ShapeTouch: Leveraging Contact Shape on Interactive Surfaces

Xiang Cao[2,1], Andrew D. Wilson[1], Ravin Balakrishnan[2,1], Ken Hinckley[1], Scott E. Hudson[3]
[1]*Microsoft Research,* [2]*University of Toronto,* [3]*Carnegie Mellon University*
awilson | kenh @microsoft.com, caox | ravin @dgp.toronto.edu, scott.hudson@cs.cmu.edu

Abstract

Many interactive surfaces have the ability to detect the shape of hands or objects placed on them. However, shape information is typically either condensed to individual contact points or categorized as discrete gestures. This does not leverage the full expressiveness of touch input, thus limits the actions users can perform in interactive applications. We present ShapeTouch, an exploration of interactions that directly utilize the contact shape on interactive surfaces to manipulations of objects and interactors. ShapeTouch infers virtual contact forces from contact regions and motion to enable interaction with virtual objects in ways that draw upon users' everyday experiences of interacting with real physical objects.

1. Introduction

Interactive surfaces allow users to manipulate information by directly touching them, thus enabling natural interaction styles and applications. The ability to interact directly with virtual objects presented on an interactive surface suggests models of interaction based on how we interact with objects in the real world. When people interact with physical objects, the shape and size of the contact regions plays an important role in determining the actions that are possible to perform with the objects. For example, we handle objects in different ways depending on the task: delicate vs. coarse manipulation of an object to achieve different goals (Figure 1a), using a finger to operate local controls and a whole hand to move the entire object (Figure 1b), or changing hand posture to accommodate different object shapes and layouts (Figure 1c). Various physical objects and tasks lead to a rich set of manual skills for grasping and manipulating objects [13].

Although many interactive surfaces [5, 23] sense the shape of the contact region, very few fully exploit the rich information embedded in hand shape for interaction. Most current systems [8, 21] inherit the "cursor" metaphor from desktop interfaces and interpret the input as one or more discrete points of contact, while discarding the contact shape and size. The result is often a simulacrum of the desktop computer interaction style that "uses the finger as a mouse". Some research explores gesture-based interaction by recognizing hand shapes and trajectories as gestures to trigger commands [20, 24]. This approach goes beyond the cursor metaphor and supports a richer set of actions, but it requires the design and definition of explicit gestures for each function in each application. It also does not allow input actions beyond those explicitly designed into the gesture set.

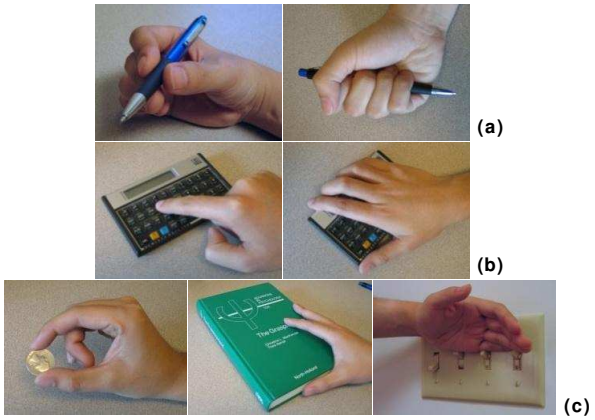

Figure 1. Various ways to manipulate physical objects.

We aim to explore direct-touch interaction that fully utilizes contact shape information on an interactive surface. In contrast to previous approaches that abstract the rich input into contact points or discrete gestures, our approach keeps all system inputs as dynamic contact regions that preserve the full expressiveness of shape input. Interaction mechanisms use information from these input regions directly in manipulations of virtual objects or interactors. The resulting interactions are consistent with, or inspired by, physical manipulation of real objects. To enable this, we choose to employ a special localized input handling process. Properties of the input regions, such as shape, size and motion, could all play a role in the interaction effects. Many of these mechanisms are driven by an emulation of physical contact forces on virtual objects such as pressing, collision and friction. The analogy between our mechanisms and the physical world may allow users to leverage existing skills in handling physical objects. Further, users may combine some of these generic actions to create higher level manipulations (e.g. bimanual operations) not explicitly defined beforehand.

978-1-4244-2897-7/08/$25.00

2. Previous work

Advances in technology have made large display surfaces supporting direct-touch input available. Such direct-touch interactive surfaces may take various form factors such as horizontal tabletops (DiamondTouch [5], Microsoft Surface[1], Philips Entertaible[2]), vertical large displays (SMARTboard [3]) or other flexible setups (Perceptive Pixel [4]). These provide contact shape information of varying fidelity, depending on the technologies employed.

