William I. Grosky
Utz Westermann

Applications and Issues in Pen-Centric Computing

Robert Zeleznik, Timothy Miller, Andries van Dam, Chuanjun Li, Dana Tenneson, and Christopher Maloney
Brown University

Joseph J. LaViola, Jr.
University of Central Florida

In the early 1960s, Ivan Sutherland's Sketchpad implemented a truly revolutionary improvement on batch processing of alphanumeric codes punched on cards: rich, graphics-based, real-time human–computer interaction.[1] However, this idea hit an early plateau with the development of windows, icon, menu, pointing device (WIMP) GUIs, which have slowly evolved over the past three decades but still bear the hallmark of their origins—Engelbart's mouse-and-keyboard interface[2] and the first bit-map, graphics-based WIMP GUI developed at Xerox PARC for the Alto.[3] We believe that it's natural for users to express themselves both at a higher-bandwidth and even more fluidly than is possible with WIMP GUIs. Therefore, our research group's interest lies in more dramatic alterations of the user interface, including haptics and immersive VR, and more recently on pen input, defined broadly to include the use of not only pens, styli, markers, photo pens, and so on, but also touch and multitouch devices.

Where are the opportunities to leverage pen input? An obvious place is wherever rapid, free-form 2D input is needed, such as for sketching, conceptual design, and other forms of artistic expression. Another is for natural input of 2D and 3D elements of traditional, more structured visual languages where pen input promises a fluidity and ease of learning and remembering, because it stays within the original 2D visual conventions. In contrast, WIMP GUIs for domains such as music, mathematics, chemical formulas, logic and circuit diagrams, and 3D geometry require either menu picking of primitives or laborious encoding of intrinsically 2D or 3D information in linear, keyboarded languages. For example, entering even the expression

$$\int_{-\infty}^{\infty} e^{-x^2} dx$$

into Wolfram Mathematica would require either learning a different and less obvious linear notation, such as ''integrate[E^−x^2, {x, −infinity, infinity}]'' or using an awkward palette-based interface. Another way pen-computing distinguishes itself is through its adaptability to a broad spectrum of environments ranging from cell phones and PDAs to ultra-mobile PCs, tablet PCs, electronic whiteboards, and surfaces based on projection and computer-vision-based multitouch input. For example, small tablet PCs have even become commonplace in areas such as healthcare for directly recording patient data into electronic health records rather than via paper and subsequent transcription.

Why are we optimistic that pen-computing is viable, given past failures (see the ''Previous Work'' sidebar for examples of earlier efforts)? Not only is today's tablet PC hardware orders of magnitude more powerful than the early pen computers, but also there have been many significant advances in recognition technology—the enabling technology of pen computing. Today's applications can build on a set of vendor-provided ink-analysis utilities, and cursive handwriting and continuous speech recognizers are finally good enough, after more than four decades of R&D, for everyday

Editors' Note

As part of the rapidly evolving field of designing more natural user interfaces for multimedia information, pen-centric computing refuses to disappear. As a quite natural and universal interface modality, it presents many challenges. In this article, the pen-centric computing group at Brown University, led by Andries Van Dam, surveys the many prototypes they have designed and implemented, and discuss the research issues in the field still to be explored.

—William I. Grosky and Utz Westermann

1070-986X/08/$25.00 © 2008 IEEE Published by the IEEE Computer Society

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.

use with acceptable error rates. Many recognition techniques now can rely on modern commodity hardware; these techniques formerly required supercomputers for processor- and storage-intensive algorithms. For example, machine learning over large data sets can be used for real-time recognition, and machine vision algorithms can be used for multitouch and gestural input.

It's an exciting time to be involved in pen-computing and, in particular, in R&D in novel interaction paradigms and techniques. In the rest of this article, we illustrate some of the opportunities with particular proof-of-concept projects done at the Microsoft Center for Research in Pen-Centric Computing at Brown University, concentrating on interaction rather than recognition techniques, although the two are, of course, strongly related.

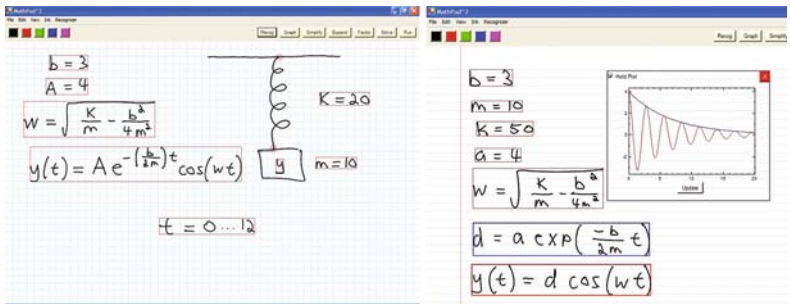
Current Application Areas

Because many application areas can potentially benefit from pen computing, we have made the conscious decision to focus on three areas that collectively represent a broadly representative spectrum of user interface styles. We hope that these projects not only stimulate other application development but also provide foundational components for building richer more unified applications.

