Hands-On Math: A page-based multi-touch and pen desktop for technical work and problem solving

Robert Zeleznik, Andrew Bragdon, Ferdi Adeputra, Hsu-Sheng Ko
Brown University, Providence, RI, USA
{bcz,acb, fadeputr, hsusheng}@cs.brown.edu

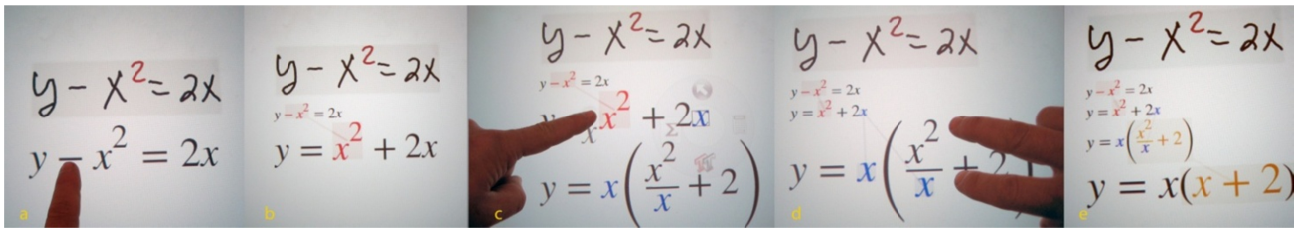

Figure 1. Math transformation. a) Dragging -x^2 across the equality. b) Result with terms highlighted. c) Dragging x (from 2x) across other right-hand side terms factors it and interactively updates the result below. D) Squeezing x^2/x simplifies it. e) The final expression.

ABSTRACT
Students, scientists and engineers have to choose between the flexible, free-form input of pencil and paper and the computational power of Computer Algebra Systems (CAS) when solving mathematical problems. Hands-On Math is a multi-touch and pen-based system which attempts to unify these approaches by providing virtual paper that is enhanced to recognize mathematical notations as a means of providing *in situ* access to CAS functionality. Pages can be created and organized on a large pannable desktop, and mathematical expressions can be computed, graphed and manipulated using a set of uni- and bi-manual interactions which facilitate rapid exploration by eliminating tedious and error prone transcription tasks. Analysis of a qualitative pilot evaluation indicates the potential of our approach and highlights usability issues with the novel techniques used.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: gestures, multi-touch, stylus, pages, paper, math.

INTRODUCTION
Problem solving with Computer Algebra Aystems (CAS) and with pencil and paper each has a number of strengths and weaknesses, and perhaps as a result, students, scientists and engineers use both. Paper is inherently bimanual [11], fluid and open-ended. Paper encourages step-by-step computation, affording insight at the expense of tedious, error-prone mental computation. Alternatively, CAS tools efficiently provide answers at the cost of complex UIs which are highly syntactic, rigid and linear. CAS distances the user from key aspects of the problem solving process [7], such as choosing alternative computational paths or understanding computational logic. Transcribing between paper and CAS provides tedious, error-prone integration but disrupts creative problem-solving. Thus, CAS is often underused, not used to its full potential, or foregone entirely [7].

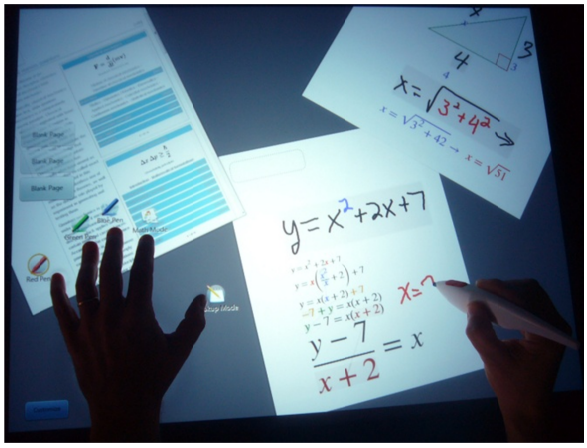

Figure 2. Document and note pages showing recognized math and embedded computation.

Pen and touch input modalities naturally complement one another, and have the promise to form the basis for a paper-like bimanual system [13]. Therefore, our approach is to create Hands-On Math, a virtual paper/CAS hybrid, which attempts to combine the fluid, bimanual advantages of physical pages with the computational power of CAS. Our core hypothesis is that if CAS tools were driven by direct, multi-touch manipulation and digital ink within a free-form note-taking environment, students and even scientists and engineers might learn and work more efficiently. We note that virtual objects/pages are not identical to physical ones[27], but we hypothesize that for paper they are similar enough for many of the benefits to carry over. Hands-On Math's current functionality limits it to high school-level math, but we believe the approach may apply broadly across domains and users, and support collaborative work.

UIST'10, October 3–6, 2010, New York, New York, USA.

Since it is not yet clear how bimanual techniques should be designed or employed, our approach in this paper is to push in a number of directions to see which techniques work out best in the context of a real system. This approach has a number of advantages: rather than evaluating techniques in isolation, we can see how they interact in a full system, and test this functional system with domain users.

The contributions of this paper are:

- The design of a virtual paper/CAS hybrid prototype that attempts to expose CAS tools fluidly as bimanual pen and touch operations (Figure 1).
- The design of several synergistic pen and touch techniques that collectively comprise a functionally rich page-based system capable of supporting domain users performing representative tasks (Figure 2).
- A qualitative pilot evaluation and usability discussion of the prototype system as a whole, and the novel bimanual pen and touch techniques developed for it.

RELATED WORK

There have been several efforts to furnish CAS engines with a handwriting-based front-end. The MathPad^2 system focused on a gesture-based UI for creating animated drawings driven by handwritten mathematical expressions but also included several CAS commands for taking derivatives, integrating, and solving multi-variate equations[19]. The MathPaper system, in addition to providing a gestural UI for simplifying and solving equations, also allowed access to the full functionality of Mathematica by recognizing its 1D syntax[31]. MathBrush, alternatively, recognized handwritten mathematics and then provided a menu-based UI to CAS functionality[18]. Thimbleby explored a novel calculator UI[28] in which handwritten input is computationally enhanced as it is written to always show a valid equality. Our work differs from these works at both a user interface and a system level: we use multi-touch and pen to provide a general-purpose enhanced virtual paper metaphor and we support a multi-touch interface for interactively manipulating and transforming mathematical expressions.