Many researchers explored interaction techniques on direct-touch surfaces. Both single- and multi-user scenarios have been investigated on interactive tabletops [14, 21], with one or several discrete input points from each user, typically simulating pointing with mouse cursors. Others explored using multiple contact points together to enrich interaction, such as scaling and rotating objects [15], manipulating complex shapes [10], or enabling precise selection [3]. Researchers have also explored gesture-based interaction on direct-touch surfaces, interpreting the shape or dynamics of human hands as commands [20, 24]. Both the "multi-touch" and "gestural" approaches condense contact shapes into more abstract forms (points or gestures) for input.

In contrast, our approach avoids unnecessary early abstraction, and directly considers the full properties of the input regions. Therefore, we can fully leverage the rich expressiveness of shape input, support more realistic physical-style manipulations, and do not necessarily require predefined gestures. Krueger [12] explores interactions that employ silhouettes of the user's hands or body, but these were not direct touch techniques. SmartSkin [19] briefly explored using the hand contour to "drive" objects away, but otherwise focuses on point and gesture-based interactions. In contrast, Wilson [23] utilize the motion field of the hands (in his case to move objects), an idea that we adopt to develop many of our techniques.

Several other systems explore interactions with virtual objects based on behaviors of physical objects. BumpTop [1] is a virtual desktop manipulated with a pen or finger using actions such as tossing and piling objects. "Alignment stick" [16] is a direct manipulation interactor for aligning objects by pushing them. Geißler [7] presents a pen technique for throwing virtual objects over a long distance. Beaudouin-Lafon [2] and Dragicevic [6] present techniques to fold desktop windows using a paper metaphor. All these techniques use a single mouse, pen or finger for input. We provide a complementary extension to these approaches and explores the role of contact shape in a rich set of physically inspired operations.

[1] www.microsoft.com/surface
[2] www.research.phillips.com
[3] www.smarttech.com
[4] www.perceptivepixel.com

3. Prototype platform and input handling

We prototype our designs on a computer-vision-based interactive tabletop (Figure 2a), with a rear-projected surface measuring 60 x 45cm. Virtual objects or interface components can be displayed on the surface. An infrared camera beneath the surface captures the image of human hands or other physical objects on it. The system input is a grayscale image of the surface and the objects resting on or near it, with higher brightness corresponding to objects in contact with the surface. Contact regions due to hands or other physical objects are treated equally by the system.

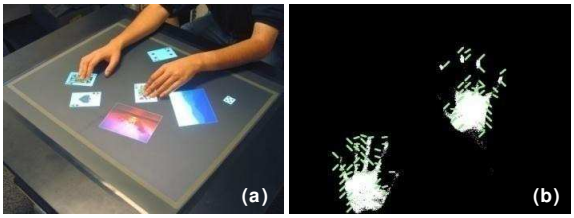

Figure 2. Prototype platform. (a) System in use.
(b) Contact shapes overlaid with motion vectors.

We obtain the exact shapes of the contact regions by thresholding the input image. In addition, we extract motion vectors throughout the surface by calculating an optical flow field on the input image, using a simple block-matching algorithm [23] between input frames (Figure 2b). Both the contact regions and the motion vectors are used as input for the interaction. Rather than discretizing the input as a set of points or blobs as many systems do, we use full shape and motion information throughout.

In general, most interactive systems first interpret the input into abstract events or commands such as cursor movements or recognized gestures, and then pass them to objects in focus (typically determined by pre-selection or hit-testing). This reflects the assumption of a single or discrete set of centralized inputs. However, in the physical world there is no such centralized control mechanism or notion of interaction focus. Every physical object (or its subparts) continuously responds to all actions applied to it. The result is a dynamic world where multiple actions on different objects can occur concurrently and interact with each other.

In order to fully utilize the rich continuous input from the system, and create an experience similar to interacting with real physical objects, we respond to the input in a manner similar to that in the physical world. Instead of receiving centralized events from the system, every virtual object continuously monitors contact regions and motions within its locality, and responds accordingly. Where applicable, these objects may also interact with each other when they contact. Thus, the system can naturally support multiple simultaneous actions or one action on several objects. In addition, this localized, shape- and motion-based approach avoids the common difficulties of reliably tracking discrete contact points/blobs, typically faced by other direct-touch systems.

4. Virtual force metaphor

To map shape and motion input onto virtual objects in a manner resembling physical behaviors, we use a metaphor of virtual contact forces inspired by how people naturally exert different forces to manipulate physical objects. In the physical world, the type and amount of force people apply to an object determines how it responds. It might be beneficial for virtual objects to respond to forces in a similar manner. Most interactive surfaces do not sense forces directly, with few exceptions [9, 22] that detect input pressures only. However, studies of physical manipulations suggest that people leverage different amounts of contact area to apply different amounts of force [13]. For example, they use fingertips to handle light objects, but the whole hand to move a heavy object. People also use different amounts of contact for delicate adjustment or coarse actions (Figure 1a).

