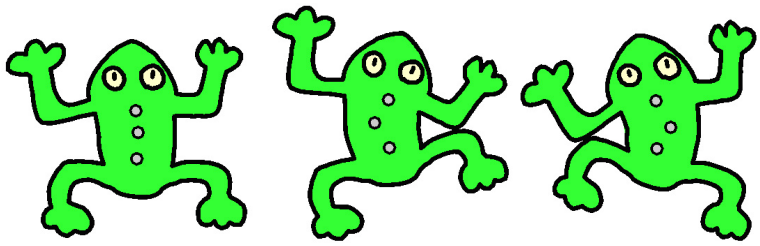

Figure 4.4: Moving an internal constraint point instantly affects the entire shape.

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 [54]. 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 4.4). 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.

4.1.4 Related Work on Accessible Animation

While previous systems have used direct whole-hand manipulation to accomplish a variety tasks [104, 81], 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 [30, 7]. More complex animation is possible through digital puppetry [86], 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 [33]. 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 [89], or pre-specified configurations [79, 53]. These approaches produce pleasing animations, but they restrict the type of motion that the user can produce to previously designed animation.

4.2 Multi-finger Cursors

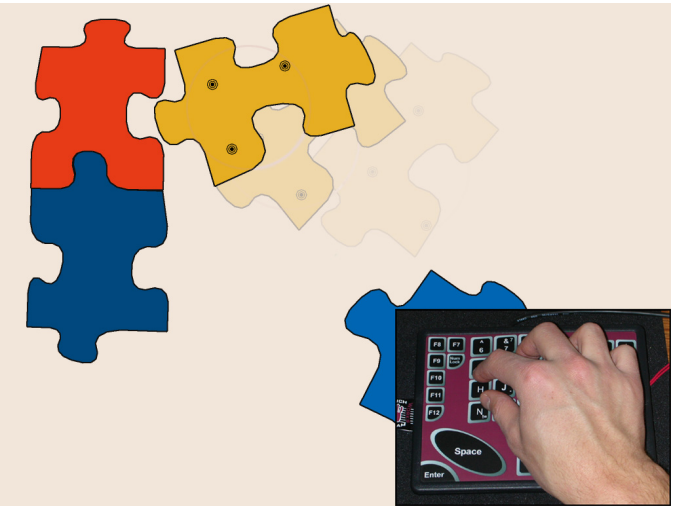

Figure 4.5: The hand cursor lets the user move the puzzle pieces as though sliding objects on a physical table.

4.2.1 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 [81, 104, 54], 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 [57, 20]. 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 [17]. 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, tools 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.

4.2.2 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 [109]. 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 [22]. Our touchpad supports a single press on the interaction-surface that corresponds to a mouse-button press (see section 4.2.4).

4.2.3 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.

The hand cursor allows users to control graphical elements as though manipulating rigid real-world objects. For example, figure 4.5 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

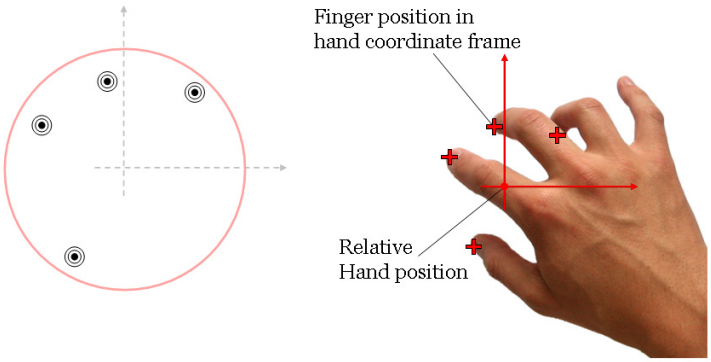

Figure 4.6: The hand cursor maps each finger contact on the touchpad (right) to a point on the screen (left). The position of the fingers is determined within a reference frame set by the position of the hand. Hand position is estimated as the mean finger position. The motion of the hand controls the relative position of the cursor on the screen, the extent and position of which are indicated by an enclosing circle.

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 [24] or for modifying the control points of a curve (see figure 4.7). Multi-point input is also useful for performance animation as discussed in section 4.1 (see figure 4.7).

