Multi-finger Cursor Techniques

Tomer Moscovich
Department of Computer Science
Brown University Box 1910
Providence, RI 02912, USA
tm@cs.brown.edu

John F. Hughes
Department of Computer Science
Brown University Box 1910
Providence, RI 02912, USA
jfh@cs.brown.edu

Abstract

The mouse cursor acts as a digital proxy for a finger on graphical displays. Our hands, however, have ten fingers and many degrees of freedom that we use to interact with the world. We posit that by creating graphical cursors that reflect more of the hand's physical properties, we can allow for richer and more fluid interaction. We demonstrate this idea with three new cursors that are controlled by the user's fingers using a multi-point touchpad. The first two techniques allow for simultaneous control of several properties of graphical objects, while the third technique makes several enhancements to object selection.

Résumé

Le curseur de la souris est un avatar digital pour notre doigt. Cependant, nos mains ont beaucoup plus qu'un seul doigt, ainsi que de nombreux axes de mouvement que l'on utilise pour manipuler notre environnement. Nous postulons que la création d'un curseur graphique qui réflèterait encore plus d'aspects de la main permetterait des interactions plus riches et plus fluides. Nous démontrons cette idée avec trois nouveaux curseurs qui sont contrôlés en glissant plusieurs doigts à la fois sur une palette de digitalisation. Les deux premier curseurs manipulent les objects graphiques directement, alors que le troisième manipule la sélection des objects.

CR Categories: H.5.2 [Information Interfaces]: User Interfaces—Interaction styles; H.5.2 [Information Interfaces]: User Interfaces—Input devices and strategies

Keywords: Cursors, Multi-touch Interfaces, High Degree-of-freedom Input.

1 Introduction

People often wish that interacting with a computer were as convenient as interacting with the real world—until we remind them of the "undo" operation, which the real world lacks. Even with the benefit of "undo," much of our interaction with computers can be frustrating, partly because it's mediated by the mouse or other similar cursor-control devices; enriching this aspect of interaction is our goal. The cursor itself is an indirection in our interaction: we control some device like a mouse, which in turn controls a cursor, which has power over either some model (a text document, a drawing) or our view of it. This indirection is analogous to the craftsman's use of tools: the same hand may first use a screwdriver and later a wrench. This is not to say that direct manipulation is a bad thing—merely that the indirection introduced through cursors also enables a powerful variety of tasks to be carried out with a single device. For some of these tasks direct manipulation would be inappropriate. Just as we like to use tweezers to remove splinters and a net to catch butterflies, we can use digital tools to select a control-point on a curve with precision, or reach across a wall-sized display with a flick of the wrist. The cursor acts as the visible end-effector of such tools.

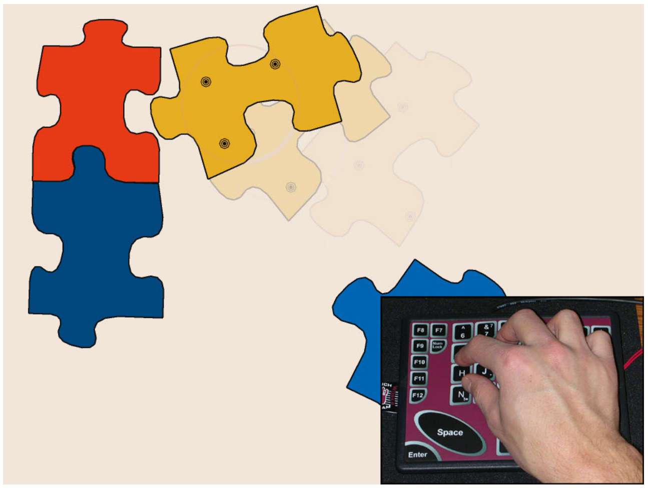

Figure 1: The Hand Cursor lets the user move the puzzle pieces as though sliding objects on a physical table.

Cursors in desktop interfaces are typically controlled by only two parameters (the motion of a mouse in x and y). This feels rather limited; it's a great deal easier to move a book to a particular place and orientation on a physical desk than to move a rectangle to a particular place and orientation on the virtual desktop. Therefore, we present here three new cursors that control several parameters at a time; the cursors themselves are controlled by multi-touch tracking of the user's hand, i.e., by the positions of one or more of the user's fingers on a touchpad that can detect multiple finger locations at once. These techniques demonstrate the power of simultaneous multi-parameter cursor control, and show how the indirection provided by a cursor can overcome the physical constraints found in similar direct-touch techniques. It is important to note that we have chosen these techniques as representative points in the design space of multi-finger cursors; they may be combined, modified, or extended to suit various applications. We also discuss the limitations of these techniques, some hurdles that must be overcome to make such cursors effective, and challenges for future work.