Anthony's tutoring work[1] is complementary to ours; for instance, virtual pages could be used to present structured tutoring materials, and multi-touch math manipulation could support additional tutoring interactions.

Also related are systems which explore various page-based interactions. WeSearch provides a multi-touch web search interface in which pages can be clipped and organized on a virtual desktop[23]. Other work explores pressure-sensitive surface techniques for arranging 2D paper-like objects, for example, to slide one page underneath another, or to peel back a page corner to see obscured content[8]. Integrating this work is a complementary research effort.

In addition, there is a large body of work on pen-based digital notetaking techniques, including commercial products such as Microsoft OneNote which even supports primitive calculator functionality and handwritten mathematics recognition, but no real CAS features. InkSeine is a notable research system that provides a virtual notebook with an *in situ* pen-based document and web search component[15]. We adopt a basic form of InkSeine's web searching bread-crumb technique to harvest page clippings.

Recent research has focused on hybrid touch and pen interactions. Brandl *et al* examine dynamically altering pen characteristics when the non-dominant hand is placed flat on the drawing surface, or bi-manually controlling widgets such as an undo history tool[6]. In addition, they explored bi-manual interactions that seamlessly transition from menu selection to direct manipulation, for example, to perform a bi-manual rectangular selection. Our related TAP gestures, however, blend into a virtual page metaphor where it is not necessarily desirable, for example, to always display a menu given a non-dominant hand finger contact or to use a menu-based interaction over a naturally parameterized, unambiguous gesture. Hinckley *et al* explored using touch to define a context for interpreting subsequent pen strokes, for example, to hold an object with touch and stroke off a copy with the pen[14]. Instead, we recognize partial pen gestures and display visual feedforward that can be dragged to confirm a TAP gesture. Frisch, et al also explore pen and touch, but not gestures requiring a combination of both[10].

OVERVIEW

Joining mathematical computation with enhanced virtual paper extends the familiar paper-and-pencil interaction paradigm to improve workflow and assist conceptual understanding. Specifically, virtual paper facilitates the encapsulation of work products into page-like entities, which, unlike physical paper, can support ink-driven computations and can be arranged on a large, pannable virtual desktop. By recognizing handwritten mathematical expressions and converting them to manipulable typeset notations, virtual paper can obviate a range of tedious and error prone interactions, such as when manually performing steps in a derivation. For example, algebraic transformations can be executed by pinching, stretching and dragging terms within an expression. Hands-On Math further provides a suite of gestural techniques for creating and manipulating pages, annotations, diagrams and web clippings. The unifying characteristic of these techniques is how pen and touch combine to produce efficient interactions that can be robustly recognized without physical buttons, pauses or other delimeters.

Hands-On Math runs on a Microsoft Surface equipped with an infrared lightpen that can be distinguished from multi-touch input because of its brightness. Hands-On Math uses the StarPad SDK[16] to recognize and convert handwritten math to typeset notations, gesturally create graph windows, and trigger symbolic and numeric computations with extended notations. Web queries, written as ink, return annotatable, live web pages. Although Hands-On Math is fully-functional, it is incomplete since it does not expose any disclosure techniques for learning multi-touch or pen gestures. However, our preliminary prototypes indicate that a customized GestureBar [5] may be sufficient.

In the next three sections, we present the novel UI design

components of Hands-On Math. First we describe the interface to a page-based environment that was designed to support scalable working sets of documents and notes. Then we describe a set of lightweight, general-purpose mode switching and command invocation gestures designed to robustly augment inking activities. Last, we describe the core multi-touch UI for manipulating and transforming mathematical expressions that are available on all pages.

PAGES AND WORKSPACE

There are two dominant traditional environments for problem solving – paper and pencil and whiteboards – each with its own benefits. Whiteboards are convenient because often entire workflows can be captured without having to perform any spatial management; however, they quickly fill up and cannot be used like a notebook for longer term workflows. Paper on the other hand is non-interactive and can provide no active assistance. Thus, we designed a hybrid tabletop solution that blends whiteboard-like interactions with resizable pages that live on a large virtual desktop. Users can grow existing pages to accommodate complex problems or pan the desktop to make room for new work.

Beyond basic multi-touch interactions for dragging and rotating pages with one or more fingers, we designed three page management interactions, including a virtual desktop with panning bar, bezel gestures to create and delete pages, and a page "folding" gesture to make room for more work.

Page Management

To facilitate an exploratory mindset when problem solving, we wanted to minimize the cost of creating a new virtual page. We dismissed using dedicated buttons on the display because we found them to be distracting and had to be inconveniently placed to avoid accidental triggering. We instead associate page creation with a bezel gesture[26] (Figure 3). Our bezel gestures are parameterized by the number of fingers that cross the bezel and which bezel is crossed. Swiping two fingers through the left or right bezel, as if reaching beyond the desktop for a new sheet of paper, creates a page. Swiping through with one finger pans the desktop. The physical distinctiveness of one versus two fingers along with bezel size enables eyes-free interaction.

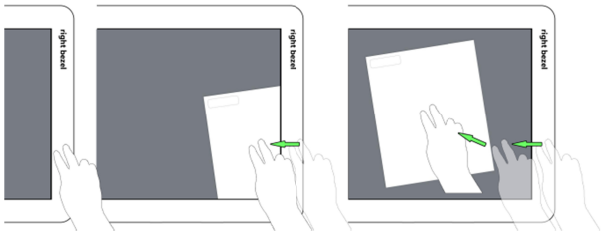

Figure 3. Swiping over bezel with 2 fingers creates a page.

To delete a page, we considered reversing the page creation gestures (i.e., drag a page off the bezel with two fingers) but decided such an interaction might conflict with an intuitive reaction to just move a page "out of the way" without deleting it. Moreover, since deleting pages is less important because of the virtual desktop, we chose a deliberate page deletion gesture (Figure 4): after dragging a page into the bezel region, a trash can icon appears near where the finger contact left the screen; by continuing the drag back across the trashcan, the page is deleted.

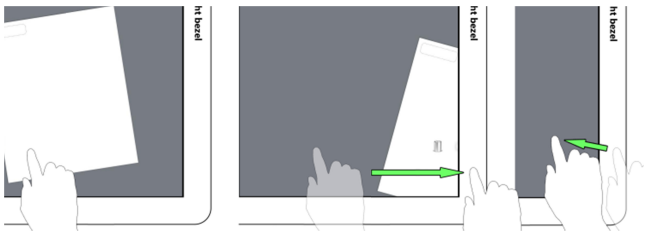

Figure 4. Dragging over bezel displays trashcan widget.