Math beyond recognition

Much of the emphasis in using handwritten math with computers has focused on the problem of simply recognizing the math. Our research has taken a complementary tack, asking what we can do using the recognition available today. Our work has fallen into three broad areas: compensating for the imperfections of current recognizers, developing fluid pen techniques for tasks previously requiring a keyboard, and investigating capabilities enabled by handwritten math not possible with typed math.

Historically, we investigated the last area first. Our application MathPad^2 is designed to aid mathematical problem-solving with the concept of mathematical sketching, making dynamic illustrations by combining handwritten mathematics and free-form illustrative diagrams of shapes that change size, location, and other properties as a function of the mathematics.[4] MathPad^2 lets users create simple illustrations as if they were working with pencil and paper. In addition, users can leverage their physical intuition by watching their hand-drawn diagrams animate in response to continuous or discrete parameter changes in their written formulas (see Figure 1). Teachers could use MathPad^2 to quickly create illustrations for use in their lessons; and students could use the application to aid in their studies.



(a) (b)

Figure 1. MathPad^2: (a) Screenshot of a mathematical sketch for a damped harmonic oscillator. The handwritten math is associated with the drawing using a gestural user interface. The diagram will animate according to this math specification. Users can change system parameters to see how the oscillator is affected under various conditions. (b) Screenshot showing MathPad^2's graphing capability, where users can write down mathematical expressions and graph them using a hook gesture.

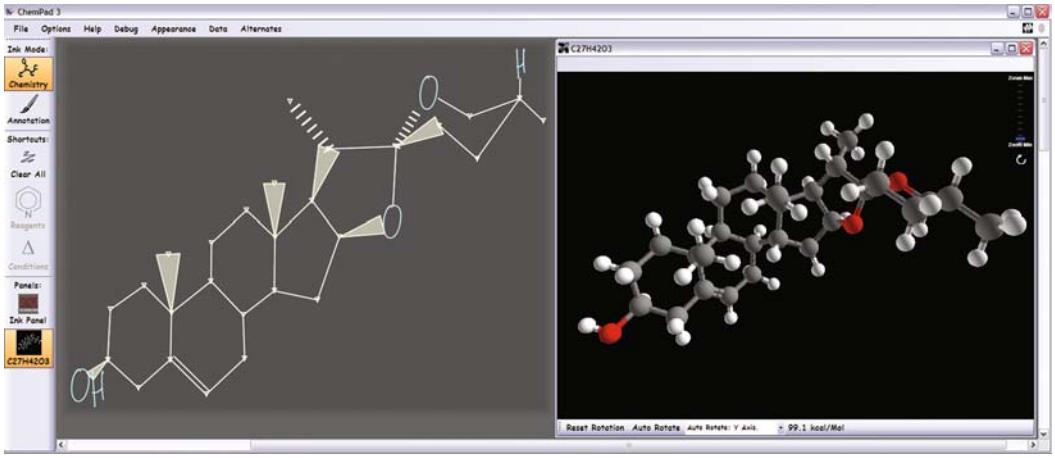
For our MathPaper project, the successor to MathPad^2, our goal is to make working with math on the computer as easy as writing it on paper.[5] MathPaper provides a virtual sheet of paper that recognizes free-form handwritten entry of multiple mathematical expressions in real time. It also provides symbolic and numeric computational assistance and exports to Microsoft Word, LaTeX, Mathematica, and so on, supporting functionality previously only available with typed-in math.

To allow productive use with today's recognition technology, however, part of this project's research involved investigating techniques for visualizing and allowing the user to correct errors in the computer's recognition with gestural and widget-based interactive editing, including real-time display of the recognition result during interaction. We have explored extensions to existing notations to control evaluation, allow entry of matrices with elided elements, incorporate flow of control and other algorithmic concepts, support debugging, and so on.[6] Our ultimate goal is for MathPaper to provide students, teachers, and professionals with an integrated environment that combines standard, symbolic math with geometry, algorithms, and symbolic computational assistance, and that supports devices ranging from PDAs to whiteboards.