2 Design Principles

We have attempted to design our cursor techniques so they would be easy for an experienced mouse user to use and understand. We do this by maintaining, whenever possible, certain key attributes of the graphical cursor. The first property is a continuous zero-order map-

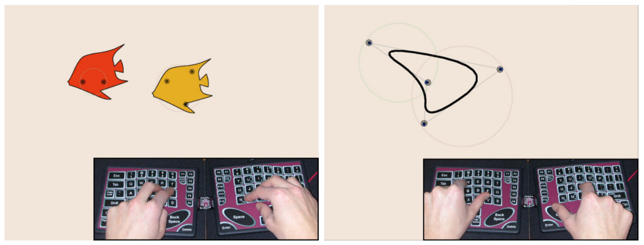

Figure 2: (Left) Controlling many parameters at once is useful for performance animation. Here the user animates two swimming fish by using two hand cursors. (Right) Using several fingers allows users to control objects with many degrees of freedom, like the control points of this subdivision curve.

ping. 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 devices lead to faster performance than other types of controllers [34]. The limited range of the controller can be addressed by making the mapping relative instead of absolute (the user can clutch by lifting and repositioning), and by applying a speed-dependent gain function which 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, since the space taken up by several fingers decreases the effective size of the touchpad.

It is important, with relative positioning devices, to be able to differentiate between the tracking and dragging states [3]. Our touchpad supports a single press on the interaction-surface that corresponds to a mouse-button press (see section 4.1).

3 Overview of the Techniques

We begin by describing the three techniques, their benefits, and their limitations. Implementation details and other considerations are discussed in section 4. All three techniques map the position of several fingers on a touchpad to parameters of the cursor.

Multi-finger touchpads are now commodity hardware [6, 30]. As general purpose input devices, they can be used to drive the traditional mouse cursor by simulating single-finger touchpads. Thus our techniques integrate seamlessly into existing GUI frameworks.

3.1 The Hand Cursor

Our first technique is a multi-finger Hand Cursor. It displays a set of points on the screen, which correspond to contact points on the touchpad. These finger points allow the user to control graphical elements as though manipulating rigid real-world objects. For example, Figure 1 shows the user moving and rotating a puzzle piece just as one would maneuver a rigid object lying on a table.

Multiple fingers also allow the user to grasp several objects at once, which is useful whenever it is necessary to control multiple parameters concurrently. For example, it may be used to control an array of sliders [4] or for modifying the control points of a curve (Figure 2 (Right)). Multi-finger input is also useful for performance animation [13] (Figure 2 (Left)).

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 [12]. 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 [13, 22]. They can also be useful in asymmetric interaction tasks, e.g., controlling the orientation of a toolglass ruler [11, 1, 18].

In the real world we frequently 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. To help the user perceive the cursors as two separate hands, we draw a circle surrounding the fingers of each hand. Note that this indicator only shows grouping, but does not fill in the "body" of the hand. Test users of an earlier implementation which used a translucent disc to indicate the hand attempted to select objects with the disk rather than the fingers. (This type of selection may actually be appropriate for some tasks, see Section 3.3.)

The touchpad-to-screen correspondence is *not* a one-to-one mapping from points on the touchpad to points on the screen. 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; touch-points can never be closer than the width of a finger, and this minimum distance may be greatly magnified on a display. A one-to-one mapping also presents problems when using two touchpads for two-handed input. Unless the touchpads were several times larger than the span of each hand, the working space for the fingers of each hand would overlap in a confusing manner that is rarely experienced in the real world.

We solve these problems by scaling the coordinate frame of the cursor so that the finger distances are appropriate for the manipulation task (i.e. small enough to comfortably fit on the interface while maintaining a reasonable reach). We then translate this coordinate frame by the motion of the hand (see section 4.2). By applying mouse acceleration to this motion, we give the hand access to the entire screen, while maintaining high precision at low speeds. This reflects the natural relationship of the hand to the fingers, where the fingers work in a frame of reference relative to the hand. This technique is transparent; no one who used our system even commented on it. 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 2 is easy, but placing fingers on the control points is difficult.

3.2 The Similarity Cursor

Since users generally control only one object at a time, 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 the user to focus on a single target, while simultaneously controlling its position, rotation, and scale (i.e., determining a orientation-preserving *similarity* transformation). The cursor is controlled using two fingers by a simple mapping from the hand's position and orientation, and from the span of the fingers (Figure 3).