Since pages can be manipulated with one or two fingers, they are subject to accidental manipulation, particularly when writing, since the palm of the writing hand may be mistaken as a finger contact. The palm rejection technique of ignoring large contacts partially addresses this problem, however, users may also explicitly hold a page steady with two or more fingers from their non-dominant hand while writing – similar to how people control physical pages [11].

Panning Bar

To free users from having to worry about what to get rid of in order to make room for new material, we adapted the notion of a continuous virtual desktop that was previously found useful for desktop code development[4]. Unlike fixed-size desktops, where the clutter of overlapping pages becomes an interaction burden, continuous virtual desktops allow users to create more space on demand while preserving a spatial record of their previous work.

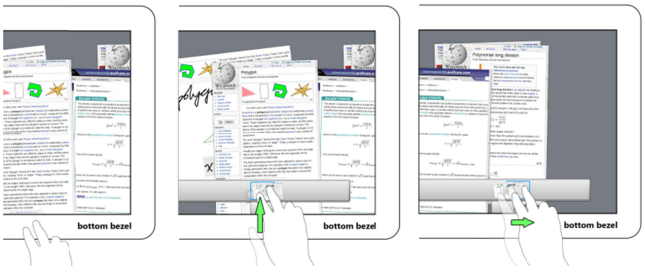

Figure 5. Swiping up through bezel with two fingers shows panning bar; continued dragging pans the desktop.

Although swiping through the bezel with one finger supports fine-grained panning of the virtual desktop, we wanted a complementary technique to facilitate larger scale navigation. After recognizing an upward two-finger swipe through the bottom bezel, we display an interactive panning bar (Figure 5). The panning bar displays a live, miniaturized panorama of the user's workspace that provides a visual overview of the user's work history. The currently visible desktop is always centered under the user's finger when the panning bar appears; as the user drags horizontally along the panning bar, the desktop scrolls correspondingly to that location. While one hand controls scrolling, the other hand can grab pages to relocate them within the virtual workspace. Invoking the panning bar requires a two-finger swipe to reduce the likelihood of unintentional triggering which can occur, for example, when leaning over the surface as accidental contacts are made with an arm, shirt, etc.

Folding

As a complement to page management, we use multi-touch folding gestures to manage space within a page [9]. Thus, for example, a user may choose to fold away part of a complex derivation in order to simultaneously view the problem statement and their current step in the derivation without having to scroll between the two. By pinching in the margin of a page, users interactively simulate a 3D fold in which the page buckles up between their fingers (Figure 6), before collapsing into a suggestive "crease" indicated with a soft shadow. Tapping the shadow line unfolds the page.

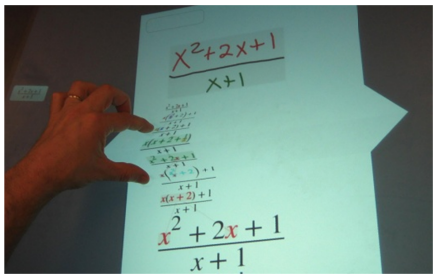

Figure 6. Folding page contents to make space on the page.

Since this technique is intended to support complex work, it does not actually shorten the page the way a real page fold would, but instead causes the page contents to slide up and create open space at the bottom of the page. This operation is akin to code elision techniques in source code editors except it does not operate on syntactic structures.

GESTURES

Inherent ambiguities exist when trying to decide whether input is intended to be ink, a gesture or a direct manipulation. To support fluid command invocation and mode switching, we developed three complementary gesture techniques which have minimal overlap with other activities and thus potentially can be robustly recognized.

Under-the-Rock menus

Under-the-rock menus are a general purpose mechanism for associating contextual actions with display elements. In essence, they are context menus that are "hidden under the rock" only to appear when objects (*i.e.*, rocks) are moved. For example, dragging a term in a mathematical expression might default to a factoring operation; the under the rock menu for that term, however, would allow the interaction to be changed to reordering, term splitting, or something else.

An under-the-rock radial menu grows, after an initial lag, out from an object's initial location as the object is dragged away (Figure 7). Growing a semi-opaque menu from the start of a drag provides unobtrusive disclosure which is critical for not interfering with a default dragging behavior. The menu is only activated and made fully opaque after a second contact is made over its center; menu items can then be selected by sliding the second contact over them. An active menu can be deactivated by sliding the second contact back to the menu center and releasing. Since the menu appears predictably, centered on the drag's starting point, trained users can anticipate this and co-articulate menu selection with dragging before the menu has become fully opaque as with marking menus[17]. Depending on manual dexterity, this interaction can be performed with one hand.

We believe under the rock menus are more appropriate for math manipulation than bi-manual marking menus which require two hands to operate and automatically center an opaque menu over the dragged object[25] which disrupts default dragging interactions. Similarly, tap-based menus can be falsely triggered by inadvertent contacts – even multi-finger tap menus can be triggered pinching and stretching gestures. We also avoided widgets for invoking menus because they add visual clutter without resolving the false activation problem as users by chance may hit a widget when interacting with the backround or nearby object.

Touch-Activated Pen Gestures

Pen gestures are often touted as being well suited for invoking spatially parameterized commands. However, pen gestures are by definition ambiguous with regular inking activities unless distinguished through special hardware buttons (e.g., stylus-mounted, or external buttons). In previous work, a set of pen-only gestures (for scribbling out, selecting, graphing and performing undo and redo) was designed and evaluated to be compatible with writing mathematical expressions [31]. However, in a general inking context, loosely defined pen-only commands, such as lasso selection and scribble deletion, are likely to conflict with regular inking and thus require additional input for disambiguation. The technique of requiring terminal punctuation in the form of a pen tap is workable but comes as pure interaction overhead since the punctuation contributes nothing to the specification of the operation or operand [30].

Hybrid pen and touch gestures, however, can be readily disambiguated from isolated ink and touch activities and enable fluid direct manipulation transitions. Flow/Control menus [12] support such transitions but require a mode switch from inking and require menu selection motions unrelated to the desired direct manipulation.

