2007 Invited Workshop on Pen Computing

Workshop Summary

March 26-28, 2007
Brown University
Department of Computer Science

Summary compiled by
Richard Davis, University of California at Berkeley
Rosemary Simpson, Brown University

Introduction

In March of 2007, Microsoft Corporation, Brown University, and the United States Disruptive Technology Office brought together approximately 30 pen-computing researchers from universities and industrial research labs around the world. These researchers were drawn in by a common but elusive vision of "pen-centric" computing. This document gives a summary of the public discussions at the workshop.

"Pen-centric" computing is a vision of pen computing in which the pen is more than a substitute for the mouse. Pen-centric computing, as defined in this workshop, takes advantage of human skill with the pen, particularly the ability to express smooth, detailed paths quickly; while it often focuses on the creation and manipulation of diagrammatic languages, it is appropriate for many tasks in which drawn input is advantageous.

The roots of this vision trace back to 1963, when Ivan Sutherland presented his SketchPad system, a light-pen-based application that is the ancestor of modern CAD systems. This feat was quickly followed by similar systems for creating circuit diagrams, mathematical expressions, and architectural drawings. Though hardware systems were too primitive at the time to realize the vision of these pioneers, the dream stayed alive.

Thirty years later, once more powerful hardware became available, the first commercial systems were created. The Wang Freestyle incorporated a pen into its office productivity applications. Go Corporation's PenPoint, Apple's Newton, and Microsoft's Pen Windows made mobile pen-centric computing possible on a large scale for the first time. However, none of these systems found commercial success,

and only a few (such as the Palm Pilot and the Anoto pen) survive today.

The attendees at this workshop are united by the question, "Why did pen-centric computing fail?" The reasons appear to be many. Many of the above companies were plagued by business and patent problems. The recognition algorithms of ten years ago were immature, and computer hardware was not sufficiently powerful to perform recognition quickly. The bottom line, however, is that none of the above tools met users' needs at a price point low enough to bring about widespread adoption.

Today, there are signs that pen-centric computing is closer to becoming a reality. Microsoft's Tablet PC has established a beachhead as a commercially viable platform for pen-centric applications. Batteries, displays, and computing hardware continue to improve. There is increasing demand for software that manipulates complex data, such as images, 3D models, and video, which may benefit from pen input. Also, algorithms for handwriting and voice recognition have improved significantly and new machine learning algorithms are available. With all this new technology available, the community is faced with the question of how to put it to work. And indeed, the question of *what work* to do with it remains unanswered.

To address these questions, the workshop gathered researchers with a number of different perspectives. Many are engaged primarily in the development of systems that interpret various visual languages, including mathematical expressions (Zeleznik, Miller, LaViola, van Dam, Tausky & Lank), chemical diagrams (Zeleznik, Miller, & van Dam), digital circuits (Alvarado), analog circuits (Sezgin, Stahovich), dynamic systems (Simulink) diagrams (Stahovich), mechanical systems (Stahovich), engineering drawings (Varley), concept maps (Saund), data structures (Koile), probabilistic networks (Sezgin graphs (Sezgin), and a host of others (using Hammond's LADDER framework). Other researchers are more engaged in the human side of the interface, with experience investigating new ink application metaphors and low-level interaction techniques (Buxton, Gross & Do, Alexa & Nealen, Markosian & Wang, Landay & Davis, Zhai). Also present were several industry and government representatives who gave valuable perspectives (Bricklin, Buxton, Morrison, & Oka).

The pages that follow present a detailed account of discussions held during the workshop. The first section lists areas of agreement, points that were generally accepted by all present. The next section presents

areas of contention between participants. In most cases, this contention is brought about by the conflicting demands of pen-centric computing, and the section is organized as a set of axes along which any real system must strike a balance. Following this, a discussion of future directions addresses community as well as research issues. The document ends with conclusions and a short list of technologies and formats mentioned during the workshop.