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 [47, 83]. 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 [54, 71]. They can also be useful in asymmetric interaction tasks such as controlling the orientation of a toolglass ruler [46, 16, 61].

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.

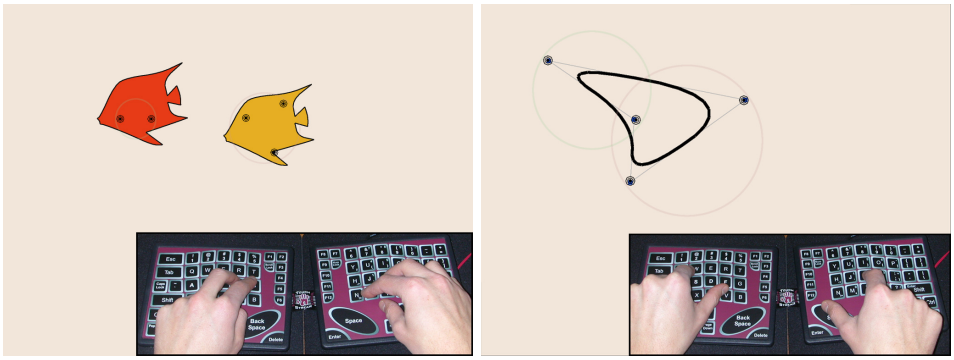

Figure 4.7: (Left) Controlling many parameters at once is useful for performance animation. Here the user animates two swimming fish by using two hand cursors. (Right) Using several fingers allows users to control objects with many degrees of freedom, like the control points of this subdivision curve.

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

4.7 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 4.2.3.)

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.*[81] 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 [43, 90]. 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.

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 4.8). 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 [94]. Instead, we indicate scaling by animating stripes which slide toward and away from the center at a rate proportional to the scale rate.

Figure 4.8: 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.

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 [64]. 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.

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 [76] 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

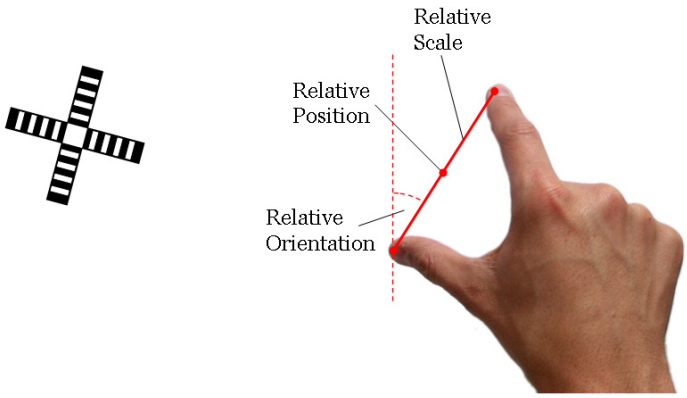

Figure 4.9: The similarity cursor is controlled using two fingers on the touchpad. Change in the position of their centroid controls the relative position of the cursor. Change in the slope of the line they define controls the relative orientation of the cursor. The changing distance between the points adjusts the size of the object manipulated by the cursor, and is indicated by animated stripes.

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 finger. 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

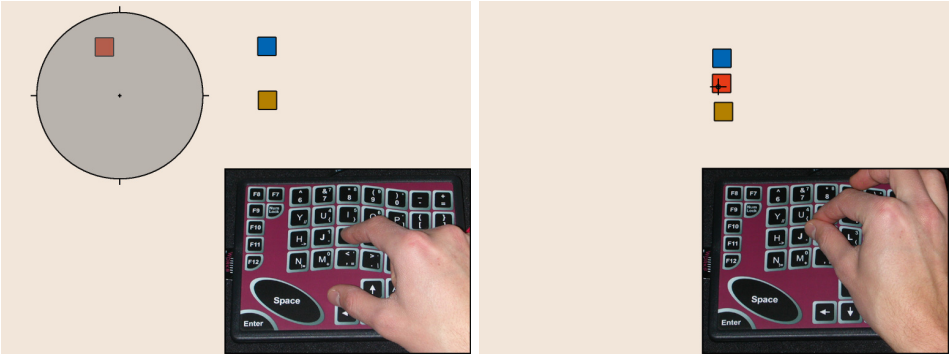

Figure 4.10: (Left) The large activation area of the adjustable area cursor reduces the time and precision needed to acquire isolated targets. (Right) The selection of fixed-radius area cursors is ambiguous in crowded regions. This ambiguity is resolved with the adjustable area cursor by minimizing the activation area.