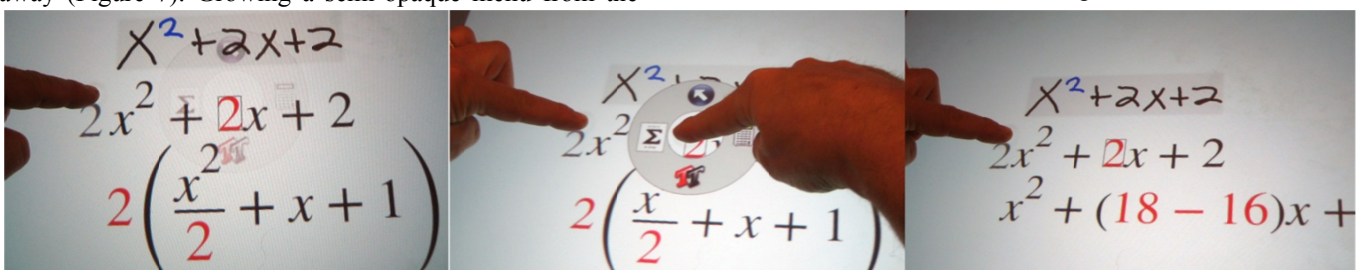

Figure 7. Under-the-rock menus. Dragging the 2 in the 2x factors it from the expression as a semi-opaque radial menu grows. Touching the menu center makes it opaque and activates it. Selecting the ∑ icon switches to a slider that splits the 2 into a sum.

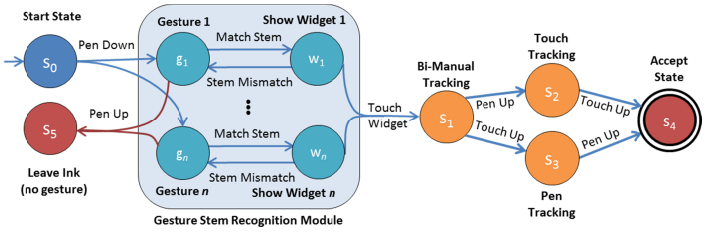

Figure 8. TAP gesture FSA. Multiple gesture stems can match and display widgets, but only one gesture can be fired.

Our solution, touch-activated pen (TAP) gestures, recognizes pen gesture *stems* on-the-fly as they are drawn (Figure 8). A gesture stem is a non-trivial subset of a complete pen gesture. Recognizing a gesture stem performs no command but rather introduces a feedforward widget(s) that can perform a command if triggered by concurrent touch input. If the pen stroke ends without touch input or if the stem no longer matches, the widget disappears. Triggering a TAP gesture with touch input allows fluid transitions to bi-manual, pen-only, or touch-only interactions. False recognition of TAP gestures is unlikely if widgets are positioned away from the hand holding the pen. In addition to improved recognition accuracy, TAP gestures are more scalable than pen-only gestures since a similar or even a single gesture stem can trigger multiple non-overlapping feedforward widgets. To avoid the "noise" of unintended feedfoward during regular inking, all widgets are displayed at reduced opacity until activated by touch input.

Example 1: Making 2D selections

Although users can lasso ink by drawing an enclosed loop, there are times when this is inconvenient or inappropriate such as when selecting large regions of ink or when making a rectangular image clipping. Thus, we support an additional gestural selection technique: as a user draws a crop mark, a semi-transparent dashed rectangle appears in registration with the crop mark (Figure 9). The user can ignore this rectangle and keep drawing, or by dragging a finger across it switch from inking to rubber-banding a rectangular selection marquee. Since the pen and finger-touches control opposite corners of the marquee, its position and size can be adjusted simultaneously [3]. For larger selections, this TAP gesture also provides the benefit of requiring less movement than a corresponding lasso would.

Figure 9. Drawing a crop marks displays a feedforward selection/clipping marquee widget. Dragging the marquee while drawing switches to bi-manual rectangular selection.

Example 2: Inserting space

When problem solving, users often are unable to plan ahead spatially for their future notations and find themselves in need of inserting space between existing notations (and occasionally wanting to remove space). In these cases, they need to specify where and how the space should be created (*e.g.*, vertically or horizontally, across the whole page or locally). We provide a TAP gesture for space insertion (Figure 10): as the user draws a straight line in any orientation, widgets appear on either side of the line starting point for selecting the page contents on either side of the line respectively. If the user ends their ink stroke, the widget disappears. However, if the user touches either widget, the drawn ink becomes a "push-bar". Moving both ends of the push-bar is like manipulating the top (or bottom) corners of a picture frame where the page contents below (or above) the push-bar is the picture. If either contact is released, then the push-bar only moves its contents along one axis.

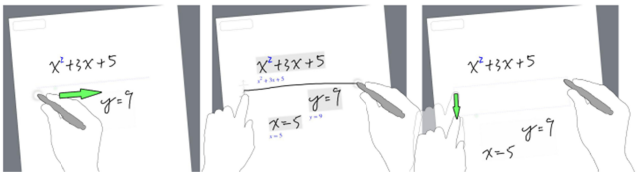

Figure 10. Drawing a line displays an insert space widget. Dragging the widget inserts or removes white space.

Example 3: Clipboard pasting

We also provide a TAP gesture for pasting clipboard contents. As a 'p' is drawn, we display a paste icon. If the user taps this icon, the 'p' disappears and the clipboard contents are pasted. If instead the paste icon is dragged, then the clipboard contents are interactively adjusted to fit within the boundary defined by the pen and touch contacts.

PalmPrints

Although gestures are efficient for executing many types of commands, they require significantly more effort than simply pressing a button. In situations where efficiency is important, such as when switching between different pen colors when drawing a diagram or alternating between math notations and drawing elements, the overhead of performing a path-based gesture (or locating and clicking a toolbar item) can be burdensome. In such cases, having the desired functionality on a button beneath one's fingertips on the non-dominant hand is both efficient and robust [20].

Figure 11. Placing palm and fingers on surface activates PalmPrint menu. Lifting and tapping a finger changes the pen mode. Chording is also possible. A Customize palette allows drag and remapping of functions.

We thus developed PalmPrints which are similar to finger-tapping techniques [22][21] but which instead activate implicitly when an open hand is placed on the surface (Figure 11) and deactivate when it is lifted to do something else. While the users palm rests on the surface, up-down fingertip transition are recognized to invoke a command associated with that finger. Identifying each fingertip is done by sorting the initial five fingertip contacts according to their radial angle relative to the larger palm contact(s). When a fingertip contact is lost, we trivially know which finger was lifted. When all lost contacts are regained (as new contacts), a chord is triggered that executes the commands associated with each finger that had been lifted.

By associating functionality with each fingertip, users can execute commands without looking; however, we display an icon above each fingertip for disclosure (Figure 11). Dragging and dropping functions from a customization palette onto an icon reassigns that fingertip's command.