Chemistry

ChemPad is a pen-centric application that has a pedagogical focus and generates 3D

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.



(a) (b)

Figure 2. ChemPad interpreting the steroid diosgenin: (a) a rectified hand-sketched 2D diagram has been converted into (b) an interactive, 3D molecule model. In the 3D molecule model, the oxygen atoms are shown in red, the carbon atoms in grey, and the hydrogen atoms in white.

molecular structures from hand-drawn digital ink, using a simulation of the molecular mechanics that determines most likely conformations of the sketched molecule.[7] Molecules are inherently 3D objects represented by chemists on paper and classroom blackboards by a system of 2D diagram notations. ChemPad allows student chemists to sketch molecule diagrams in a quick, natural fashion and then have the software automatically generate the corresponding 3D models (see Figure 2).

For students who have difficulty thinking in terms of 3D chemistry—a critical skill for would-be chemists—ChemPad's automatic construction of 3D models helps them learn to visualize molecules as the 3D objects they are. Hundreds of students have used ChemPad in Brown's introductory organic chemistry course since 2004, and is available as a free download at the project website (see http://graphics.cs.brown.edu/research/chempad/).

Also worth noting is Reactor, which is a ChemPad-related project that allows students to direct chemical reactions with guidance from the system during the process. One of the most complex issues for beginning organic chemistry students is predicting the outcome of a reaction process among two or more molecules. In this system, students draw molecules, select them, and click the react button. The application presents a series of possible pathways; the student directs the reaction, making choices with the help of hints. The system shows the reaction's final products and a mechanism that describes how they form (see Figure 3). This process can be repeated as many times as desired, a potentially useful aid for chemical synthesis problems.

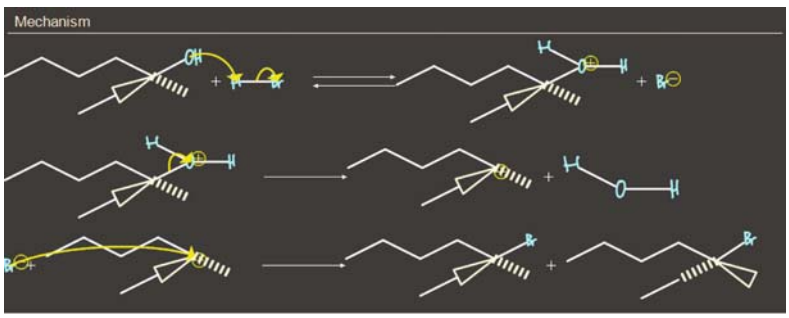


Figure 3. Reactor showing the steps of a substitution reaction between a tertiary alcohol and hydrogen bromide. First, the alcohol acts as a base, taking a hydrogen atom from the acid. Next, water dissociates from the hydrocarbon chain. Finally, negatively charged bromine, a nucleophile, attaches to the positively charged carbocation to form a racemic mixture.

Diagramming

Given the complexity of creating precise models from completed hand-drawn sketches—a problem traceable to foundational problems of computer vision and cognitive science—we instead pioneered an alternate recognition approach that converts input strokes to a beautified form as they are entered. Our work began more than a decade ago with the pioneering Sketch system, which recognized a set of gestural commands corresponding to primitive drawing elements such as cubes, ovoids, and extrusions.[8] Because these gestures were geometrically similar to features of the primitives they represented, users could draw 3D shapes without the complexity or recognition artifacts commonly associated with deferred recognition. In essence, the interpretation process had become an interactive dialogue in which users could identify errors as they occurred and seamlessly fix them using drawing or other interactive manipulation.

More recently, in hope of gaining momentum in more mainstream venues, we devel-

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.

oped Lineogrammer, which focuses on the arguably simpler problem of creating 2D diagrams.[9] With Lineogrammer, users create 2D diagrams, including text labels, by drawing them with a stylus. Each input stroke is classified either as a line that is snapped to other drawing elements, as a shape, as text, or as a gestural command (see Figure 4). For example, a user might draw a rectangle one line at a time or in one stroke—in either case, after completion the system would hold a dual interpretation of four lines and as one rectangle. The user could then move or scale the diagram as a rectangle, drag an individual vertex or edge, write a text label, or add additional lines and scratch out unwanted line segments to create an irregular shape.