Areas of Agreement

Although a broad range of perspectives was present at the workshop, a number of areas showed general agreement, especially the need to identify good applications of pen-centric computing and to focus efforts on those applications. There was also consensus that the absence of conventions for low-level interaction techniques in pen-centric computing is hindering the development of new applications. Handling of recognition errors was also widely recognized as a common interface problem. Finally, most workshop attendees recognized the need to continue exploring new hardware configurations for pen-centric computing.

Application area identification

The attendees present were not trying to identify the "killer app" that would create demand for pen-centric computing. In reference to the search for a killer app, one participant pointed out, "You'll know it when you find it." Instead, participants thought it best to look for tasks in which the absence of a pen is noticeable, perhaps *painful*. The pen becomes promising when the absence of a drawing tool is painful, *i.e.,* when a keyboard and mouse interface focus too much of a user's attention on precision. The pen is also attractive when being tied to a keyboard and mouse is painful, as in many mobile applications. Some pointed out that the pen was appealing when manipulating data with many dimensions, *e.g.,* whenever a time dimension is added.

Many of those present found education to be a particularly interesting application area. Students are encouraged to draw as part of learning many subjects, and pen-centric applications have the potential to make "frightening" or "boring" disciplines like science, math, and engineering more engaging. Much of the research in this area gives students the ability to create diagrams and to explore them using simulation and animation (*e.g.,* logic circuits (Alvarado), music (Zeleznik & Miller), calculus (LaViola), graph algorithms (Sezgin), and other domains (Hammond)). Other research focuses on the development of tutoring systems that can recognize students' drawings and give appropriate help when they are solving problems

(*e.g.,* Stahovich's tutoring systems for analog circuits and statics). There were also examples of systems that transform diagrams into more physically accurate representations (*e.g.,* chemical diagrams (Zeleznik & Miller)), and systems that collect and categorize student responses during lecture to enable instructor feedback (*e.g.,* data structures, (Koile)).

Participants also found design applications to be a promising area. Many participants presented systems that support faster drafting or editing of designs. FiberMesh (Alexa & Nealen) and Jot (Wang & Markosian) supported drafting of 3D models, and SilSketch (Alexa & Nealen) allowed 3D models to be posed by manipulating silhouettes. Stahovich presented three systems that used sketching as a fast input to simulation systems (AC-SPARC for analog circuits, Sim-U-Sketch for Simulink models, and VibroSketch for mechanical systems). Varley presented ParSketch for fast production of mechanical drawings. Finally, Tausky & Lank presented MathBrush for fast input of mathematical expressions into a computer algebra system.

Rough sketches were also seen as particularly useful during early design stages. Participants pointed out that individuals think and communicate better with rough sketches than they can with formal computer models. Also, teams can collaborate better around rough prototypes that they can manipulate together. Several participants presented systems that took advantage of these properties of sketches. Gross & Do presented many examples of such systems, including VR Sketchpad for collaborative annotation and lighting design of 3D architectural models, and Design-o-saur, Furniture factory, and FlatCAD for fast sketching and fabrication of rough 3D objects. They also presented the Electronic Cocktail Napkin, in which users compose their own rough spatial grammars for exploring complex systems. Landay and Davis also presented two systems, DENIM for web site designs and K-Sketch for short animations, both of which left content in rough form to foster creative thinking.

Finally, there was a general consensus that the existence of pen-centric games would spur the adoption of pen-centric computing techniques. One example of an attempt to build a professional-quality pen-centric game was given (InkBattle). Simpler (possibly end-user created?) games such as "Line Rider" are also promising.