Alternatively, the user can use their primary hand to "lock" their PalmPrint on the display. If the user then lifts their hand after having locked the PalmPrint, the PalmPrint will transform itself into a five-item toolbar. This toolbar can be dragged with either hand similar to a Toolglass [3]. The toolbar can be dismissed with a tap, or it can be restored to a PalmPrint if the user simply places their hand back down on the surface in the registration pose.

FingerPose

Since touch-and-drag is preferable to the more heavyweight bi-manual PalmPrint for common actions such as dragging a window or panning its contents, we created FingerPose which selects one of two input modes based on vertical finger posture. For comparison purposes, we also implemented multi-finger drag alternatives.

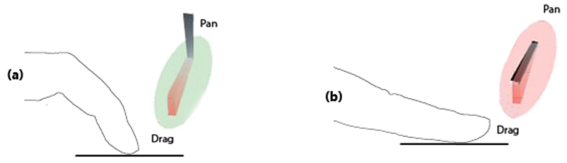

Figure 12. Finger posture selects a drag mode. A widget activated on pause reveals functionality and thresholds.

FingerPose estimates the posture of a finger based on *initial* contact geometry to select between two manipulation functions (Figure 12). By using one-finger FingerPose for window dragging, and content scrolling, and two-finger pinching for zooming, each function is isolated and inadvertent manipulations are unlikely. Alternatively, using two-fingers for both zooming and panning (*e.g.*, MacBook UI) affords seamless transitions and requires less physical dexterity.

FingerPose is similar to approaches which analyze surface contacts to determine finger orientation parallel to a touch surface[29] and to recognize the individual finger poses of a compound simulated button press[2]. However, FingerPose is well suited for repeat interactions and omni-directional dragging since, unlike SimPress[2], it requires no initial finger movement for activation. The fingertip posture is recognized as a contact with a physical area below a calibrated threshold, otherwise a finger pad contact is recognized. Since contact area is dynamic, particularly just after the finger touches the surface, we do not distinguish between poses until after the contact has moved by 25 pixels. In addition, if the contact has not moved after 100ms, we display a dynamic 3D representation of the recognized "posture" and its associated function. Finger posture is ignored after dragging begins.

FingerPose recognition works without calibration which can be problematic for users with small fingers or a soft touch; however it was sufficient for pilot study evaluation. More robust recognition is possible with additional sensing.

MATH

Using existing SDKs [31], we support math-specific interactions for writing mathematical expressions, computing values with extended mathematical notations, and creating graphs with gestures. In addition, we extended this work with a suite of multi-touch interactions for transforming mathematical expressions both to compute solutions and to gain insight on the problem domain and the process of solving problems – concerns that apply equally to students and experienced mathematicians[7]. From a mathematics perspective, our goal was to extend the functionality of previous pen-based math systems to support algebraic manipulations that are fundamental to mathematical reasoning but which are generally hidden by CAS systems. Although pen-based manipulation techniques are possible, we felt that multi-touch interactions were more suitable for the manipulation nature of mathematical transformations. Unlike pen interactions, which inherently conflict with writing mathematics and require a compound selection/manipulation interaction, multi-touch gestures have the promise of integrating selection with manipulation in a single, memorable and efficient physical action. For example, to join two additive terms, users can just pinch them together; a hypothetical pen-based counterpart would need to select terms and indicate the join operation (and not conflict with the entry of new math notations) before initiating direct manipulation feedback. Nonetheless, the set of possible mathematical operations is large, requiring subtlety to increase the expressiveness of multi-touch manipulation.

Fingertip Area Selections

Tapping an expression activates it for manipulation, and increases its font size to facilitate syntax-aware, contact area-based selection[24]; touching a symbol selects it subject to convenience shortcuts based on how the fingertip contact regions intersect mathematical symbols (Figure 13).

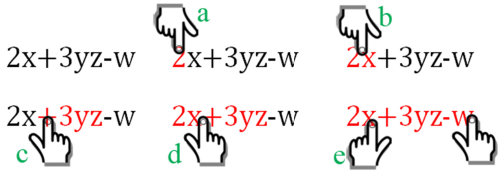

Figure 13. Fingertip area selection: a) one symbol; b) two symbols by touching both; c) term by touching operator; d) neighboring terms by touching part of one and operator; e) a long span.

Touch Manipulations

The interface for performing mathematical transformations consists of sequences of pinch, drag, and stretch direct manipulations of terms within an expression (Figure 1), as summarized in Table 1. The associated algebraic transformations are naturally parameterized by where the terms are dragged. Thus, to preserve context, we do not modify the touched expression directly. Instead, "ghost" copies of just the touched symbols follow the user's finger contacts while the resulting transformed expression interactively updates below the original. However, there are more math transformations than unique input affordances. To avoid the disruption of an explicit mode switching UI, we select different default manipulations based on what was touched. For example, by default dragging an operator reorders the term on its right, whereas, dragging a variable(s) successively factors it from the expressions it passes over. During all manipulations, an under-the-rock menu grows out from the original location of the dragged symbol – when needed, this menu can override the default manipulation. The limited expressivity of touch input makes it easy to start "playing with math" but also demands the adoption of interaction strategies. For example, to simplify x+y+x,one x must be moved next to the other before a pinch can join them.

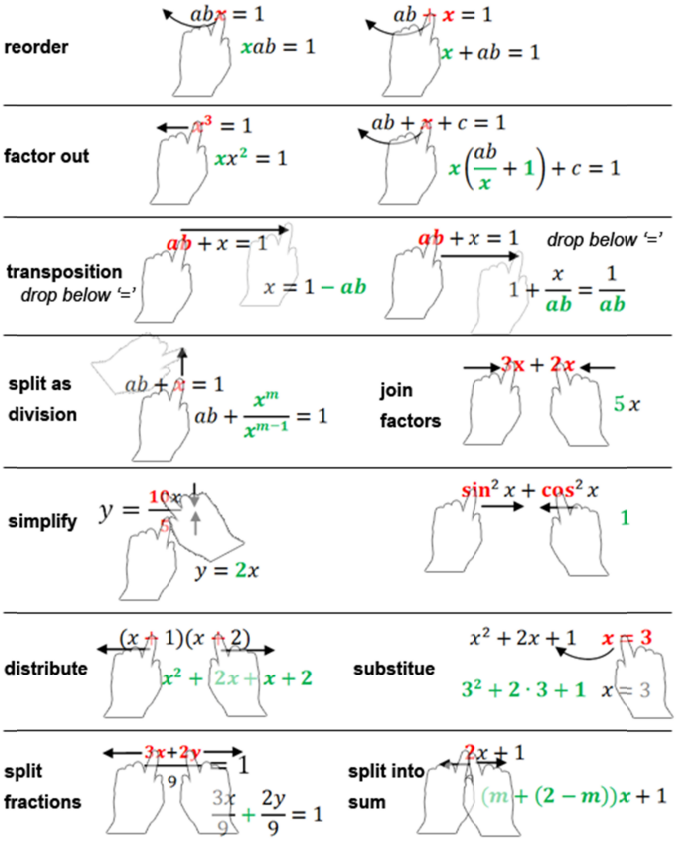

Table 1. Multi-touch algebraic identity transformation UI. Manipulated terms are highlighted in red. Resulting expressions are shown below with modified terms in green.

