The InfoPad User Interface

Andrew Burstein, Allan Christian Long, Jr., Shankar Narayanaswamy,
Richard Han, Robert W. Brodersen
Department of Electrical Engineering and Computer Sciences
The University of California at Berkeley, Cory Hall
Berkeley, CA 94720
burstein@eecs.berkeley.edu [other authors may be reached through http://infopad.eecs.berkeley.edu/]

ABSTRACT

We have shown a prototype user interface for the InfoPad, a portable terminal with multi-modal input and multimedia output. The InfoPad's main features are:

- Portability
- Continuous network connectivity using a high-bandwidth radio link
- Pen input with handwriting recognition
- Audio input with speech recognition
- Full-motion video playback with synchronized audio
- Text/Graphics display

The InfoPad's unique input and output characteristics offer challenges and opportunities for user interface design. We have implemented an API for network access to audio, pen, graphics, and video data; we have also implemented speech and handwriting recognizers along with programming interfaces and toolkits. We are prototyping applications and user interfaces to explore how handwriting and voice recognition may best be used together. We believe that the lessons we will learn can be applied to other multi-modal platforms.

INTRODUCTION

The Infopad project is a large, multi-disciplinary research effort involving a number of faculty and graduate students. Its goal is to build portable multimedia terminals connected to the network via high-bandwidth radio links (2 Mbits/sec downlinks and 64 kBits/sec uplinks) in a picocellular environment. The research encompasses low-power integrated circuit design, high-frequency radio design, network design, protocol and coding design, handwriting and speech recognition, and user interface design.

Although we plan for the InfoPad to have a color display, current LCD technology is still expensive in both money and power. Therefore the present implementation uses a 9-inch monochrome text and graphics screen and a 4-inch color screen for full-motion video. Future versions may have large, color LCD displays or head-mounted displays. InfoPad also supports audio input and output. A keyboard is not included because it would add bulk, weight, and cost to the terminal.

InfoPad uniquely combines portability, network connectivity, and state-of-the-art interface technology. We believe the ideal user interfaces for InfoPad are fundamentally different from WIMP (window, icon, mouse, and pointer) interfaces.

The three technologies InfoPad brings together make it an ideal platform for many application areas not well supported by traditional workstations:

- Groupware (shared white boards, collaborative design)
- Note taking and document annotation
- Mobile, multimedia information browsing and retrieval
- Personal information management
- Personal communications

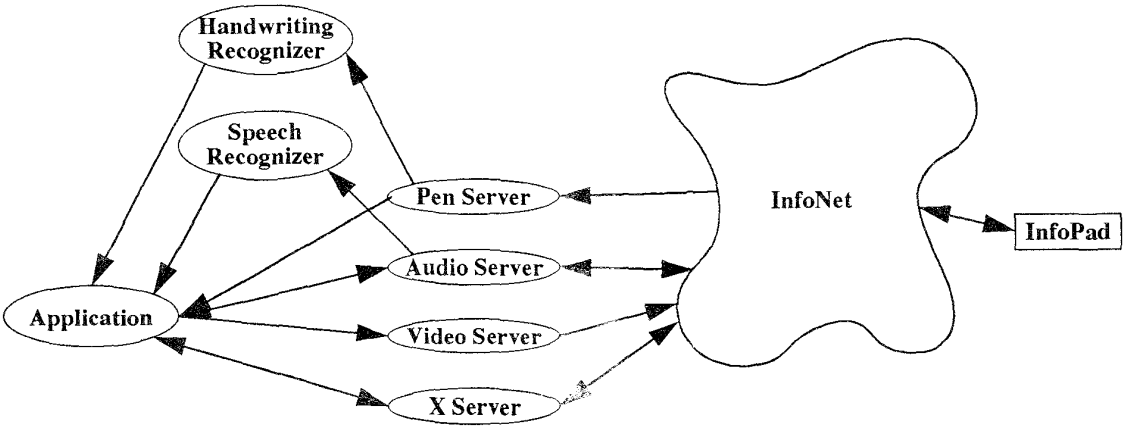
These applications are inherently mobile, are well-suited to the use of pointing devices in the user interface, and do not require mass text entry, which we feel is more efficiently done with a keyboard. For some of these applications, we can use existing workstation versions as starting points; for others, we must develop interfaces from the ground up.

We do not intend the InfoPad to surpass workstations in areas in which they already excel. For example, programming is more time-efficient on a desktop workstation than on an InfoPad since a proficient typist can type at least twice as fast as he or she can write. The InfoPad is a complement to workstations, not a competitor.

Since the InfoPad's input modalities are so different from those of traditional computers, we believe that even where the same application (e.g. e-mail, text editing) is re-implemented for the InfoPad, the user interface should be redesigned to take advantage of InfoPad's strengths. For example, although one could easily write a keyboard widget to enter text into one's favorite text editor, it would be better to implement a text editor that used the InfoPad's speech and handwriting capabilities. Rather than relying on ascii, which is suitable to keyboard entry, as the primary data type, applications will also store phonetic (speech) and gesture and ink (pen) data types.