Well engineered low-level interaction language

Many participants lamented the continuing confusion over which low-level interaction methods to use in pen-centric applications. Many

techniques have been proposed, but they are often difficult to generalize. The community needs to rally around a set of techniques in order to reduce the design and development time of new applications and to reduce the time needed to learn a new interface. Several interaction techniques were presented at the workshop including flow selection or "Cheshire Cat" selection (Gross & Do), overtracing (for selection, emphasis, or editing) (Gross & Do, Alexa & Nealen, Wang & Markosian), use of pressure (Gross & Do, Alexa and Nealen), a transparent window (for capturing traces of screen items) (Gross & Do), and a flick + mnemonic method for executing shortcuts (Zeleznik & Miller).

Recognition technology improvement

Participants were in general agreement that recognition technology was flawed and that the interfaces for recovering from errors could make or break a pen-centric application. Interfaces should make it easy to correct recognition problems and should avoid depending on correct recognition if possible. Some participants (Alvarado, Landay) proposed the use of Wizard of Oz testing to help designers observe the effect of recognition errors on users and design appropriate recovery interfaces.

Continued exploration of new hardware configurations

Though most participants designed their software for "slate"-style devices (*e.*g., the Tablet PC without the keyboard), no participant thought that pen-centric computing should be limited to such devices. Participants expressed interest in wall displays, table-top displays (some with camera sensors), multi-modal systems (usually speech and pen), and multi touch. Some participants (Sezgin, Stahovich) were engaged in (or attempting to engage in) research with such systems.

Areas of Contention

Though there were many points of agreement between participants, there was also a great deal of spirited discussion. Because of their different perspectives, many participants chose different tradeoffs between the many conflicting demands of pen-centric applications. Here, we list the most important areas of disagreement that emerged during the workshop and document some of the tradeoffs chosen by various participants.

Mimic pen and paper vs. move beyond pen and paper

Should pen-centric applications seek to take advantage of users' experience and skill with traditional pen and paper, or should these applications seek to provide a new kind of experience? Participants

saw advantages to both options and were split regarding which deserved more attention at present. Some wanted to move to a day when humans and computers can recognize the same diagrams. This preserves the free-form nature of sketching which stimulates creativity. If the systems can recognize existing diagrammatic languages, they can also take advantage of users' existing knowledge, making them easier to learn and use.

On the other hand, some participants noted that constraining or streamlining an input language may allow systems to recognize it more accurately or remove the need for recognition technology altogether. Simplifying an input language can also allow users to sketch diagrams faster. A few participants felt that holding to old diagrammatic conventions was not even important, since technology constraints could influence the evolution of diagrammatic languages. In any case, most participants agreed the choice of whether or not to streamline an input language should be determined ultimately by the tasks users need to accomplish with a particular system.

Training vs. "walk up and use" interfaces

A related question is how much time users should spend learning to use a pen-centric application. Some participants stressed that there are times when the benefits of learning a new, streamlined input language will outweigh the costs. An analogous example is a bicycle: the benefits of bicycles over tricycles justify the time spent learning to ride a bicycle. The question then arises, what form should training take, and can it be made more efficient or enjoyable? Zeleznik and Miller created training sheets that guided users through the process of creating visual language elements and provided links to more detailed help. For commands issues through menus, they also tried showing animations of gestures that could accomplish the same command. Zhai pointed out that games might be useful as training devices.

However, many users have little or no time for training. Zhai pointed out that most users will not take time to learn a new method for entering text. Koile wanted to avoid interrupting the flow of her classes with training. Saund wanted users to become proficient with his tool after only a few minutes of observing another person using it. Building a system that requires no training is particularly challenging. Current recognition technology is seldom up to the challenge, but several participants presented designs that kept input languages simple and provided additional functionality through special widgets (Wang & Markosian, Landay & Davis, Zeleznik & Miller).

Leave things rough vs. beautify

Another related question, specifically for systems that support text input or diagramming, is whether a pen-centric interface should leave user input in rough form or convert it to a more precise or polished form. There were many voices in support of leaving content in rough form. As mentioned above, the imprecision of a drawing can often facilitate the expression of an imprecise idea. This fact seemed to be a factor in the demise of structured editors that were a popular research topic at one time. Bricklin noted that Go's PenPoint supported handwriting recognition, but it was slow, inaccurate, and in the end, few people used it. Zeleznik and Miller conducted a study in which they compared interfaces that beautified user-drawn math expressions or left them in rough form. They found that leaving input in rough form was preferred by users; though placing a small, beautified version near the rough version was often helpful.