Inspired by this, we infer the amount of "virtual force" (abbreviated as "force") the user applies to a virtual object, based on the contact area (practically calculated as the number of contact pixels) upon the object (Figure 3a, b). We consider the following types of forces (Figure 3c):

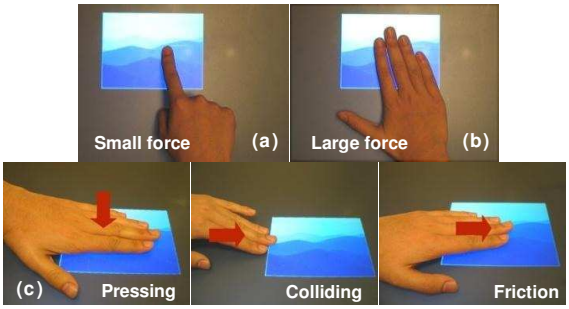

Figure 3. Virtual force. (a) Less contact means smaller force. (b) More contact means larger force. (c) Types of forces.

Pressing: when the user touches upon a virtual object, a pressing force pointing down is applied to it. The amount of the force is proportional to the contact area upon the object (i.e. within the object boundary). Although in the physical world pressure may vary regardless of the contact area, we feel this simplified proportional conversion is intuitive enough for interaction purposes. Example usages include pressing a button, or pinning an object.

Colliding: when the boundary of a contact region moves into contact with the boundary of a virtual object (or inversely, the object moves into the contact region), a colliding force is applied to the object in the direction of the relative motion. Example usages include pushing an object, and creating obstacles to stop an object. The amount of the force is set to be a large constant, to prevent the colliding contact region from intruding into the object.

Friction: when the user touches upon a virtual object, a force due to friction may apply to the object when there is relative motion between the contact region and the object. The direction of the friction corresponds to the relative motion of the contact region, and the amount proportional to the contact area. Since the relative motion vectors may not be equal throughout the contact region (e.g. when the contact region is rotating or deforming), in practice we divide the contact region into a discrete grid and consider the friction in each grid cell according to the motion vector within it. The response of the object results from the accumulated effects of all these elemental friction forces. Depending on the situation, the friction may be used to drag an object (static friction), to slow a moving object (kinetic friction), etc.

The virtual force metaphor is utilized throughout our interaction mechanisms. As we describe these mechanisms, we will demonstrate how the virtual forces affect the object behaviors in various ways. It is important to note that some of these mechanisms could be implemented without shape input or our proposed virtual force metaphor. However, the affordances of shape input enrich their behaviors in ways more similar to their real world counterparts. The virtual force metaphor naturally enables this by considering the direct effects of the contact input on the objects, rather than individually recognizing them as explicit gestures. Thus, users are not constrained to one particular way of performing the actions, and multiple actions can be seamlessly integrated without interfering with each other.

5. Interaction mechanisms

5.1. Object Manipulation

Inspired by how people manipulate physical objects, we support a set of mechanisms for object manipulation that reflect effects of virtual forces. Where appropriate we also consider dynamics of the objects and interactivity between them. Although for simplicity we refer to user input as a "hand", all actions can also be performed using any physical objects contacting the surface.

5.1.1. Dragging and rotating. The user may drag a virtual object by a contact of any size or shape upon the object. Rotating the hand rotates the object at the same time (Figure 4a). Although this action may first appear similar to the widely adopted two-finger technique for simultaneously rotating, scaling and translating objects [15], it is fundamentally different, as the object movement is determined by accumulating all the friction forces on it (implemented by least-squares fitting of the translation and rotation parameters to all motion vectors within the object boundary, similar to [23]). Therefore this action is independent of the number and shape of contact regions over it, thus is insensitive to error-prone finger detection/tracking processes typically used in previous work [15]. For example, an object can be rotated with one palm, with two hands on different sides of it, or any other rotation motion on the object. The user can also easily drag a small object using the whole hand placed on it without the need to aim precisely, an interaction reminiscent of area cursors [11].

Another natural outcome is that users can constrain the movement by pressing an additional static hand/finger on the virtual object as an anchor (Figure 4b). The larger the pressing force, the more constrained (hence smaller) the resulting movement will be. This provides a natural way to

perform coarse vs. fine movement on the object by applying lighter vs. heavier constraints. Wilson [23] describes a similar technique to our dragging and rotating mechanism, but does not consider the effects of static contact, therefore does not support this interesting action.

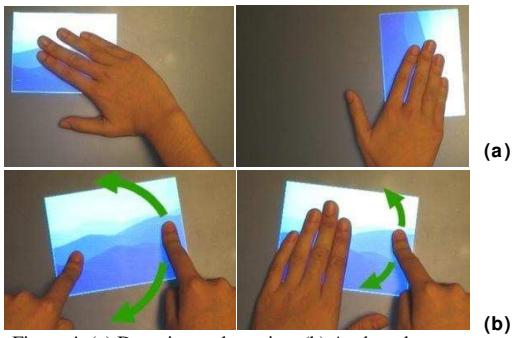

Figure 4. (a) Dragging and rotating. (b) Anchored movement.