Although users may enjoy learning the math transformation UI by playfully exploring "*what happens if*" scenarios, a technique such as GestureBar [*5*] could provide additional, more efficient, disclosure of functionality and strategies.

Visual Feedback

Since the structure of an expression may change significantly as the result of a single transformation, we did not feel that it was possible to show the results of a transformation in-place with the original expression. In addition, since a transformation may produce a result that is structurally quite different, we felt that it was important to leave a visual trail that would explicitly show what happened in a transformation step. Thus, as users manipulate a term, the resulting transformation is displayed directly below. We use an arrow-like visualization in combination with colorization and shading cues to depict how terms have changed from one transformation step to the next (Figure 1).

PILOT EVALUATION

To gain insight into the utility of Hands-On Math for domain users and to gain feedback regarding its techniques and features, we conducted a pilot evaluation.

We recruited 9 participants (aged 18-24; mean age: 20.7; SD: 1.79; 2 female; 8 right-handed) from the undergraduate student population of Brown University. Participants were required to use mathematics to support their coursework (e.g. a physics class). We believe undergraduate students are representative of the potential user population because they use mathematics to support their coursework and studies on a regular basis. Participants were compensated.

Participants first completed a pre-questionnaire. Participants were then read an introductory statement that introduced them to the study, and asked to think aloud during the tasks they would be working on. Due to the dependence of our system on a large and expensive Surface, and other hardware limitations (*e.g.*, the infrared light pen is sometimes mistaken for touch input and produces ink at a low sampling rate making some gestures unreliable), we chose to do a lab study. We instructed participants to ignore minor bugs/glitches where possible, as our goal was to evaluate the system design rather than the implementation.

Users were asked to complete a set of exploratory tasks:

- Creating and manipulating pages
- Performing a back-of-the-envelope calculation
- Solving a more complex math expression by performing a multi-step derivation
- Graphing an equation and manipulating the graph
- Using the PalmPrint to change modes and drawing an annotated diagram in different colors of ink
- Web clipping
- Manipulating the contents of a page with TAP gestures and page folding

Participants were permitted to perform additional actions to "play" with the system or to describe a thought they wanted to express. Since the system is not yet designed for self-disclosure, they were assisted with usage when needed.

At the completion of the study (which lasted approximately 45 minutes), participants completed a post-questionnaire.

OBSERVATIONS

Participants were overall very positive about the potential of the system although there was a strong sentiment that the system might be more widely used if it were available in a portable form factor, such as a Tablet PC, or at least on a more ergonomic drafting table display that would make it easier to reach across the screen. In this section, we report specific details of their interactions with system features.

Pages

Participants appeared to experience little difficulty manipulating and writing on pages, as several participants picked up the stylus and began adjusting the paper and writing without instruction. When asked how the virtual pages compared to normal paper, all participants reported that it felt natural despite the awkward light pen used for the experiment. One user was unsure of whether he could place his hand on the page while writing, but all others seemed to write naturally. In addition, users were enthusiastic about the panning bar and played with it longer than needed to perform the requested tasks. Several participants spontaneously remarked that it was "cool" and that it would help them organize their pages into distinct regions.

However, we also observed that participants experienced initial difficulty with the page creation and deletion interface. Not surprisingly, no one discovered the bezel UI without instruction and some participants needed several attempts before remembering that two fingers were required to create a new page. Also, instead of performing the delete gesture as a single fluid motion, most users did it in two steps by first dragging the page partly off-screen and then lifting and tapping on the trashcan icon that appeared.

Paper folding, which was only enabled at the end of the experiment, received mixed reviews. Some users thought it would help them make space on a page, while others did not feel that they needed such a feature. Several users commented that they did not feel that pinching was precise enough for them to accurately select the region that they wanted to collapse and suggested that a TAP gesture would allow them to draw precise boundary lines that could be confirmed as a fold with a pinch gesture.

Gestures

Since no mechanism was provided for gesture discovery, all participants required explanation for how to perform gestures. Verbal descriptions often were misinterpreted whereas a single hands-on demonstration was sufficient.

TAP gestures received mixed reviews from study participants. Several participants noted that using two hands to perform an action seemed unnatural and thought that a unimanual technique would be preferable. In the case of the two selection gestures, participants generally noted that simple lassoing seemed more natural and would be used predominantly, although two thought that rectangle select would be more efficient. Participants also agreed that the rectangle selection TAP gesture would be best for making precise selections, such as when cropping an image. The fact that all selection gestures were simultaneously available was considered a plus since there was no perceived cost to having the extra gestures despite the fact that TAP gestures generate feed-forward. However, in the case of the space insertion gesture, participants were enthusiastic and eagerly explored different ways to manipulate their ink.

The PalmPrint technique was particularly sensitive to hand posture in order for contacts to be generated for the palm and each finger using the Microsoft Surface SDK. Several users placed their hand flat on the surface and needed prompting to arch their hand because the SDK would not generate thumb contacts in particular. No participant reported any discomfort when asked to successively lift and tap each of their fingers, although two noted that lifting the ring finger was difficult. When asked how they felt about having their hand on the surface when drawing, only one participant noted a preference to keeping their hand off screen. All participants were comfortable switching between red, green and blue ink without looking, although they all thought that being able to customize the function mapping for each finger was a useful feature. One participant suggested that it might be more convenient to have different function "palettes" instead of customizing each finger one at a time. Several participants noted that the PalmPrint would work well when switching modes frequently, for instance to change colors in a drawing, but that the feature of having it collapse into a tool palette when the palm was lifted would be more useful when making occasional mode switches. No participant indicated a preference for dragging the tool palette as a toolglass. Several thought that it was nice that they could "re-snap" the tool palette by simply placing their palm anywhere on the surface.