Lineogrammer relies on gestures, rather than modes, meaning that users don't have to switch between various drawing tools, as in popular drawing programs such as Microsoft PowerPoint or Adobe Illustrator. Instead, users perform the appropriate gesture modelessly. For example, rather than switching to the selection tool to select items, a user can perform a lasso-like gesture around the items they would like to select; or to enter text, a user can write text in place and rely on the system to automatically disambiguate it from shape geometry. Lineogrammer enriches drawing as an interactive dialogue by reinventing conventional straight edges as virtual rulers, which not only can push shapes, but also can align vertices, serve as an axis of symmetry for mirroring operations, and evenly distribute drawing features along a line.

As part of Lineogrammer, we developed a GestureBar to address the obvious concern of how to teach new users the concept and nuances of gestural interaction. Unlike WIMP interaction's discrete, moded input, GestureBar gestures have a modeless, analog nature in which their geometric definitions overlap with the geometries of diagram elements and text. New users thus need to become familiar with the concept that their hand-drawn input will be classified as a text, drawing, or command action, and that, to a degree, they need to develop an understanding of the form of the gestures along with physical skill so that their input can be correctly interpreted. Thus, Lineogrammer's GestureBar looks like a familiar toolbar with iconic buttons representing different system functions, such as undo, zoom, or create shape. However, toolbar items in the GestureBar don't perform commands, but rather show annotated animations with context and provide a practice area for learning how to perform commands (see Figure 5).

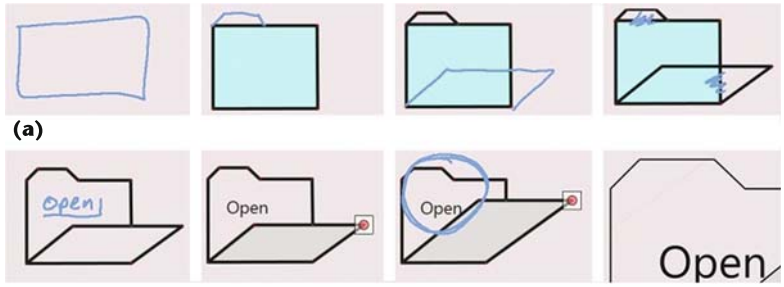


Figure 4. A drawing sequence in Lineogrammer. (a) This row shows drawn lines beautified and snapped. Unwanted segments are scribbled out. (b) In this row, text is recognized and typeset, a vertex is moved, and a zoom gesture is made.

Research issues

Our experience designing pen-centric applications gives us a perspective on research problems that is strongly influenced by user-interface issues. Certainly, research in recognition techniques will continue to be an open field of research, but we believe that current recognition technology is good enough to incorporate as a tool in highly interactive systems. We believe some of the most interesting research problems come from the intersection of these two paths, such as when user interface techniques require novel recognition approaches that emphasize the timeliness of a recognition result over its accuracy. The research questions we tend to focus on are: what to recognize, what to show, how to recognize, and how to measure research progress.

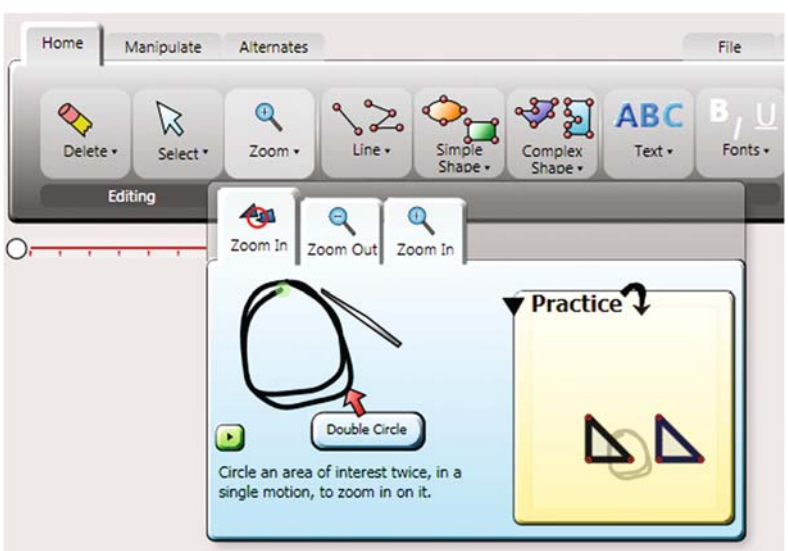


Figure 5. GestureBar from our diagramming project. The user chooses Zoom from a familiar-looking, toolbar-like interface to see related animated demonstrations of gestures along with a practice area.

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.