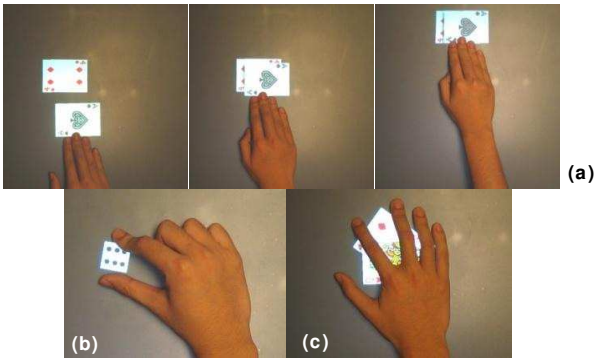

Figure 5. (a) Pushing objects. (b) Pinching an object. (c) Caging objects.

5.1.2. Pushing. The user can also push an object on its side to move it. This results from applying a colliding force to the object. Any hand shape or part can be used to push an object. A single push action can move several objects at once (Figure 5a). This is not easily supported without utilizing the exact shape of the contact regions. The pushing mechanism can implicitly support several other interesting manipulations. For small objects, the user may use two or more fingers to "pinch" it from opposite sides (Figure 5b). The colliding force from either side keeps the object within the fingers when moved. The user may also cage one or several objects in an outstretched palm and move them in unison (Figure 5c). The colliding forces from the fingers keep the objects trapped within the "cage" of fingers. The SmartSkin system [19] supports a similar action by creating a potential field around the hand to repel objects, but our mechanism utilizing virtual colliding forces can result in more precise and realistic pushing actions.

5.1.3. Flicking. If an input contact on a virtual object is suddenly removed while the object is being moved, such as by quickly lifting the dragging hand, the object will continue to move in the current direction and rotate if applicable, as if by inertia. The velocity of the object after flicking is proportional to both its velocity at the moment of flicking (i.e., the velocity of the hand before being removed), and the virtual force applied by the hand. For example, with the same hand velocity, the user can use a finger to cause it to move slowly, or use the whole hand to make it move quickly (Figure 6). After flicking, the object moves with a constant deceleration rate that emulates the friction on the surface, until it fully stops or hits the boundary of the surface.

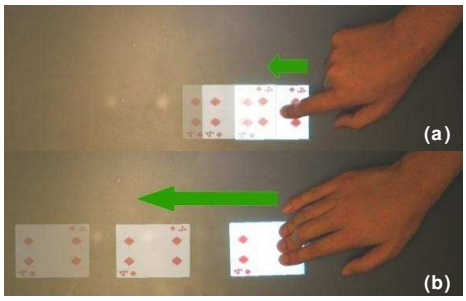

Figure 6. Flicking. (a) Small contact flicks an object slowly. (b) Large contact flicks an object quickly.

The user can stop a flicked object using a colliding force by putting a hand in front of it as an obstacle, or using a friction force by putting a hand on top of it. In the latter case, the deceleration rate is proportional to the amount of friction (i.e., contact area) applied. Thus the object can be stopped instantly (using more contact) or gradually (using less contact). Flicking has been previously explored on interactive tabletops for quickly passing objects [18], however our incorporation of virtual force enables the user to more easily and subtly control the flicking behavior.

5.1.4. Peeling. The faces of virtual objects can be peeled back in a way that mimics how people peel back a piece of paper. To begin peeling back the edge of an object, the user first presses on the object to anchor it, and then moves another hand or finger on one of its edges toward the inside of the object (Figure 7a). The edge folds back to match the peeling movement. Note that without the pressing force for anchoring, the colliding force caused by this movement will push the object away. Further motion on the folded portion of the object causes a virtual friction force to fold it more or less, depending on the movement direction. Pressing statically on the folded portion holds it. When the contact on the folded portion is released, it stays for an additional half second, and then springs back until fully unfolded. The half second delay gives the user time to peel repeatedly (clutching), or operate on objects beneath. Various hand configurations can be employed to perform peeling to suit different situations (Figure 7b).

Multiple stacked objects may be peeled together, either one by one (from top to bottom) or at once, depending on how they are aligned, and where the peeling motion starts (Figure 7c). When the peeled parts are released, they spring back (from bottom to top), giving the user the opportunity to hold the operation at any intermediate state.

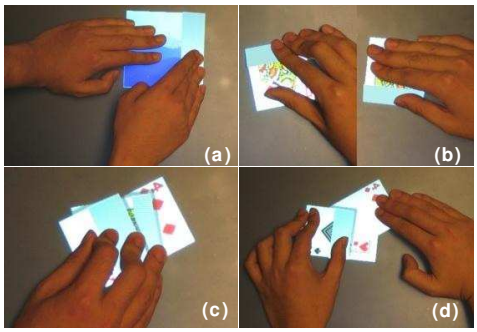

Figure 7. (a) Peeling back the edge of an object. (b) Alternative ways of peeling. (c) Peeling multiple objects. (d) Inserting under a peeled object.