Other users pointed out once again that the decision of whether or not to beautify is dependent on the tasks that users are meant to accomplish with a particular system. Several users presented systems that did beautify user input after it was created. Saund was interested in moving back and forth between rough and beautified forms.

Task-specific interfaces vs. reusable components

The need for designers of pen-centric interfaces to pay attention to user tasks was a recurring theme in this workshop. Some speculated that pen-centric systems have been unsuccessful in part because designers and developers have not focused enough attention on meeting real user needs. Bricklin stated that Go PenPoint failed in part because no vendor provided an application that met a real user need at an affordable price point. For this reason, it is not surprising that many of the interfaces presented were designed with specific tasks in mind.

Yet there is good reason to focus attention on creating reusable components that interoperate elegantly. Go's PenPoint, for example, provided support for sharing ink data between components running in different parts of the same screen. Gross and Do noted that a "generic" inking interface is important, because users often do not know the purpose of a drawing at the time they create it. Another participant pointed out that it is common in the real world for people to appropriate a tool intended for one context for use in another. Landay added that pen-centric interfaces can be designed around this phenomenon using activity-based design methodologies. Thus, the

need to focus on real user tasks does not prevent designers from using reusable components, though it may complicate the design process.

Explicit vs. implicit modes in inking

No participant claimed that modes could be eliminated altogether, but should designers attempt to switch between them without explicit user commands? Much of the work presented at the conference can be differentiated by how it answered this question. Landay & Davis favored explicit mode switching and pointed to research that compares different methods for switching modes. Alexa & Nealen designed an application with implicit modes but added explicit modes later in response to user confusion.

Implicit modes remain attractive for many reasons however. If a system can detect users' intent accurately, implicit switching can allow more fluid interaction than explicit switching. Zeleznik & Miller explored this type of mode switching in their Fluid Inking system. Saund presented a system that attempts to infer a user's desired mode, but prompts the user for input when their desire is unclear.

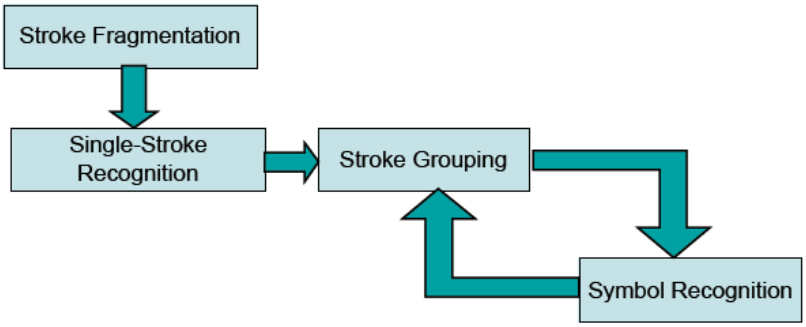
Recognition techniques

Some participants gave details about their approach to segmentation, recognition, and determining 3-D structure from 2-D drawings. We close this section with a very brief comparison of the recognition approaches presented at the workshop.

Common threads: Participants generally preferred recognition technologies that required no training. Hammond pointed out that training should be avoided because many students might use a single machine in educational environments. Some, however, found that training was unavoidable (*e.g.,* Tausky & Lank). van Dam mentioned that it was possible to avoid training but still adapt to a user's style by watching how recognition errors are corrected.