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 [19]. This is especially evident at the final stage of alignment, where separately adjusting each property may undo the previous

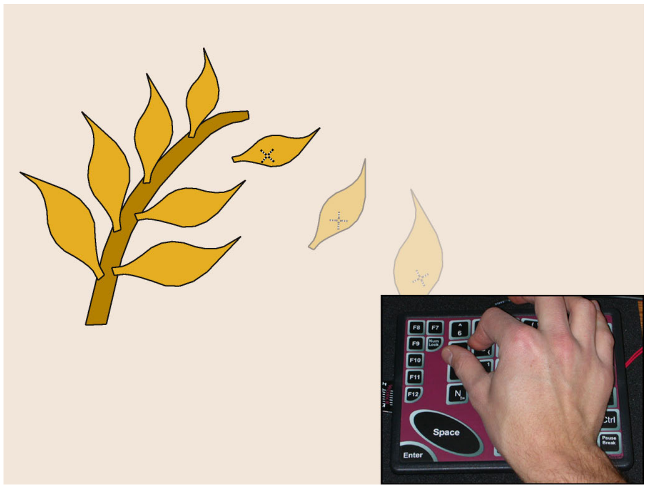
Figure 3: A user simultaneously translates, rotates, and scales a leaf using the Similarity Cursor. Parallel control of an object's properties allows for more fluid interaction, and may decrease task completion time.

adjustment. We expect that this will hold for parallel input using one hand.

We provide feedback regarding the cursor state even when the user is not controlling an object. To do this we render the cursor as rotating cross-hairs which show the translating and rotating motion of the hand. We do not scale the cursor to indicate scaling, since it is undesirable to have a cursor that is too large or too small for the task [32]. Instead, we indicate scaling by animating stripes which slide toward and away from the center at a rate proportional to the rate of scaling.

3.3 The Adjustable Area Cursor

Our third technique extends the idea of area cursors [16], 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 (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 (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 5 (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 [16, 10] 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 5 (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.

4 Implementation Details and Discussion

4.1 A Relative Multi-point Touchpad

For multi-point input we use a FingerWorks TouchStream touchpad [6]. The touchpad provides a 16.2cm$\times$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. [4], and enhanced by MacKenzie [23], 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.

To overcome these problems, we place a single tactile button beneath the touchpad (Figure 6), 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. However, since finger movements are not completely independent [12] 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

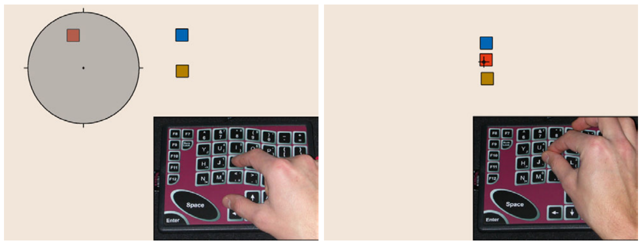
Figure 4: (Left) The large activation area of the Adjustable Area Cursor reduces the time and precision needed to acquire isolated targets. (Right) The selection of fixed-radius area cursors is ambiguous in crowded regions. This ambiguity is resolved with the Adjustable Area Cursor by minimizing the activation area.

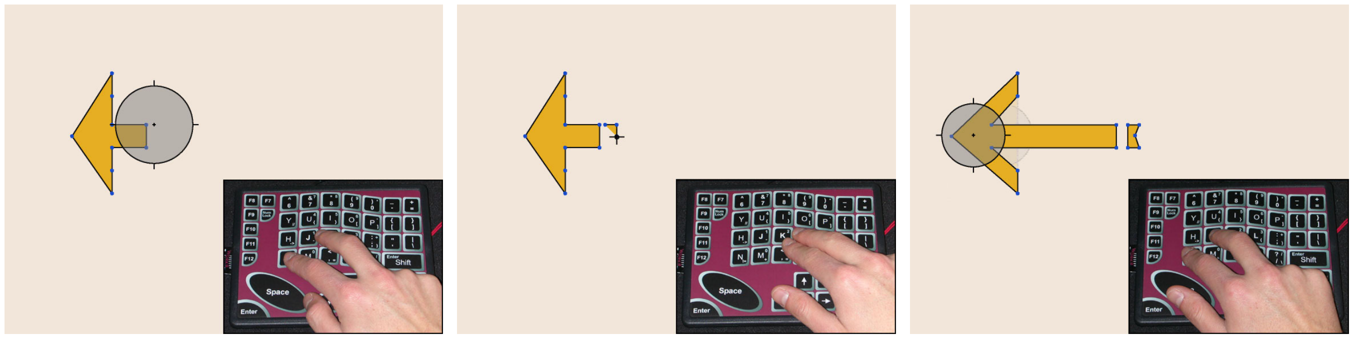
Figure 5: (Left) Traditional area cursors make it impossible to click on empty space near selectable objects. (Center) The Adjustable Area Cursor can emulate a point cursor without requiring the user to switch modes. (Right) Since the Adjustable Area Cursor does not suffer from the ambiguity of fixed-area cursors, it can be used to control groups of nearby objects.