The peeling mechanism is useful in several situations. One example is arranging and ordering overlapped objects. By default, when an object moves towards others, it moves on top of them. However, when an object is peeled, other objects moving towards it will be inserted beneath (Figure 7d). The user can also peel back an object to access the objects beneath it. These actions help in quickly rearranging a stack of objects. With the ability to peel back one or multiple objects at a time, users can easily access every object in the stack, indicate precisely where to insert an object into the stack, or drag an object out of the stack. An object may also be peeled to quickly glance at the object hidden below it. Beaudouin-Lafon [2] and Dragicevic [6] present techniques for peeling desktop windows using a single cursor, however the affordances of shape input and bimanual operation enable much richer peeling actions.

5.1.5. Pinning. A very large pressing force (typically using the whole hand) on the object pins it (Figure 8). When an object is pinned, other objects moving towards it will stop when they collide with it, thus cannot move over or below it. A large colliding force is applied to prevent them from overlapping. This is useful for aligning objects side by side, or moving objects close to one another without overlapping.

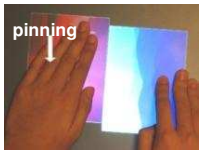

Figure 8. Pinning an object.

5.1.6. Friction between objects. When objects overlap, friction may also occur between them, with the frictional force proportional to the amount of pressing force applied over them. As a result, an object may translate and rotate along with the object on top of it, by a proportional factor in the range of 0~1, determined by the friction (and in turn the pressing force), as illustrated in Figure 9. This is best utilized for manipulating a stack of objects (Figure 10). The user can use a large virtual force to drag the whole stack, or a small force to drag only the object on the top. A medium force fans out the stack in the direction of the hand movement. The reverse movement restores the fanned objects into a stack. Within the force range of the fanning action, the user can choose a larger force to make the objects couple more tightly (hence less spread out), or *vice versa*. Incidentally, the user may also move the whole stack by pushing on its side.

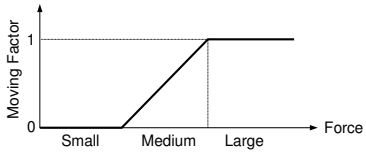

Figure 9. Proportional moving factor by pressing force.

Figure 10. (a) Large force drags the whole stack.
(b) Small force drags the object on top. (c) Medium force fans the stack.

Without deliberate design, these mechanisms can be combined and performed on multiple objects concurrently, resulting in a casual interaction style. Users can apply many existing skills used in manipulating physical objects, such as using both hands to quickly sweep and collect scattered objects into a pile (Figure 11a). Users may also use physical objects or tools to facilitate manipulation, such as using a ruler to sweep objects, a box to keep and move a group of objects together, or a cup to roll a die inside it (Figure 11b).

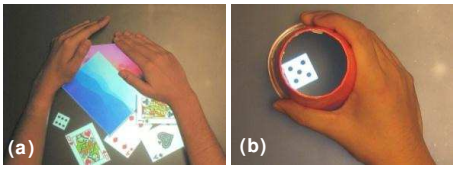

Figure 11. (a) Piling objects. (b) Rolling a die inside a cup.

5.2. Control interactors

For higher level operations, shape input enables us to augment the behavior of common control interactors, and to explore novel designs. These interactors may be local controls on virtual objects, or standalone interface elements.

Standard GUI controls respond to a single cursor input, thus usually only one control can be operated ("in focus") at a time. Since we are not constrained by the notion of a single point of input (consequently a single interaction focus), in our design, control interactors respond independently to the inputs in their locality, utilizing the concept of virtual forces. Multiple interactors can thus be operated concurrently.

5.2.1. Button. A button is pressed when the pressing force upon it exceeds a certain threshold (Figure 12a). Multiple buttons can be pressed at the same time using either multiple fingers or the whole hand over all of them, useful for triggering multiple commands at once, or chording input [4].

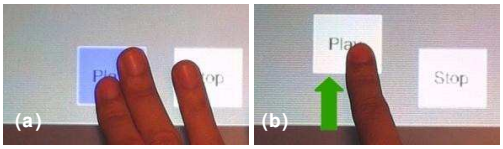

Figure 12. Button. (a) Pressed with a larger force.
(b) Moved with a smaller force.

Where desirable, we can move the location of a button (by dragging or pushing) using a smaller virtual force, just like manipulating virtual objects (Figure 12b). This enables the user to dynamically customize the interface layout without resorting to a special customization mode or dialog, which could be particularly useful for concurrently operating multiple controls with a comfortable hand configuration.