When contrasted with using two fingers to pan graph contents, finger posture was generally considered more difficult to remember and perform. Users seemed to exaggerate the instruction to use their fingertip by rotating their hand into unnecessarily awkward positions. The feedback widget showing the recognized posture was well received as users wanted to find the transition point between finger tip and finger pad. After a brief introductory period, participants seemed to control their posture without difficulty which indicates it may be useful as an additional mode.

The under the rock menu initially appeared to induce unnatural hand positions. Some participants dragged terms across their body and thus had to cross their hands to reach the menu; others tried to find ways to access the menu with the same hand they used to drag a term. In both cases, additional instruction about how to efficiently use the menu with one or two hands appeared to resolve the difficulty. Once learned, however, all participants found the technique to be convenient with only one suggesting that it could be improved by allowing the dragged term to be released as soon as the menu was activated (our implementation treated the menu as a modifier of the initial finger contact and disappeared when the initial contact was released). Users overall did not find the semi-transparent menu to be distracting, although one noted that it took "getting used to."

Math

Participants reported that the ability to manipulate mathematical expressions with simple manipulations would be very useful to them when doing math-related work – they liked the idea of working step by step and not just being given an answer. Participants were unanimous in noting that being able to manipulate math would both save them time and would help them avoid being confused by transcription errors. Several commented that manipulating math would help them to explore possible transformations when they were unsure of the next step. All participants found that being able to seamlessly write mathematics and then create and explore its graphical representation was very powerful. However, some participants felt that multi-touch dragging and zooming were insufficient and that we needed mechanisms for "resetting" the graph domain and range and for automatically choosing "good" bounds. We were surprised that despite the very poor quality of the light pen that we provided, participants could write math that was generally correctly recognized.

In terms of functionality, some participants wanted higher-level operations than simple algebraic transformations, with one requesting us to support all of Mathematica. Since StarPad already can use Mathematica for computation, adding additional math functionality is quite feasible. All participants thought that the ability to simplify equations and compute values would be very useful, although some worried that notations written on one page might affect computations on another and were satisfied when informed that expressions were scoped to their page.

DISCUSSION AND FUTURE DIRECTIONS

Our central hypothesis, that people would learn and work more efficiently if CAS functionality were available in a paper-like environment, was not contradicted by our evaluation. There seemed to be unanimous agreement that mathematical problem solving is most naturally driven by an unconstrained handwriting-based UI, but that paper and pencil, despite being the tool of choice, suffers from requiring tedious and error prone transcription and from failing to provide basic computational assistance needed to avoid making "stupid mistakes." The ability of our system to support free-form note taking, symbolic and numerical computation, graphing, and function transformation all "without" a UI led participants to conclude that the system has "great potential." The most significant perceived obstacle to adoption was the bulky, non portable form factor of our Microsoft Surface hardware and the low quality of light pen input, as compared to physical pens or even Tablet PC ink. It was also clear that extending the set of possible mathematical operations should be a high priority. We also feel important recognition techniques still need to be addressed, for example, to interpret math written on angled baselines, to automatically distinguish mathematics notations from diagrams and free-form inking, and to recognize, anchor and track annotations of typeset terms and symbols.

Page metaphor. We also found that the choice of using manipulable pages as a primary UI element, as opposed to a whiteboard or book metaphor, appeared to provide a viable alternative to explicit grouping as an organizing principle. Users seemed to have a strong, *a priori* sense of how to organize information with pages. They expected math written on one page to be in the same computational scope, and distinct from math written on other pages. They had strong feelings about wanting to grow pages to add related information and to use a new page to enter logically different information. Being able to fold pages to make more space seemed natural and "cool" to most participants; however, many felt that a pen-based technique was needed to precisely define the pinch boundaries while also admitting they might not need folding functionality very often. Alternatively, users were captivated by the panning bar, identifying it as a convenient tableau for collecting informal collections of pages and addressing the desire to spread a working set of pages out beyond the limited dimensions of the display surface. Pages also provided a natural work unit in which users could explore a problem, then discard the page if they were off-track, or push it aside to use a new page when handling an interruption. We expect that pushing the page metaphor more, for example, to flip, curl, staple, hyperlink, or embed pages may reap benefits.

Sandwich Problem. With regard to specific UI choices, we were somewhat surprised to find that users were not inclined to be receptive to bi-manual interaction. We summarize their reticence as the *sandwich problem* in which participants felt that it was unnatural to require bi-manual interaction since their other hand might be doing something else, like holding a sandwich. We interpret this to mean that users are not only concerned with *actually* using their other hand to do something else but they also were concerned that they *might* want to do something with their other hand besides improve a manipulation they could do almost as well with one hand. In essence, if the effort expended on bi-manual interaction appears to greatly exceeds any performance benefit gained, then uni-manual interaction may be preferred. It is possible bimanual gestures take "getting used to" and so it may be appropriate to always have uni-manual alternatives to ease the learning curve and to address the sandwich problem.

Recognition. Counterbalancing the sandwich principle somewhat, we observed that the most likely gestures to be misrecognized were those that required either multi-touch or pen input, but not both. Hybrid TAP gestures were not accidentally triggered during the evaluation. However, largely due to the poor quality of the pen used, there were occasions when a TAP gesture stem was not recognized causing ink to be left on the display. Thus we expect that it may be important to increase recognition latitude of the pen part of a TAP gesture, perhaps in response to sensing a hand hovering over the display, and/or to develop efficient recovery techniques when the pen stem is not recognized.

Physical skill. We also found it notable that different users employed different, often inefficient, physical strategies when performing gestures. When shown a more efficient technique, they were almost instantly able to improve their performance, in many cases having an "Aha" moment. For example, switching between finger tip and finger pad touching requires only the bending of the second joint of the index finger; however many users adopted awkward poses such as fully extending the index finger and rotating their arm to be perpendicular to the surface. Similarly, when switching from writing ink with a stylus to dragging terms with their finger, several users tried to find a place to put the stylus on the table instead of tucking it up in their palm. Several users noted that they would like to use under the rock menus with one hand, but did not figure out on their own that this could often be accomplished more easily with the index and forefinger instead of the thumb and index finger. Thus, we expect that pen and multi-touch techniques may require more sophisticated and in-depth disclosure mechanisms than pen only gestures, for instance.