What to recognize

Even in cases where established 2D languages already exist, such as math or chemistry, the languages need to be extended to control computation and workflow in various ways. In the case of math, we developed persistent notational extensions, with one example being an arrow notation that results in the display of a computational result that is updated whenever the math changes. A more general concern is to design a standard gestural style that would work in multiple domains to control editing activities, such as deletion and moving.

Designing these gestures so that they can be interpreted from 2D languages is a simple example of the deeper ambiguity issue that arises when notations from one or more languages are interspersed on the same surface. For example, a math recognizer and a diagram recognizer would not, in general, support geometric diagrams with algebraic labels. Creating monolithic recognizers for commonly overlapped domains is a tractable but labor-intensive alternative to the challenge of making a general framework that can interpret multiple types of languages.

A potentially important issue with language recognition is to control to what extent input is interpreted literally. For instance, it might not make sense to recognize geometric dimensions in quickly sketched diagrams in the same way as in carefully drawn pictures. Such geometric interpretation is further complicated because users might intermix sketched input with more precise input. A common technique in diagramming is to express precision through explicit language notations, but doing so can be time consuming, and can lead to further recognition problems. To approach the flexibility of pencil and paper, users might need to define these and other notations to suit their specific needs.

What to show

Although one commonly expressed goal for pen-centric interaction is that it be as transparent as writing on paper, feedback is, in fact, critically important. This feedback comes in two forms: displays that disclose what can be done, and displays that show what was interpreted and computed.

Interface disclosure. A longstanding obstacle to the development of pen-based and gestural systems is that they are not as inherently self-disclosing as the more popular WIMP user interfaces, which can be visually explored as needed. Several research efforts have attempted to address this obstacle with approaches ranging from interactive tutorials and reference crib sheets to interactive references embedded in conventional user interfaces. The detraction of traditional tutorials and crib sheets is that they don't seem to mesh with workflow patterns.

Thus, the challenge is to develop techniques that can be incorporated into conventional user interfaces. However, because pen interactions intrinsically require some degree of skill, simply disclosing their existence isn't sufficient. Instead, techniques should make it easier to learn gestures and to motivate users to develop that skill. A good example of this is the design of marking menus, which can allow users to learn a limited class of abstract gestures as a byproduct of interacting with a familiar, visual user interface.[10]

Visual feedback. Recognition in pen-based computers—whether text, math, or diagrams—is never going to be 100 percent accurate due to handwriting variability and the probabilistic nature of recognition algorithms. Thus, visual feedback is an integral part of the recognition process because it lets users see how the recognition system is interpreting the digital ink. An important research question is how to render the form, content, and timing of this feedback.

The location of the immediate visual feedback can play an important role in how a user interprets and determines if a recognition error has been made. Possible options include replacing user input with a typeset display, augmenting with a typeset display, or symbolically annotating. For example, we have experimented with different visual display styles for math-expression recognition (see Figure 6).[11] Our observations indicate that people like interactive feedback as long as it doesn't disrupt them; in the case of math, one preferred option was displaying a small typeset representation below their handwritten ink.[12] In more extensively 2D contexts, such as diagram drawing, providing effective feedback is an open problem.

An additional concern is when feedback should be displayed. People frequently have

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.

two conflicting goals: first, not to be disrupted; second, to have immediate access to a display of the computer's interpretation of their input. One technique is to continuously update a display of the recognition state, which makes it easy to spot recognition errors but can be distracting. At the other extreme, recognition results can be displayed only in response to an explicit user request, which is nondisrupting but can be frustrating. Because the user's task likely will affect their response, it seems unlikely that any one technique will be sufficient. User control over the timing of the feedback's format seems like a useful customization.

How to recognize

Recognizing pen input is a significant research field in its own right. Pen recognition problems include extracting low-level features, such as cusps (that is, sharp corners), from raw input strokes, recognizing symbols and primitive shapes, inferring spatial relationships, and interpreting semantics. Davis provides an overview of many of these problems.[13] We are particularly interested in user-interface issues that impact recognition, and vice versa. For example, because none of the existing math recognizers were either fast enough to support interactive feedback displays for math or provided enough control over errors, we explored an efficient rule-based approach to math instead of the more popular machine-learning approach. Although our recognizer is faster than others, it requires significant manual effort to extend and might require significant work to figure out how to express rules for some concepts. A hybrid, rule-based, machine-learning strategy offers a potential solution to this problem.

Another important interface consideration is how users can adapt the recognizer to their input style. Ideally, this type of learning would occur automatically in the background as users erase and redraw misrecognized characters. However, we suspect that explicit strategies might provide more robust results—the challenge being to make them nondisruptive.