Our third technique extends the idea of area cursors [58], 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 4.10 (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 4.10 (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 4.11 (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 [58, 45] 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 4.11 (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

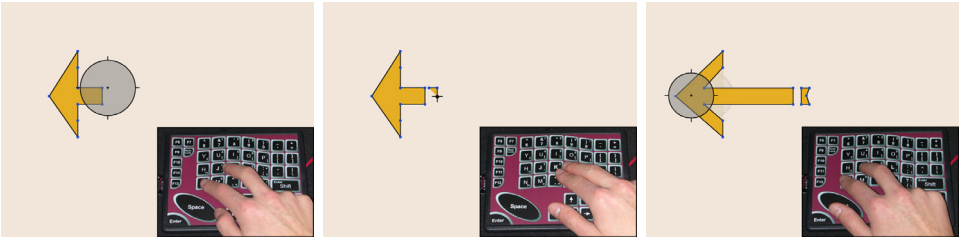

Figure 4.11: (Left) Traditional area cursors make it impossible to click on empty space near selectable objects. (Center) The adjustable area cursor can emulate a point cursor without requiring the user to switch modes. (Right) Since the adjustable area cursor does not suffer from the ambiguity of fixed-area cursors, it can be used to control groups of nearby objects.

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 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).[1]

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 4.10 and 4.11). 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

[1] 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.

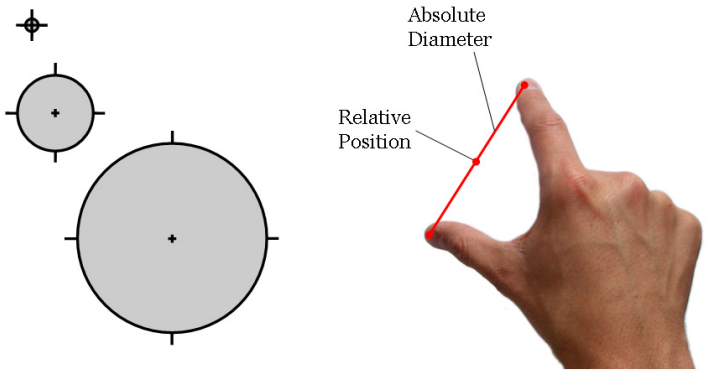

Figure 4.12: The motion of the centroid of the fingers on the touchpad controls the relative position of the adjustable area cursor. The greatest distance between contact points is mapped to the radius of the cursor.

boundary to include all selected objects (as in the Bubble Cursor [45]). A simpler alternative is to highlight all selected targets.

4.2.4 A Relative Multi-point Touchpad

For multi-point input we use a FingerWorks TouchStream touchpad [35]. 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.*[24], and enhanced by MacKenzie [73], 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.* [14] 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).

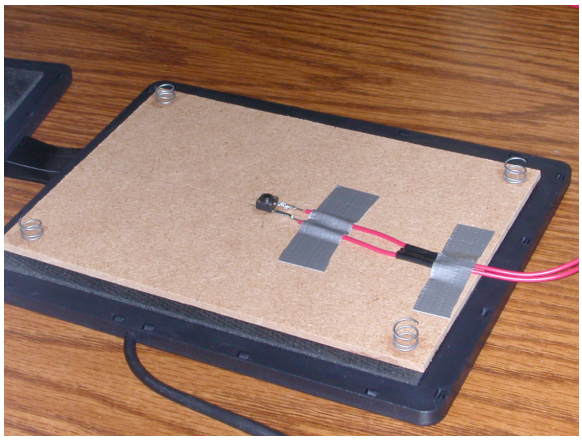

Figure 4.13: Our prototype touchpad uses a tactile switch to allow users to distinguish tracking from dragging motion.