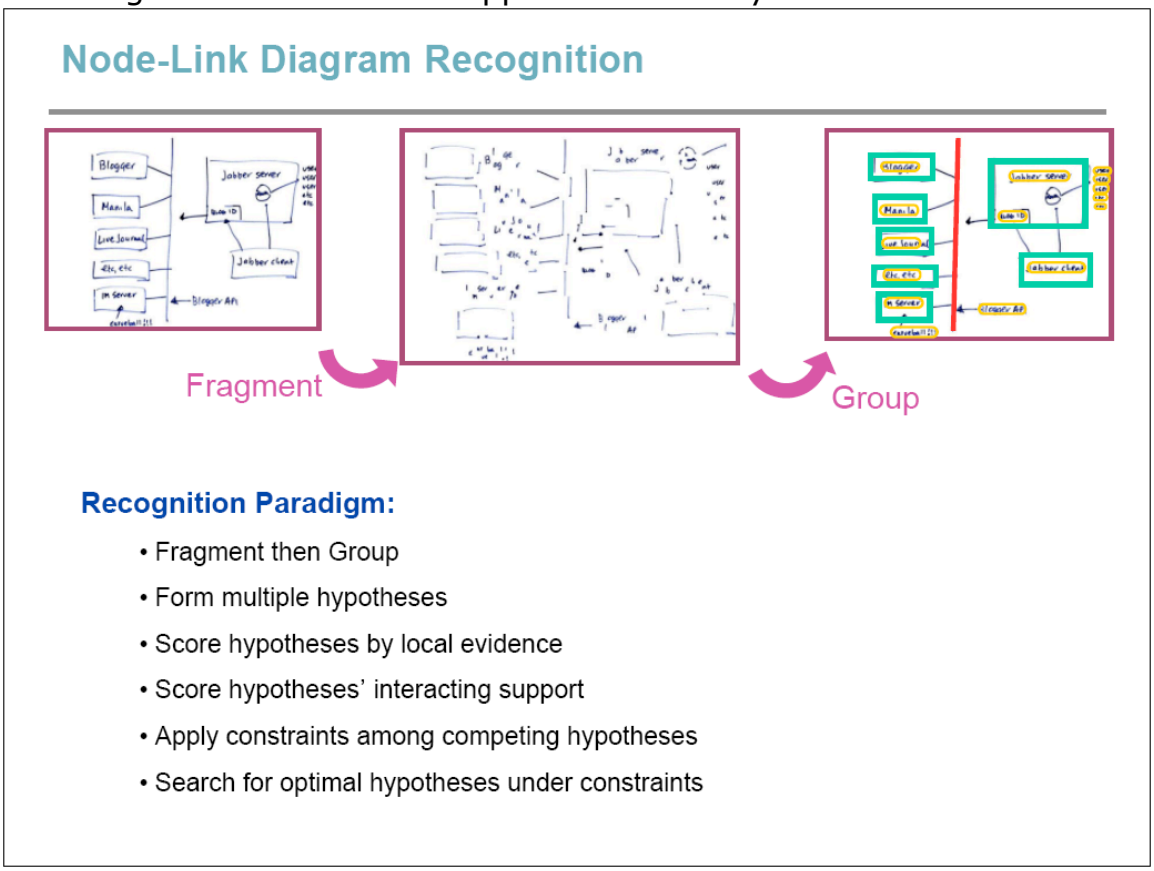
Most participants ignored timing information in their recognition algorithms, but a few thought it was important to take advantage of timing. Sezgin used Hidden Markov Models and Dynamic Bayesian Networks to learn temporal patterns in individuals' stroke orderings, and these patterns were used to improve recognition accuracy. Varley used stroke order in interpreting engineering drawings. Stahovich used timing information to segment lines.

Specific approaches: Alvarado used a single-stroke classifier as a pre-recognition step, followed by grouping step. She diagrammed her approach as follows.