The recognition problem is complicated by hardware and multimodal input issues. Tablet PCs typically support both high accuracy and high sampling rates; however, few people would willingly accept the feeling of writing on even a high-end tablet PC over that of a 10-cent pencil and a sheet of paper. This can be attributed to calibration problems associated with the hardware, fundamental parallax caused by writing on a thin layer of glass, and physical properties associated with the glass surface and stylus tip. These problems are exacerbated by a tablet's heft, which makes it more difficult than paper to hold in a convenient drawing position. Alternative devices, such as electronic whiteboards, typically suffer more because of generally-lower sampling rates and less-desirable surface materials.

The research community has gone beyond basic pen input to consider multimodal issues, such as multitouch and hybrid speech systems. Doing so is becoming increasingly relevant as new tablet PCs will soon support simultaneous pen, finger, and speech input. Research on how to use these capabilities is wide open, ranging from high-level applications to low-level issues, and includes, for example, palm rejection (disambiguating a finger used in conjunction with a pen from a palm placed on the surface to support the hand while drawing with a pen). An interesting problem to consider is how a pen might be used to augment a traditional keyboard and mouse.

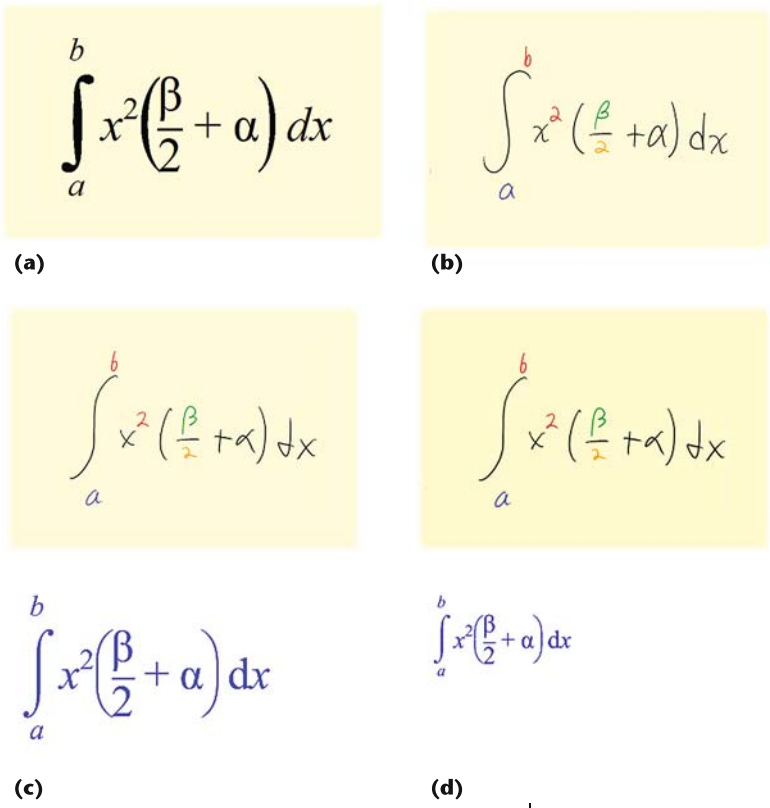


Figure 6. Immediate feedback styles in MathPaper: (a) Typeset in place replaces handwritten ink strokes with typeset mathematics on the fly; (b) adjusted ink replaces handwritten ink strokes with characters from a clarified and colorized ink font; (c) large offset displays typeset recognition results below and scaled to the same width as the handwritten ink; and (d) small offset displays typeset results below the handwritten ink, but at a constant, relatively small font size.

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.

Related Work

Pen-centric research, having started around the time of Sketchpad in the early 1960s, has proven challenging. Ivan Sutherland's brother Bert used the same homebrew Massachusetts Institute of Technology computer, the TX-2, to create the first sketching and recognition program, using a light pen as the sketching and drawing tool.[1] The work of the Sutherlands and other pioneers and their projects, such as Robert Anderson's PhD work in the mid-1960s on a Rand tablet, introduced the still-active research area of interactive diagram and math creation and recognition.[2]

Alan Kay's Dynabook papers of the late 1960s[3] provided a vision equally revolutionary in the days of multimillion-dollar, time-shared mainframes: that of a commodity, portable, pen-driven, simulation-and-animation platform that is exemplified today not only by modern laptops, but also by pen- and touch-driven tablet PCs. However, it took until the late 1980s and early 1990s before commodity hardware was sufficiently powerful to allow commoditization.