To overcome these problems, we place a single tactile button beneath the touchpad (Figure 4.13), 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 [47] 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 4.2.3. 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 [57]. Our cursors use the Windows XP gain function [76] which in practice yielded a Control/Display ratio ranging between 0.8 and 3.6.

4.2.5 Discussion

When an artist 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 [68].

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, the additional cognitive load of the adjustable area cursor may render it less than useful for single target selection in dense areas. It is also important to study the effects of using the same muscle groups to control the cursor parameters while simultaneously indicating the dragging state by maintaining pressure on the touchpad.

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? [2]

We have observed an interesting phenomenon in our informal tests of the system: When using multiple fingers to control cursors, certain behaviors that resemble gestural interfaces appear. 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.

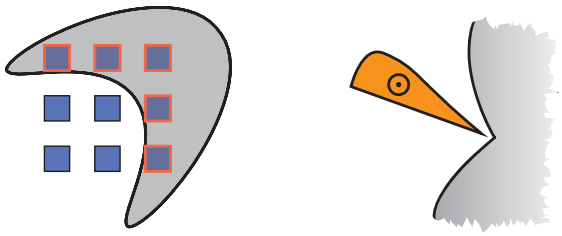

Figure 4.14: More cursor possibilities: A shaped area cursor for more presice selections (left) and a parameterized 2D sculpting tool (right).

The design space for multi-finger cursors is still largely unexplored, and contains many enticing possibilities. For example, adjusting the shape of the area cursor may be used for more precise grouping (see figure 4.14(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 parameterized cursor who's 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 4.14(right)).

4.3 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,

[2] Some of these questions are addressed in Chapter 3.

and to properly represent it requires interaction in three dimensions. Moving from two to three dimensions does not add only a single parameter. For example, positioning and orienting an object in 3D requires control of twice as many parameters as in are needed in 2D. As a consequence of this, software for 3D manipulation is often brimming with a variety of control widgets and manipulation modes. To address this problem several specialized three and six-degree of freedom input devices have been created [2, 51, 111, 39, 8]. While these may work well for 3D manipulation, they may not perform as well in standard 2D interaction tasks which have been designed with a 2D device in mind. Since the supporting interface in 3D software is often 2D (menus, sliders, etc.) users are forced to alternate between input devices. 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 [105, 42], and while their usability has not been formally tested, they show that this may be a promising direction. We propose to further explore the possibilities of multi-touch 3D control, as the benefit of integrated 2D and 3D input is great. Possible uses include object manipulation and camera controls, shape and surface editing, as well as bimanual 3D interaction techniques [52, 11].

Chapter 5

Conclusions

5.1 Summary of Contributions

This dissertation presents two types of contributions. First, it shows that multi-touch interaction is useful for a variety of interaction tasks. Novel techniques presented in this work allow people to easily accomplish tasks that were formerly difficult. For example, the multi-touch deformation interface lets people 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, it increases 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.

5.1.1 Human abilities in multi-touch interaction

Our experimental results establish multi-touch interaction as a viable means of coordinated multi-parameter input. Furthermore, they provide information that will be useful fur designing such interaction. The relationship of human cognitive processes with the physiology of the arms and hands is complex, and often leads 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, forcing the designer into prolonged cycles of iterative design and testing. A better understanding of human abilities can help eliminate poor design choices. For example, an early study by Buxton and Myers [21] on bimanual interaction indicated that using two hands to perform two sub-tasks in parallel

leads to shorter completion times in a positioning/scaling task and in a navigation/selection task. However, a later study by Dillon *et al.* [32] using a menu-selection/drawing task found only a nominal improvement in the bimanual condition. For some reason, the tasks tested by Buxton and Myers were better suited for bimanual interaction than the task chosen by Dillon. This difference can be explained by theories such as Guiard's kinematic chain model for bimanual interaction [46], which describes how two hands work cooperatively, and by results of studies such as one by Balakrishnan and Hinckley [10], which reveals the importance of visual integration for coordinated two-handed interaction.