Designing a user interface for the InfoPad's input modalities raises many questions. Some of the issues we intend to address are:

1063-6390/95 $4.00 © 1995 IEEE



- How should a user interface deal with the uncertainty inherent in handwriting and speech recognition?
- To what extent can or should pen and voice be used as substitutes for mouse and keyboard?
- How can handwriting and speech recognition be used synergistically?
- What is the proper input focus model for multiple applications using speech recognition?

SYSTEM SOFTWARE ARCHITECTURE

Due to size and cost constraints on the InfoPad terminal, we moved as much processing as possible off the terminal and onto the network. The greatest advantage of this architecture is that the InfoPad has access to massive computational power, allowing the InfoPad to be "smarter" (e.g. with handwriting and speech recognition) than other portable devices. The most significant disadvantage to the user interface is latency. We are optimizing our network software so that network latency does not cause our interface to be unresponsive.

The user interface software architecture is shown in Figure 1. Applications interface to the network through a set of abstracted "type servers," one for each of the basic data types: pen, audio, video, and text/graphics. In addition, applications interact with network based handwriting and speech recognizers using a standard API.

PEN INPUT AND HANDWRITING RECOGNITION

Applications may use pen data from the InfoPad in three ways. First, they may treat the pen as a mouse, reading mouse events from the X Window System. Second, applications can bypass X to get higher resolution data from the pen. For example, a drawing program would likely want as much resolution as possible. Third, an application can use a handwriting recognizer, described below, to treat the pen almost like a keyboard.

We use a Logitech Gazelle digitizer which provides about 100 pixels/sec at 200 lines/inch resolution. Each pixel is represented by a 5 byte packet. The packets are sent unmodified over the radio link to the base station using 500 bytes/sec of bandwidth. The gateway program running on the basestation sends these packets to the pen server which decodes them into x and y coordinates, button-1 status, and button-2 status.

The pen server spatially subsamples this data to screen resolution and sends it to the X server to control the mouse cursor and exactly replicate the functionality of a 2 button mouse. In addition, the pen server creates an Internet socket at a well-known address and makes full-resolution pixel data available over this socket. Applications as well as the handwriting recognizer can connect to this socket simultaneously to obtain pen pixels. An Applications Programming Interface is provided.

Handwriting recognition is provided by a stand-alone program that runs on a compute-server on the network. This allows the use of compute-intensive algorithms in the recognition server. The current recognizer uses hidden Markov modeling to recognize printed characters. It uses simple geometric features such as slope, difference of slope, and y-coordinate sequences. Pre-processing applied before feature extraction includes subsampling, truncation, and normalization.

This recognizer allows the user to change the recognized vocabulary on the fly and returns a list of the top few candidates and their relative confidence levels. It connects to the pen server directly for full-resolution pen data and creates an Internet socket at a well-known address so that applications may connect and obtain recognized text.

An API is provided for application programmers to easily access and control the recognizer. An application may use more than one recognizer to customize the grammar and vocabulary for different contexts. A handwriting recognition widget provides an easy-to-use abstraction for graphical user interface builders. The widget accepts handwriting and allows the user to correct it before returning the recognized text to the application. The application does not need to be aware that data is being entered by pen rather than on a keyboard.

AUDIO I/O AND SPEECH RECOGNITION

InfoPad applications either use sampled audio as a data type in its own, or perform speech recognition on the audio input. InfoPad records and plays 8 bit mu-law encoded audio sampled at 8kHz (telephone quality audio). A standard 1/8 inch connector allows the user to wear a head-mounted speaker/microphone to help reduce background noise. Some applications such as telephony, voice mail, and voice annotation will use raw speech as a data type. Most

other InfoPad-based applications will use speech recognition, along with pen input, as a source of command and control instructions. In general, speech recognition will not be used as a source of mass text entry because of the difficulty of achieving high recognition accuracy; however, as better speech recognizers become available, automatic transcription of dictation may become feasible).