Systems from this period included Wang's Freestyle, which integrated digital ink and voice annotations with simple word processing;[4] Microsoft's Windows for Pen Computing, released in response to Go's PenPoint,[5] a hardware–software system oriented toward pen-computing; and Apple's handheld Newton, which was capable of recognizing handwritten input. However, only the 1990s technology of Palm Pilot's Graffiti (a streamlined alphabet designed for stylus input on PDAs) and the Anoto photo pen (designed to facilitate information entry from hard copy to computer) have survived.

References

1. W.R. Sutherland, *The On-Line Graphical Specification of Computer Procedures*, doctoral dissertation, Massachusetts Inst. of Technology, 1966.
2. R.H. Anderson, *Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics*, doctoral dissertation, Dept. Applied Mathematics, Harvard Univ., 1968.
3. A.C. Kay, ''Dynabooks: Past, Present, and Future,'' *Library Quarterly*, vol. 70, no. 3, 2000, pp. 385-395.
4. S.R. Levine and S.F. Ehrlich, ''The Freestyle System: A Design Perspective,'' *Human–Computer Interaction: Toward the Year 2000*, R.M. Baecker et al., eds., Morgan Kaufmann Publishers, 1995, pp. 871-880.
5. Go Corporation, *Penpoint Application Writing Guide*, Addison Wesley, 1993.

How to measure research progress

Pen input recognition is notoriously sensitive to individual variations in writing style. Therefore, any results need to be widely tested, an impractical requirement for many organizations. Developing appropriate benchmark datasets could partially address this issue. Additionally, researchers might want to test interface techniques independently of a specific underlying recognizer or technology, because techniques are so intimately connected to user performance.

Developing benchmark datasets. Unlike more traditional user interfaces, two individuals with similar backgrounds can have dramatically different experiences with the same pen-based application because of physical differences in writing style. Therefore, the it-works-for-me design philosophy is completely untenable. However, much of the recognition can be tested without involving the user interface; doing so allows for the possibility of collecting benchmark datasets. Given the effort involved in creating these datasets, it's important that researchers carefully design the data collection procedure. For instance, a math database could be different if people were asked to copy equations as opposed to solve problems, and in either case would likely be different if the input occurred on a tablet PC, electronic whiteboard, or pencil and paper.

Testing and evaluation. The goal of usability evaluation is to gain some fundamental understanding of how people would use an interface—but in the case of recognition technology, there is no ideal technology against which to evaluate test results. The error rates of existing recognizers can be expected to be different, and presumably higher than they will be in the future. It's possible to control the error rate to attempt to gain a fundamental understanding of how people use and react to interfaces independently of the underlying recognition technology, but this is difficult to achieve in an ecologically valid way, as the recognizers will react differently to different people. For example, if input is given that all expected recognizers would interpret input correctly, then it might not be valid to artificially generate a recognition error. These issues are compounded by the fact that recognition technology depends on user skill level, which makes larger user samples as important as longitudinal evaluations.

Conclusions

We believe that pen-computing has an exciting role to play in augmenting or even replacing the canonical WIMP GUI style of interaction with a more fluid, natural, and

Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.

efficient style of interaction that decreases cognitive distance between task and expression of that task, and takes us a step closer to the ideal of user-interface transparency. Our optimism is based on nearly a decade of experimentation and the continuous improvement in hardware and software technology that makes new approaches feasible even on small platforms. With a rich R&D agenda focused not only on fundamental recognition algorithms but also on fluid interaction styles that minimize mode switching and cognitive loading, projects in pen computing are showing encouraging results. **MM**