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. 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 the cursor motion is essential to minimize clutching [15]. Our cursors use the Windows XP gain function [27] which in practice yielded a Control/Display ratio ranging between 0.8 and 3.6.

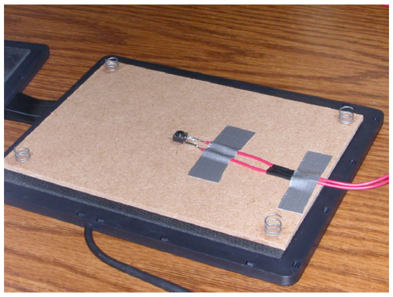
Figure 6: Our prototype touchpad uses a tactile switch to allow users to distinguish tracking from dragging motion.

4.2 Hand Cursor Implementation

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.

4.2.1 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. [28] 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 [8, 31]. 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.

4.3 Similarity Cursor Implementation

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 above-mentioned gain function to this motion to reduce the control footprint and increase precision.

4.3.1 Rotation

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

We use the same gain function for rotation as we do for translation. However, since the gain 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: $d = ||RP - P||$.

Because the best-fit rotation is computed after we have accounted for the best-fit translation using centroids, objects end up rotating about the cursor-center, which is moved by the centroid of the fingers, so rotations of objects seem to the user to be about the centroid of the fingers. If the user chooses to hold one finger fixed and rotates the others 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.

4.3.2 Scaling

We set the 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.

4.4 Adjustable Area Cursor Implementation

This cursor is controlled by one or more fingers. The cursor is moved by the gain-adjusted translation of the centroid of the contact points, while the diameter of the cursor is set to a multiple of the maximum distance between touch-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]

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

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 and 5). Short radial segments extending along the main axes become a cross-hair indicator when the radius is zero. This indicator may be enhanced to disambiguate the selection of partially overlapping targets by modifying its boundary to include all selected objects (as in the Bubble Cursor [10]). A simpler alternative is to highlight all selected targets.

5 Related Work

Multi-point touch-sensitive tablets have existed for more than two decades [20], yet, to our knowledge, they have never been used to control cursor parameters beyond position and, occasionally, pressure. Closer to our work are video-based tracking systems such as the Visual Touchpad [26, 25], and VIDEOPLACE [17] which overlay segmented live video of the user's hands (or body) onto the screen. While the digital hands do act as cursors, the systems do not abstract the video parameters, but rather rely on a homography between the video and the screen. This means that the digital hands possess many of the physical constraints real hands have (e.g. limits in finger resolution and hand rotation).

Work using the Visual Touchpad has focused on a style of interaction where hand gestures invoke a mapping between a continuous hand parameter and some control widget or action. This style of multi-finger interaction has been studied by Grossman et al. for volumetric displays [9], and by Wu et al. for table-top interaction [33].

Another branch of multi-finger interaction systems have focused on touch-screen type interaction, where the display shares the same space as the touch-surface [28, 5]. Many graspable user interfaces also share this property [7, 14]. The techniques described by these systems illustrate the power of high-degree-of-freedom input. Since all of the interaction is performed directly by the hands, no intermediary cursor is needed. However, these systems are limited by physical constraints such as occlusion, and space required by fingers or handles.

Two handed input techniques also allow for simultaneous control of multiple parameters [2]. However, these parameters are typically mapped to two positional cursors [19], or are mapped directly to model parameters.

Cursors with more than two degrees of freedom are common in 3D interaction [35, 24]. Generally, the position and orientation of a 3D tracking device is mapped to the position and orientation of the cursor. Sturman and others have investigated whole-hand interaction using instrumented gloves [29]. Much of this work has focused on using hand-gestures to trigger actions, or on using a direct mapping of hand parameters to the parameters of some model.

6 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 [21].

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?

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.

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. 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.

7 Acknowledgments

We would like to thank Dan Vogel for an implementation of the Windows XP gain function, Dan Keefe for hardware assistance, Guillaume Marceau for abstract translation, and Bob Zeleznik for helpful discussions.

References

[1] 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.

[2] 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.

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

[4] 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.

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

[6] FingerWorks. iGesture Pad.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *CHI*, pages 234–241, 1997.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] Jock D. Mackinlay, Stuart K. Card, and George G. Robertson. Rapid controlled movement through a virtual 3d workspace. In *Proceedings of SIGGRAPH '90*, pages 171–176, New York, NY, USA, 1990. ACM Press.

[25] 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.

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

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

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

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

[30] Tactex Controls Inc. Kinotex pressure sensing material.

[31] 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.

[32] 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.

[33] 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.

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

[35] Shumin Zhai, William Buxton, and Paul Milgram. The "silk cursor": investigating transparency for 3d target acquisition. In *Proceedings of CHI '94*, pages 459–464, New York, NY, USA, 1994. ACM Press.