In addition to buttons that respond as long as enough pressing force is applied, we can also create specially shaped buttons that only respond when the contact shape closely matches theirs. This implicitly requires a special hand posture to press the button, preventing unintended activation, which can be useful for critical operations. For example, an "L" shaped button can only be activated by putting a similarly shaped hand on it. (Figure 13). Alternatively, a special physical object can be used as the "key" to press a button that matches its shape. A possible extension is to let users to customize the shape of such a button by taking a snapshot of their hand posture as the template.

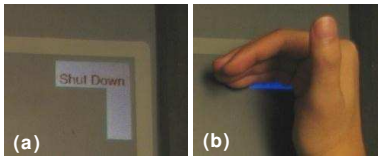

Figure 13. (a) Specially shaped button
(b) Pressed with a special hand posture.

5.2.2. Slider. Sliders are used to adjust continuous parameters. The knob of a slider can be moved by either dragging or pushing it. Multiple sliders in a row can be adjusted simultaneously, either using multiple fingers to adjust them independently, or using the whole hand to swipe all of them together (Figure 14). The user can easily keep the relative positions of the slider knobs (or more specifically keep them at the same value) while adjusting them. This can be done with either a swiping motion, or moving the whole hand with each finger on a different slider. In a sense, the user is now adjusting a dynamically defined compound slider that might control a higher level parameter. For example, on an audio mixer the user may adjust the volume of each channel separately, or move all sliders together to adjust the overall volume. On a color picker, the user may change the red, green and blue (RGB) values separately, or move all three sliders together to change the brightness only. More specifically, swiping with a hand perpendicular to the sliders keeps the RGB values equal, choosing a grayscale color. The user can also quickly swipe all sliders to the maximum or minimum value, or use various hand postures to form specific arrangement of the sliders. These actions are commonly observed when people operate physical sliders. Pressing the arrow buttons on the slider's sides also allows adjustment, with speed proportional to the pressing force.

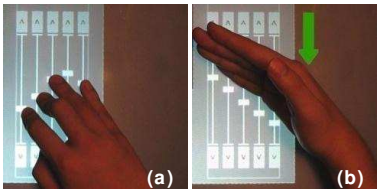

Figure 14. Operating multiple sliders.
(a) Using multiple fingers. (b) Using the whole hand.

5.2.3. Expandable interactor. As an example of less traditional designs, an expandable interactor can reveal more detailed options by a "blooming" action (essentially centrifugal friction forces) on it. Figure 15 illustrates a button for a higher level command (in this example for automatically adjusting an image), which can be expanded into three sub-buttons (for adjusting contrast, brightness, and color respectively). Conversely, a shrink action (centripetal forces) restores it to the original representation.

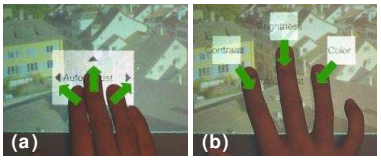

Figure 15. Expandable interactor. (a) Original. (b) Expanded.

Users may also utilize physical objects to operate the interactors. For example, a physical "weight" can be put on a button to keep it pressed down, potentially converting a momentary button into a toggle switch. A small physical object (e.g. chess piece) can be placed on a slider knob, and then moved as a physical proxy for it. Elongated objects (e.g. ruler) can associate with multiple slider knobs and move them together, and specially shaped physical templates could be used to quickly create specific arrangements of sliders.

5.3. Types of operation

The amount of virtual force can also be used to differentiate the between intended operations, enabling fluid selection and smooth transition between different operations.

5.3.1. Scope. Where there is an ambiguity, the amount of force can be used to indicate the scope of the operation (local vs. global). For an object that supports local operations (such as operating controls on the object), the user can use a smaller force to perform them without moving the object itself. Conversely, the user can use a larger force (typically with the whole hand) to manipulate the entire object in the various ways we described earlier, without triggering local operations (Figure 16). Alternatively, the user can push the entire object by the side. These are consistent with physical world conventions (Figure 1b).

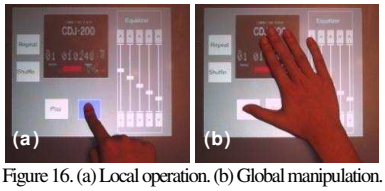

Figure 16. (a) Local operation. (b) Global manipulation.

5.3.2. Functionality. Different amounts of virtual force may also map to different functionalities. We demonstrate this on objects with scrollable content (e.g. document or list-box) to distinguish between scrolling and local operation on the content. A large force on a scrollable object manipulates it as a whole. To scroll its content, the user can either use the scrollbar (slider) on the side, or apply a medium force on the content to directly drag it (Figure 17a). Similar to flicking an object, the content can also be flicked (and stopped) at different speeds, modulated by the force applied.