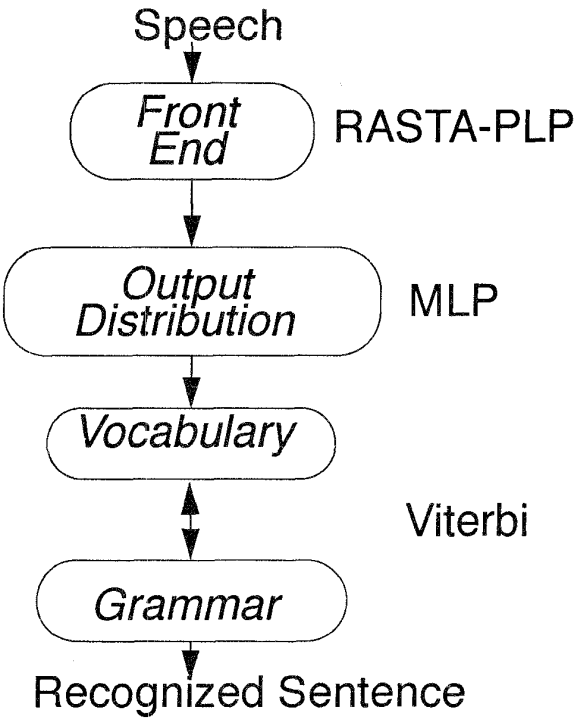
Much of the effort in developing speech recognition for the InfoPad has gone towards resolving two conflicting goals: ease of use versus recognition accuracy.

Ease of use applies to both the end user as well as the InfoPad application developer. For the user, ease of use entails being able to speak in a normal rhythm (connected speech recognition) using a large number of words (large vocabulary) with little or no training of the recognizer (speaker independent). Unfortunately, this set of conditions describes the most difficult speech recognition task.

A fourth parameter, the perplexity of the grammar, a measure of how many different words can follow any given word, also has a profound impact on recognition accuracy. For example, if the grammar allows any word in the vocabulary to follow any other word in the vocabulary, there will be more ways to incorrectly recognize a spoken sentence than if each word can only be followed by a few other words (assuming the sequence of words in the spoken sentence fits within the grammar to begin with). It is not straightforward how changing the perplexity of the grammar affects the system's ease of use. As long as the user's sentences fit within the rules of the grammar, the user will perceive no difference (except perhaps higher recognition accuracy); however, if the grammar is too tightly constrained then it will prevent the recognizer from identifying many spoken sentences. Unfortunately, the rules of natural language grammars (such as that of English) are not well enough understood at the present time to be incorporated into an automatic speech recognizer. Instead of using a natural language grammar, the InfoPad's speech recognizer uses a statistical model called a bigram grammar, which contains the probability of each word following each of the other words in the grammar.

Rather than creating a bigram model of English as a whole, the InfoPad strategy is to create different grammars and vocabularies for each application. To the user, it will seem like the speech recognizer has a very large vocabulary, but the recognizer only has to have a small part of the vocabulary active at a given time. In effect, each application is using its a priori knowledge of its own subject matter to help the speech recognizer. Ideally, applications should have different grammars and vocabularies for each state or mode of operation. For example, a video playing application might use different grammar and vocabularies when the user is watching a video than when the user is browsing through a list of titles because the program accepts different sets of commands in the different states.

This strategy - having customized vocabularies and grammars for each application - conflicts with the goal of making the applications programmer's job easier. From the programmer's perspective, the easiest to use recognizer is a black box that simply tells the program what words the speaker said. In order to make it easier for the programmer to couple the application with the speech recognizer, InfoPad uses a high level programming interface implemented in the tcl language, along with a graphical tool that the programmer uses to build vocabularies and grammars.The programming framework also contains widgets to aid customization and error recovery.



The speech recognizer uses the RASTA-PLP [1,2] algorithm to extract coefficients that are fed to a hybrid multi-layered perceptron (MLP) and hidden Markov model (HMM) algorithm [3]. RASTA-PLP has two main advantages over other front-end algorithms. First, because it weights frequency bands in a manner analogous to the sensitivity of the human cochlea, it encodes speech information in a more compact manner. Second, it is designed to filter out convolutional noise, so variations in the acoustic channel, for example changing the microphone, have minimal effect on recognition accuracy. The MLP, serves as a statistical model that estimates the probability that a 10 millisecond frame of speech is each of the phonemes (basic speech sounds) of the English language. Because the MLP is trained on a large set of speakers, it forms a speaker-independent model of English phonemes. The probability estimates from the MLP are fed to an HMM that contains the pronunciation information of the vocabulary and the word-to-word transition information of the grammar. Using the Viterbi algorithm, the recognizer the most likely sequence of spoken word, i.e. the most likely sentence. One of the main advantages of using the viterbi algorithm is that it can recognize continuous speech without having to explicitly identify boundaries between words. The InfoPad recognizer can also use a variation of the Viterbi algorithm that produces several best estimates of the sentence; in the event the first estimate is not correct, the application may present the user with the other choices, one of which is likely the correctly recognized sentence.