As the input space increases in complexity, interface designers are faced with more and more choices. In multi-touch interaction, important choices include deciding which sub-tasks the user should execute concurrently, and how hand parameters should be mapped to software parameters. The results of our proposed experiments will help guide this design. More specifically, we hypothesize that users find it easier to coordinate parameters that have an integral control structure, and that parameter assignment that matches the input control structure to the structure of the software parameters will lead to better performance than otherwise. We describe the properties of the control-structure of multi-touch input in chapter 3. The study results will also help designers maintain the perceptual-motor compatibility of their interface mappings.

Our work also relates multi-touch interaction to the well-studied topic of bimanual interaction. It shows when and how it is possible to transfer bimanual interaction techniques to unimanual multi-touch methods. They also reveal new facts about bimanual interaction, such as its sensitivity to perceptual-motor compatibility issues in a rotation task.

5.1.2 Multi-touch interaction techniques

We present 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 introduce the concept of a multi-touch cursor as an instrument for enhancing our natural abilities. We provide three examples of how such cursors can be designed and used, and discuss further directions for cursor design. The cursor techniques we describe, 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 ma-

nipulation into a single step). These and similar techniques have immediate use in illustration and presentation software.

Our multi-touch deformation technique can save expert animators time in creating key-frames, while allowing 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.

Finally, we discuss the utility of multi-touch interaction in 3D manipulation. These types of techniques will allow users to seamlessly switch between interaction in two and three dimensions using a single general-purpose input device.

5.2 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 some other interference

5.3 Future Research Directions

While this work has advanced our understanding of multi-touch interaction, there are many more questions to answer, and ideas to explore. For example, our experiments 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 one hand. Research has shown that about 80% of the variance in static hand postures is explained by only two principle components [82], 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? 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 and typing. This ability may prove useful for multi-touch interaction techniques designed with expert users in mind.

Another interesting direction of research is the study of multi-touch input in the context of tangible user interfaces. Many of the benefits of tangible UIs, such as spatial multiplexing and parallel input, stem from the fact that they allow the use of multiple fingers. The downside of tangible UIs is that they require a lot of special-purpose hardware that users must maintain and organize. Can we attain the benefits without the drawbacks? How important is the palpability of the interface? Would a simulation provide the same benefits as a physical implementation? The union of multi-touch interaction with the affordances of physically simulated objects and tools [3] creates new opportunities for interaction design.

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. These techniques also create space for new hardware designs, such as mice that sense orientation and finger extension.

There are also 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 [65]. 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Gábor Blaskó and Steven Feiner. Single-handed interaction techniques for multiple pressure-sensitive strips. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1461–1464, New York, NY, USA, 2004. ACM Press.

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

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Didier Casalta, Yves Guiard, and Michel Beaudouin Lafon. Evaluating two-handed input techniques: rectangle editing and navigation. In *CHI '99: CHI '99 extended abstracts on Human factors in computing systems*, pages 236–237, New York, NY, USA, 1999. ACM Press.

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

[29] 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.

[30] 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.

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

[32] 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.

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

[34] 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.

[35] FingerWorks. iGesture Pad.

[36] 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.

[37] George W. Fitzmaurice, Hiroshi Ishii, and William A. S. Buxton. Bricks: laying the foundations for graspable user interfaces. In *Pro-*

ceedings of CHI '95, pages 442–449, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[38] 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.

[39] 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.

[40] 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.

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

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

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

[55] Measurand Inc. Shapetape.

[56] 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.

[57] 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.

[58] 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/Addison-Wesley Publishing Co.

[59] Stelios Kourakis. On gestural interaction and modular gestures, 2004.

[60] 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.

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

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

[63] 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.

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

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

[73] 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/Addison-Wesley Publishing Co.

[74] 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.

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

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

[77] 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.

[78] Tomer Moscovich and John F. Hughes. Animation sketching: An approach to accessible animation, 2003.

[79] 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.

[80] 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.

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

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

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

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

[85] David J. Struman. Whole-hand input, 1992.

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

[87] Tactiva Inc. TactaPad.

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

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

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] 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.

[95] 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/Addison-Wesley Publishing Co.

[96] 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.

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

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

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

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

[101] 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.

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

[103] 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.

[104] 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.

[105] 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.

[106] 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.

[107] 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.

[108] Shumin Zhai. Human performance in six degree of freedom input control., 1995.

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

[110] 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/Addison-Wesley Publishing Co.

[111] 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.