A small force (typically with a single finger) is used to perform operations local to the content. On a list-box, the user uses a finger to select an item, and can slide the finger through the items to change selection without scrolling the content (Figure 17b). For a document, a single finger acts as a pen to annotate on its content (Figure 17c).

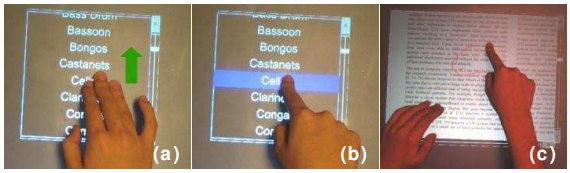

Figure 17. (a) Scrolling. (b) Selecting an item. (c) Annotating a document.

Users can smoothly transition between these functionalities by simply increasing or decreasing the contact area without lifting the hand. Different operations may also be divided between hands, such as using the non-dominant hand to scroll a document, and using the dominant hand to annotate. The user can also use a physical stylus to annotate.

6. Initial user feedback

Given our research focus on exploring design concepts rather than technical implementations, our current prototype was realized using the simplest available algorithms without significant effort expended on performance optimization. Thus, the system's precision is somewhat imperfect (e.g., the drift problem discussed by Wilson [23] especially in the center of the palm where not enough texture is available for reliable optical flow calculation), and much can be improved. Nonetheless, it is important to get some an informal qualitative user feedback on our prototype, leaving formal evaluation of the techniques for future studies.

Five university students participated. None had used interactive tabletops before. Each session lasted about an hour. They were first asked to freely explore the prototype without instructions for 2 minutes, to observe whether they could self-discover the basic manipulation mechanisms. Then features were progressively introduced to the user, starting from basic concepts including the idea of emulating physical manipulations and the virtual force metaphor, to more advanced techniques. Between and after these explanations participants were given time to freely explore and discover unexplained techniques by themselves. The participants were asked to perform a set of simple tasks, including: sorting a stack of virtual playing cards; selecting items in a scrollable list; tuning an image's brightness using multiple sliders; and making annotations in a document.

Initially, participants used a single finger like a cursor to drag the virtual objects. Some even attempted to double-click on the objects, as they would with a mouse. Some tried two-finger rotation on the objects, as perhaps learned from products such as the iPhone™. Only one (the youngest) participant discovered our physical-like manipulations in a 2-minute exploration period. The interaction style from desktop computers seems to restrict the imagination of many users!

However, once the system's concepts were explained, all participants abandoned cursor-like behaviors, and were able to grasp the interaction mechanisms quickly, either by discovering by themselves, or with simple instructions. They also commented that the mechanisms were straightforward and easy to understand, and "the virtual force metaphor worked nicely". Participants manipulated virtual objects in ways similar to manipulating physical objects, especially in using both hands and arms to quickly arrange objects. Various techniques, resembling real world actions, were used to complete the designated tasks. For example, to sort a stack of cards, 2 participants used the peeling action, while 3 others preferred spreading all the cards out to restack them.

Participants liked using physical tools to interact with the system and spent time using objects at hand. Physical objects were also used in unexpected ways, such as using the tip of a chess piece to select a list-box item, or using a physical object to anchor a virtual one for peeling or rotating.

Subjective feedback indicated that participants liked the physical feel of the manipulations, and the ability to utilize their hands rather than relying on single fingers. The ability to operate multiple controls at once, especially swiping multiple sliders, was appealing. Other liked features include flicking and peeling (one person especially liked peeling an object to hide something below it), the magic lens defined by hand shape, and the slider's force-sensitive arrow buttons.

The evaluation also highlighted a few problems. Performance issues caused the system to not always respond as expected. Participants felt that the lack of visual feedback indicating virtual force also caused some uncertainty in the resulting actions. During the global manipulation of an object, a transient smaller virtual force is sometimes caused by the hand lifting on or off the object, resulting in unintended local operations. This may be addressed by filtering techniques, but we must be careful about possibly introducing undesirable lag. One user tried using a pressing force to lower a virtual object so that other objects could be moved over it, but pinning behavior resulted instead.

Participants also suggested features they would like to see added. Visual feedback about the virtual force was widely desired. They also wanted to use one hand to hold

an object by pinning or pinching, and the other hand to perform local operations. Some participants expected friction to depend on the material properties of the virtual object. Others suggested rendering stacked objects in a way that indicates the height of the stack. One participant suggested self-revealing cues to help novice users discover the new interaction paradigm.

7. Discussion and conclusion

The virtual force metaphor allows the user to impart more or less force by changing the amount of contact area. We believe that this captures some of our intuitions regarding how people apply forces physically. It is interesting to consider the possibility of interactive surfaces that sense pressure directly. Such capability has been utilized in pen interfaces [17], where contact shape cannot be changed. With a pressure sensitive touch surface, the user could use different pressures to indicate that a palm placed on an object is to be used as a movement constraint, or to hold the object and enable local operation with another hand.