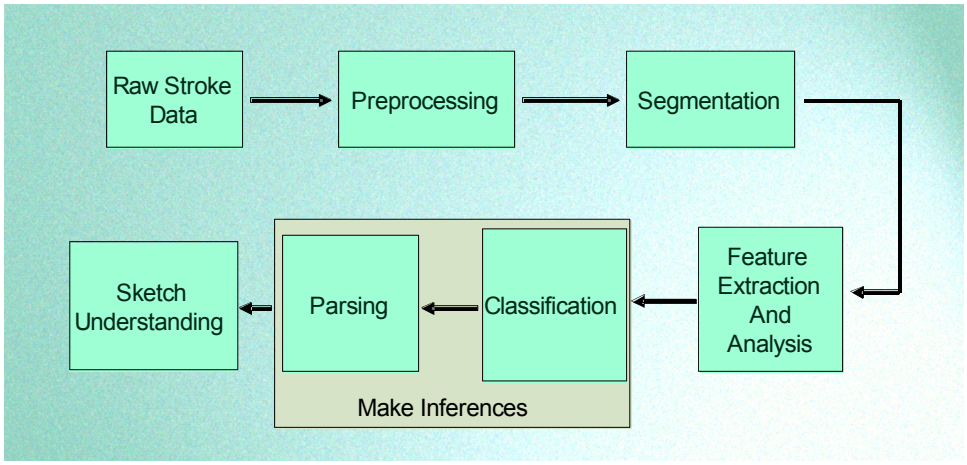


Sezgin used a single-stroke classifier that took advantage of individuals' temporal ordering patterns (as mentioned above).

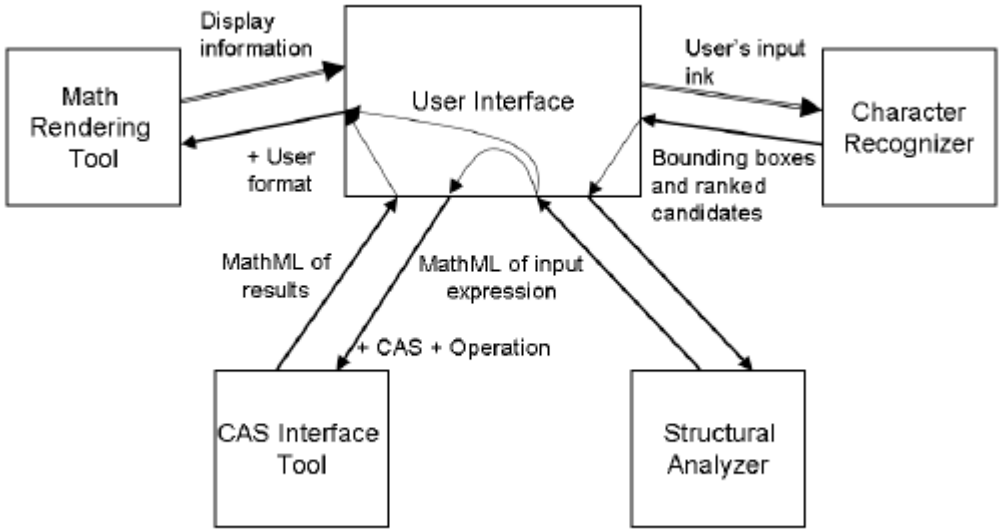
Saund used perceptual grouping rules and structural modeling. The following slide described his approach concisely.



In a pre-workshop presentation at UCLA van Dam gave the following picture of his group's approach to recognition.



Tausky & Lank gave this picture.



Stahovich also gave a detailed account of his approach. He starts with segmentation (timing base). He then uses one of three approaches to symbol recognition: a bitmap-based method, a structural-representation-based method, and another method based on features extracted from structural representation (with a Naïve Bayesian approach). The next step is parsing, which is performed either using geometric differences or ink density. Finally, an error correction step uses the context of diagram to detect and fix problems.