Acknowledgments

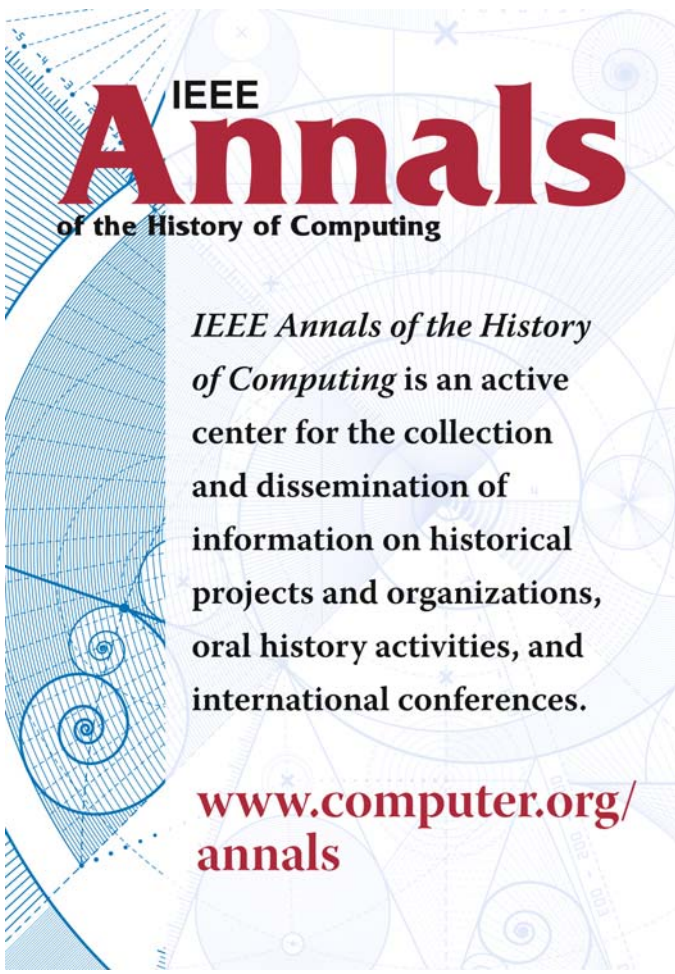
We thank Andrew Forsberg and Andrew Bragdon for critical readings of drafts of this article. Microsoft and the Intelligence Advanced Research Projects Activity supported this work in part.

References

1. I. Sutherland, ''Sketchpad: A Man–Machine Graphical Communication System,'' *Proc. AFIPS Spring Joint Computer Conf.*, 1963, Thomson Book Co., pp. 329-346.
2. D.C. Engelbart and W.K. English, ''A Research Center for Augmenting Human Intellect,'' *AFIPS Conf. Proc. 1968 Fall Joint Computer Conf.*, vol. 33, 1968, Thomson Book Co., pp. 395-410.
3. Xerox PARC, *Alto User's Handbook*, 1979.
4. J.J. LaViola, Jr. and R.C. Zeleznik, ''MathPad^2: A System for the Creation and Exploration of Mathematical Sketches,'' *Proc. Siggraph*, vol. 23, no. 3, ACM Press, 2004, pp. 432-440.
5. R. Zeleznik et al., ''MathPaper: Mathematical Sketching with Fluid Interaction Support for Online Computation,'' *9th Int'l Symp. Smart Graphics*, LNCS 5166, 2008, Springer, pp. 20-32.
6. C. Li et al., ''AlgoSketch: Algorithm Sketching and Interactive Computation,'' *Proc. Eurographics Workshop Sketch-Based Interfaces and Modeling*, Eurographics Assoc., 2008, pp. 175-182.
7. D. Tenneson, *Interpretation of Molecule Conformations from Drawn Diagrams*, doctoral dissertation, Brown Univ., 2008.
8. R. Zeleznik, K. Herndon, and J. Hughes, ''Sketch: An Interface for Sketching 3D Scenes,'' *Proc. Siggraph*, ACM Press, 1996, pp. 163-170.
9. R. Zeleznik et al., ''Lineogrammer: Creating Diagrams by Drawing,'' *Proc. ACM Symp. User Interface Software and Technology* (UIST), ACM Press, 2008, pp. 161-170.
10. G. Kurtenback and W. Buxton, ''The Limits of Expert Performance Using Hierarchic Marking Menus,'' *Proc. Conf. Human Factors in Computing Systems* (Interchi), ACM Press, 1993, pp. 482-487.
11. R. Zeleznik, T. Miller, and C. Li, ''Designing UI Techniques for Handwritten Mathematics,'' *Proc. Eurographics Workshop Sketch-Based Interfaces and Modeling*, AK Peters, 2007, pp. 91-98.
12. J.J. LaViola, Jr. et al., ''Evaluation of Techniques for Visualizing Mathematical Expression Recognition Results'' *Graphics Interface*, Canadian Human–Computer Comm. Soc., May 2008, pp. 131-138.
13. R. Davis, ''Magic Paper: Sketch-Understanding Research,'' *Computer*, vol. 40, no. 9, 2007, pp. 34-41.

Contact author Robert Zeleznik at bcz@cs.brown.edu.

Contact editor William I. Grosky at wgrosky@umich.edu and editor Utz Westermann at utz.westermann@gmail.com.



Authorized licensed use limited to: IEEE Xplore. Downloaded on January 14, 2009 at 00:04 from IEEE Xplore. Restrictions apply.