In keeping with InfoPad's network-based computational model, the speech recognizer is currently implemented as software (in C++ and tcl) running on network based workstations. By using a standard API,

InfoPad will be able to employ new speech recognizers as they are developed

In addition, we are developing low power hardware that implement the speech recognition algorithm while consuming a few tens of milliwatts. One pair of chips performs the RASTA-PLP algorithm. By including these chips on board the InfoPad, the uplink bandwidth may be reduced by about a factor of 10 when transmitting speech to a recognizer. By including another pair of chips, an MLP chip and a Viterbi chip, the entire speech recognizer can be implemented in the pad, giving two main benefits: first, the pad will be better able to function during periods when it is out of contact with the network. Second, by sending up recognized speech, the uplink bandwidth can be reduced by orders of magnitude. Including the entire recognizer on the pad would be suitable in future derivatives of InfoPad that have more on-pad functionality.

FULL-MOTION VIDEO

Video is becoming pervasive in digital documents, so we think InfoPad users should have access to video as well as text and graphics. For example, we want to support Mosaic access to video documents.

The InfoPad's color display is 18 bits deep with a resolution of 128x240 (using wide pixels for a 4:3 aspect ratio). On this screen, the InfoPad can play full-motion digitized video at 30 frames per second. We use vector quantization (VQ) encoding for two reasons. One is to compress the video to our radio downlink bandwidth of 1 Mbit/s. The other reason is that VQ (unlike many other encoding schemes such as MPEG) is tolerant of data errors introduced by the radio link. We plan to further explore the trade-offs between different compression schemes.

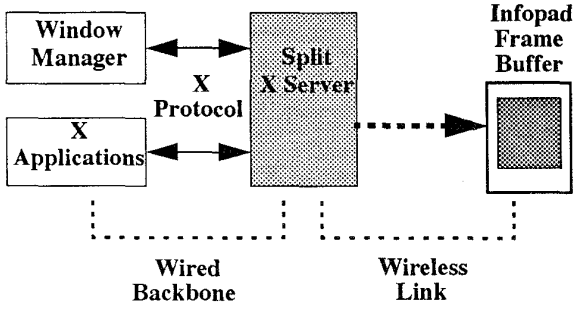
Two important network issues for video playback are bandwidth and jitter. In terms of bandwidth, the wireless link will be the bottleneck unless there are many Pads in a single cell. In the extreme case of 50 Pads/cell, the wired network would have to support 50 Mbits/s. Because of this (and because of latency) we are using ATM. Jitter is a problem because it causes the audio and video to become unsynchronized. At present, the audio and video streams are synchronized several network hops away from the Pad, which gives less than optimal performance. We are exploring different alternatives to improve playback, including adding synchronization to the gateway and the Pad itself.

TEXT/GRAPHICS SERVER

Windowed text/graphics in Infopad is accomplished through a modified X architecture. In particular, the low-power implementation of Infopad requires the X server to operate on the backbone remotely from the display hardware. Hence, we call our modified server the "split-X" server, where the split occurs at the lowest level of the X server, just prior to frame buffer writing.

We have chosen initially to send raw pixels over the wireless link to the monochrome text/graphics frame buffer. This bitmap data will suffer some of the non-idealities of this wireless channel, including delay, limited capacity, and especially corruption. Hence, part of our efforts include coding techniques to deal with these impairments.

Currently, we are exploring methods not only of compression, but also of error resiliency. The intent is to provide rapid response time with reasonable image quality for such typical tasks as menu pull-down, text editing and paging, and window management activities. In particular, we are implementing an algorithm which refreshes and cleans up the screen after corrupt data has been displayed.



APPLICATIONS

We are building applications to test our user interfaces and to demonstrate the usefulness of the InfoPad system. We are concentrating especially on applications that take advantage of both pen and speech input.

The first application is a Mosaic-like WWW client, which demonstrates the network access and retrieval, multimedia output, and recognition capabilities of the InfoPad. The second is a voice-driven command interface for the Magic integrated circuit layout editor. This demonstrates the use of voice commands in driving pre-written applications and can be used on regular workstations as well. The third application is a circuit schematic editor that will recognize text and schematic symbols drawn by the pen, as well as speech commands, to create and edit circuit schematics and simulate them in SPICE Other applications include a cross-network shared white board, a video browser, and voice mail

CONCLUSIONS AND FUTURE DIRECTIONS

Multimedia output, mobility, network connectivity, and recognized input make the InfoPad project unique. Pen and audio input, and in particular handwriting and speech recognition, make for a more natural user interface and will allow us to go beyond the traditional WIMP interface. We believe that the results we obtain from our exploration of pen and voice interaction techniques will be useful for designers of interfaces for many different applications on many different computers.

REFERENCES

[1] H. Hermansky, "Perceptual Linear Predictive (PLP) Analysis of Speech," *J. Acoust. Soc. Am.*, vol. 87, pp. 1738-1752, April 1990.

[2] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "RASTA-PLP Speech Analysis Technique," in *IEEE International Conf. Acoustics, Speech and Signal Processing* (New York, 1992), pp. 121-124.

[3] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," in *Proc. of the IEEE*, vol. 77, pp. 257-286, Feb. 1989.