Participants with systems that produced 3-D shapes from 2-D sketches also had a variety of approaches. Varley used a multi-stroke recognizer with geometric constraint satisfaction. Alexa & Nealen would alternate between inflating curves and using them to define or modify feature lines. Wang and Markosian would also inflate (and smooth) some user drawn-curves. They added other curves for cutting out geometry and widgets for modifying geometry.

Future Directions

When it came time to discuss future directions, participants had many opinions with varying levels of agreement. Here, we highlight two themes that emerged in during discussions: common research obstacles that we face and the need for collaboration. We then list participants' recommendations for future research direction in three areas, human-computer interaction, software (particularly recognition software), and hardware.

Common Research Obstacles

Confusion over what pen applications and technologies will be considered valuable continues to be a major problem. Some expressed frustration that no compelling applications have been created in 40 years of research. A few felt that new design methods were needed, such as activity-based computing (Landay), or pen-based "Laws of action" (Zhai). Some contented that compelling applications have been discovered (e.g., Graffiti for text input) but patent issues got in the way. In any case, there was a general sense that the lack of compelling applications made it increasingly difficult to obtain support for pen-centric computing research.

Another obstacle we face is the popularity of mobile computing, which is overshadowing laptop-based pen-centric computing. There are several orders of magnitude more small, mobile devices being sold than Tablet PCs. With such small drawing areas, these mobile devices are not optimal for pen-centric computing.

Need for Collaboration

Another common theme during discussion was collaboration. There was little agreement on what form this collaboration should take, but many had the sense that the community could produce higher-quality results if members could build on each other's work. Buxton stressed that Sutherland was able to build SketchPad in 1963 because many people at Lincoln Labs were sharing their work and building on each others' techniques. To make this happen today, Buxton argued that we need a common vision of where we are going, how to get there,

and how to give members the necessary incentive to contribute to the community.

Several participants were engaged the building tools and toolkits that they hoped to share with the community. Sezgin described a probabilistic graphical model builder for recognition systems. Hammond presented GUILD and LADDER which greatly simplified the development of a class of pen-centric applications. Zeleznik, Miller, and van Dam also mentioned that they were working on an application framework called the *Pad toolkit.

In the last session of the workshop, there was some discussion about the requirements of a common tool/SDK/framework for pen-centric applications. Saund expressed a desire for a good system that abstracts away hardware, analyzes ink, and computes trajectories of ink strokes. Alvarado stressed that it should define a set of data types that could be shared across applications, and asked for a shared ink format. Landay warned that any such work should take into account previous work in inking toolkits and file formats, including Satin, Jot, and InkML. Another participant pointed out that similar efforts have failed in the virtual reality community. In order for such a project to succeed, it needs to involve multiple researchers with diverse (but overlapping) interests, support multiple platforms, gather ideas from other similar systems, and foster the creation of shared code and tools.

Research Directions

Ideas for future research directions varied widely. Here, we attempt to collect them into a coherent structure, but do not explain them in detail. Most have their roots in discussions that were presented earlier in this document.

Human-Computer Interaction

- **User**
  - Training methods for character, symbol, and gesture sets
  - Activity-based approach to design (focuses on three levels: activities, actions, and operations)
- **User interface**
  - Low level interaction techniques
    - Distinguishing commands from content

        - Reserved gestures, reserved areas, or "punctuation"?
    - Timing of recognition
        - Continuously, after time delay, after every stroke, or when user initiates?
    - Novel techniques (e.g., multi-modal w/speech)
    - Using "Laws of Action" to design techniques
- Issues with feedback
    - How to use subtle forms of feedback (*e.g.,* color, transparency) effectively?
    - How much should users be exposed to recognition process?
    - How should alternative interpretations be presented?
    - How can feedback indicate pen mode?
    - Handling imperfect recognition
- Higher level design issues
    - New metaphors for manipulating things other than paper documents
    - Enhancing existing apps vs. creating new apps
    - Visual Languages: how to enhance existing notations?
    - How to go about shortcut customization?