Disclosure and entrenchment. Even though multi-touch input is relatively new, it seemed clear that some techniques have already become entrenched and others were harder to discover and master. For example, several users had trouble considering finger posture as an option for panning a graph because they felt that two-finger dragging was the *de facto* scrolling standard based on their experiences with MacBooks. They also considered their experiences with iPhones and MacBooks where all touches are equal. Thus extending disclosure techniques like GestureBar for surface interaction is worthy area for future research.

CONCLUSION

We presented a prototype system, Hands-OnMath, which reduces the barriers to accessing computational assistance during math problem solving by unifying CAS functionality with a virtual paper UI. This system contributes novel bimanual and gestural techniques for managing and writing on virtual note pages in addition to direct manipulation techniques for algebraically transforming mathematical expressions. Pilot studies indicate that, after refinement, a mature version of Hands-On Math would be a desirable tool for scientific and academic note-taking and ideation.

ACKNOWLEDGMENTS

Thanks to Andries van Dam for his guidance and to Michael Haller, Christian Rendl, Jakob Leitner, Florian Perteneder, Alexandra Feldman and Nicholas Sinnott-Armstrong for stimulating discussions and code support. Thanks to Microsoft, Inc., for their sponsorship and to Ken Hinckley, Daniel Wigdor and Joseph LaViola, Jr. for technical support. This material is also based upon work supported under a National Science Foundation Graduate Research Fellowship and in part by NSF grant IIS-0812382.

REFERENCES

1 Anthony, L, Yang, J, and Koedinger, K. Adapting Handwriting Recognition for Applications in Algebra Learning. *International Workshop on Educational Multimedia and Multimedia Education* (2007).

2 Benko, H, Wilson, A, and Baudisch, P. Precise Selection Techniques for Multi-Touch Screens. *UIST* (2008), 77-86.

3 Bier, E, Stone, M, Pier, K, Buxton, W, and DeRose, T. Toolglass and magic lenses: The see-through interface. *Siggraph* (1993), 73-80.

4 Bragdon, A, Zeleznik, R, Reiss, S et al. Code Bubbles: A Working Set-based Interface for Code Understanding and Maintenance. *CHI* (2010).

5 Bragdon, A, Zeleznik, R, Williamson, B, Miller, T, and LaViola, Jr., J. GestureBar: improving the approachability of gesture-based interfaces. *CHI* (2009), 2269-2278.

6 Brandl, P, Forlines, C, Wigdor, D, Haller, M, and Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. *AVI* (2008), 154-161.

7 Bunt, A., Terry, M., and Lank, E. Friend or foe?: examining CAS use in mathematics research. In *Proceedings of CHI'09* (), 229-238.

8 Davidson, P and Han, J. Extending 2D Object Arrangement with Pressure-Sensitive Layering Cues. *UIST* (2008), 87-90.

9 Elmqvist, N, Henry, N, Riche, Y, and Fekete, J. Melange: space folding for multi-focus interaction. *CHI* (2008), 1333-1342.

10 Frisch,M, Heydekorn,J, and Dachselt,R. Investigating Multi-Touch and Pen Gestures for Diagram Editing on Interactive Surfaces. *ITS* (2009).

11 Guiard, Y. Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model. *The Journal of Motor Behavior*, 19, 4 (1987), 486-517.

12 Guimbretriere, F, Martin, A, and Winograd, T. Benefits of Merging Command Selection and Direct Manipulation. *ACM Transactions on Computer-Human Interaction* (2005), 460-476.

13 Hinckley, K., Pahud, M., and Buxton, B. Direct Display Interaction via Simultaneous Pen + Multi-touch Input. In *Proceedings of SID'09* ().

14 Hinckley, K, Pahud, M, and Buxton, B. Direct Display Interaction via Simultaneous Pen + Multi-touch Input. *Proceedings of Society for Information Display* (2010).

15 Hinckley, K, Zhao, S, Sarin, R, Baudisch, P, Cutrell,E, Shilman,M, and Tan, D. InkSeine: In Situ Search for Active Note Taking. *CHI* (2007).

16 http://pen.cs.brown.edu/starpad.html.

17 Kurtenbach, G and Buxton, W. The limits of expert performance using hierarchic marking menus. *CHI* (1993), 482-487.

18 Labahn, G, Lank, E, MacLean, S, Marzouk, M, and Tausky, D. MathBrush: A System for Doing Math on Pen-Based Devices. *IAPR International Workshop on Document Analysis Systems* (2008), 599-606.

19 LaViola, Jr., J and Zeleznik, R. MathPad2: A System for the Creation and Exploration of Mathematical Sketches. *SIGGRAPH* (2004).

20 Li, Y, Hinckley, K, Guan, Z, and Landay, J. Experimental analysis of mode switching techniques in pen-based user interfaces. *CHI* (2005).

21 Malik, S, Ranjan, A, and Balakrishnan, R. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. *UIST* (2005), 43-52.

22 Matejka, J, Grossman, T, Lo, J, and Fitzmaurice, G. The Design and Evaluation of Multi-finger Mouse Emulation Techniques. *CHI* (2009).

23 Morris, M, Lombardo, J, and Wigdor, D. WeSearch: Supporting Collaborative Search and Sensemaking on a Tabletop Display. *CSCW* (2010), 401-410.

24 Moscovich, T. Contact Area Interaction with Sliding Widgets. *UIST* (2009), 13-22.

25 Odell, D, Davis, R, Smith, A, and Wright, P. Toolglasses, Marking Menus, and Hotkeys: A Comparison of One and Two-Handed Command Selection Techniques. *Graphics Interface* (2004).

26 Roth, V and Turner, T. Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. *CHI* (2009).

27 Terrenghi, L. Affordances for manipulation of physical versus digital media on interactive surfaces. *CHI* (2007).

28 Thimbleby, W. A novel pen-based calculator and its evaluation. *NordiCHI* (2004), 445-448.

29 Wang, F, Cao, X, Ren,X, and Irani,P. Detecting and Leveraging Finger Orientation for Interaction with Direct-Touch Surfaces. *UIST* (2009).

30 Zeleznik, R and Miller, T. Fluid inking: augmenting the medium of free-form inking with gestures. *Graphics Interface* (2006), 155-162.

31 Zeleznik, R, Miller, T, Li, C, and Laviola, Jr., J. MathPaper: Mathematical Sketching with Fluid Support for Interactive Computation. *Smart Graphics* (2008), 20-32.