Visualization of the virtual forces (and the resulting actions) was desired by most study participants. One idea is to display ripples caused by the force on the objects, as if they were made of rubber sheets. The shape, location and size of the ripples could be used to present the type, direction and amount of the force. Another possibility is to give the objects a thin 3D look, and enable them to "tilt" slightly on the surface depending on the location and amount of force applied, as if they were pivoted at the center.

One of our goals is to allow interaction with virtual objects as if they were tangible physical objects. However, the lack of true haptic feedback limits the degree to which users may rely on their intuitions in interacting with virtual objects. The addition of vibro-tactile output techniques may provide useful cues to collisions and pressing buttons for example, but is unlikely to fully recreate the feeling of a real physical object on the surface. On the other hand, the ability to sense real objects on the surface adds an element of tangible interaction, such as when physical tools or props are used to interact with virtual objects.

In summary, we have presented an exploration of interaction mechanisms that leverage the contact shape input on direct-touch interactive surfaces. Building upon the metaphor of virtual forces, ShapeTouch enables interactions that mimic those used in the physical world and beyond. Initial evaluation shows that ShapeTouch creates a natural and intuitive interaction style.

8. Acknowledgements

We thank Patrick Baudisch, Meredith Morris and other colleagues at Microsoft Research for valuable comments and discussions.

9. References

1. Agarawala, A. and Balakrishnan, R. (2006). Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. *CHI*. p. 1283-1292.
2. Beaudouin-Lafon, M. (2001). Novel interaction techniques for overlapping windows. *UIST*. p. 152-154.
3. Benko, H., Wilson, A.D., and Baudisch, P. (2006). Precise selection techniques for multi-touch screens. *CHI*. p. 1263-1272
4. Conrad, R. and Longman, D. (1965). Standard typewriter versus chord keyboard: An experimental comparison. *Ergonomics*, 8. p. 77-88.
5. Dietz, P. and Leigh, D. (2001). DiamondTouch: a multi-user touch technology. *UIST*. p. 219-226.
6. Dragicevic, P. (2004). Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. *UIST*. p. 193-196.
7. Geißler, J. (1998). Shuffle, throw or take it! working efficiently with an interactive wall. *Extended Abstracts of CHI*. p. 265-266
8. Han, J.Y. (2005). Low-cost multi-touch sensing through frustrated total internal reflection. *UIST*. p. 115-118.
9. Herot, C. and Weinzapfel, G. (1978). One-point touch input of vector information for computer displays. *SIGGRAPH*. p. 210-216.
10. Igarashi, T., Moscovich, T., and Hughes, J.F. (2005). As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3). p. 1134-1141.
11. Kabbash, P. and Buxton, W. (1995). The "Prince" technique: Fitts' law and selection using area cursors. *CHI*. p. 273-279.
12. Krueger, M. (1991). Artificial Reality II: Addison-Wesley.
13. MacKenzie, C.L. and Iberall, T. (1994). The grasping hand. Amsterdam, Netherlands: North Holland.
14. Morris, M.R., Paepcke, A., Winograd, T., and Stamberger, J. (2006). TeamTag: exploring centralized versus replicated controls for co-located tabletop groupware. *CHI*. p. 1273-1282.
15. Moscovich, T. and Hughes, J.F. (2006). Multi-finger cursor techniques. *Proceedings of Graphics Interface 2006*. p. 1-7.
16. Raisamo, R. and Raiha, K.-J. (1996). A new direct manipulation technique for aligning objects in drawing programs. *UIST*. p. 157-164.
17. Ramos, G., Boulos, M., and Balakrishnan, R. (2004). Pressure widgets. *CHI*. p. 487-494.
18. Reetz, A., Gutwin, C., Stach, T., Nacenta, M., and Subramanian, S. (2006). Superflick: a natural and efficient technique for long-distance object placement on digital tables. *Graphics Interface 2006*. p. 163-170.
19. Rekimoto, J. (2002). SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. *CHI*. p. 113-120.
20. Ringel, M., Berg, H., Jin, Y., and Winograd, T. (2001). Barehands: implement-free interaction with a wall-mounted display. *CHI (Extended Abstracts)*. p. 367-368.
21. Shen, C., Vernier, F., Forlines, C., and Ringel, M. (2004). DiamondSpin: An extensible toolkit for around the table interaction. *CHI*. p. 167-174.
22. So, E., Zhang, H., and Guan, Y. (1999). Sensing Contact Shape with Analog Resistive Technology. IEEE International Conference on Systems, Man, and Cybernetics. p. 806-811.
23. Wilson, A.D. (2005). PlayAnywhere: a compact interactive tabletop projection-vision system. Proceedings of the 18th annual *UIST*. p. 83-92.
24. Wu, M. and Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. *UIST*. p. 193-202.