Software

- **Recognition Techniques**
    - Character, symbol, and simple gesture recognition
    - 2D expression parsing
        - chunking and phrasing
    - segmentation/containerization
        - defining scope for groups
    - Use of temporal patterns in recognition
    - Which techniques work best for which problems?
        - rule-based? graph matching? Bayes nets? Markov Random Fields? Conditional Random Fields? hierarchical grouping?
    - What techniques are suitable for arbitrating among alternative interpretations?
    - What are the best ways to parse network-like diagrams?
    - How can similar sketches be aggregated?
- **Software Engineering**
    - Managing complexity of data (links between possible interpretations and multiple renderings)
    - How to stay responsive to user input in presence of compute-intensive recognition?

Hardware

- Drawing surfaces: precision, surface feel, pressure
- Buttons on stylus and tablet
- Parallax for display tablets
- Better designs for attaching (or detaching) keyboards
- Weight, battery life, and robustness
- E-paper

Conclusions

This workshop brought together researchers and representatives from industry and government to discuss the future of pen-centric computing. Participants shared a definition for pen-centric computing, but had very different perspectives. We have outlined points of agreement between researchers, areas of differences among them, and their thoughts on future directions.

Most participants strongly agreed that the workshop was valuable, and many commented on how important it is to get support from colleagues at such a venue. Most also agreed that the workshop had the right length and pace, with ample opportunities to pose questions and try demos. Many commented on the value of bringing in Dan Bricklin's and Bill Buxton's historical perspectives, as well. Finally, there was nearly universal approval for the effort put into providing working demonstrations during the workshop.

If similar events were to be held in the future, however, there were clear areas for improvement. Many left with a sense that the community has not united itself around common goals, methods, or tools, and had hoped that the workshop could accomplish more. Some suggested allocating more time to discussion and brainstorming sessions.
Others complained that public discussion during the workshop was dominated by senior figures and focused too much on the past. Feelings were mixed about the value of the final panel discussion for similar reasons.

In spite of these difficulties, it is clear that workshops such as this one are vital to the heath of the pen-centric research community. Every workshop participant surveyed said that they would like to attend a similar event in the future. Who will step forward and take responsibility for the next event?

Appendix: Technologies, Formats, and Tools Mentioned

- Anoto Pen (used by Stahovich)
  - http://www.anoto.com/
- BNT – Bayes Net Toolbox for Matlab (used by Sezgin)
  - http://bnt.sourceforge.net/
- GMTK – University of Washington's Graphical Models Toolkit (used by Sezgin)
  - http://ssli.ee.washington.edu/~bilmes/gmtk/
- ILiad from iRex Technologies (eBook that uses state-of-the-art digital paper)
  - http://www.irextechnologies.com/
- InkBattle – Pen-centric game
  - http://inkbattle.marctenbosch.com/
- InkML ink format (mentioned by Landay)
  - http://www.w3.org/TR/InkML/
- Jot ink format (mentioned by Landay)
  - http://unipen.nici.kun.nl/jot.html
- Line Rider – Pen-centric game
  - http://www.official-linerider.com/play.html
- MathML (used by Tausky & Lank)
  - http://www.w3.org/Math/
- Microsoft Tablet PC API (used by many)
  - SDK V1.7 - http://www.microsoft.com/downloads/details.aspx?FamilyId=B46D4B83-A821-40BC-AA85-C9EE3D6E9699&displaylang=en
  - Vista - http://msdn2.microsoft.com/en-us/library/ms696350.aspx
  - Microsoft recognizer is great, but no panacea
    - Has trouble recognizing isolated symbols (*e.g.,* math symbols, parentheses in Scheme code, anything other than English words)
- PNL - Intel's Probabilistic Networks Library (used by Sezgin)
  - http://www.intel.com/technology/computing/pnl/
- Satin ink toolkit (mentioned by Landay)
  - http://dub.washington.edu/projects/satin/
- iLiad from iRex
  - http://www.irextechnologies.com/