R&D Strategies for Input Interfaces

Nils Klarlund

January 23, 2004

Background

My background in the area of alternative input interface research is unusual. For years, I have relentlessly searched for novel ways of offloading hands and arms when using the computer. I have built keyboards, pointing devices, and foot operated devices, and for more than a decade I have used such equipment on a daily basis.

My work has addressed the common problem of typing injuries and other upper extremity discomfort in ways that aim at being radically more effective than currently available offerings.

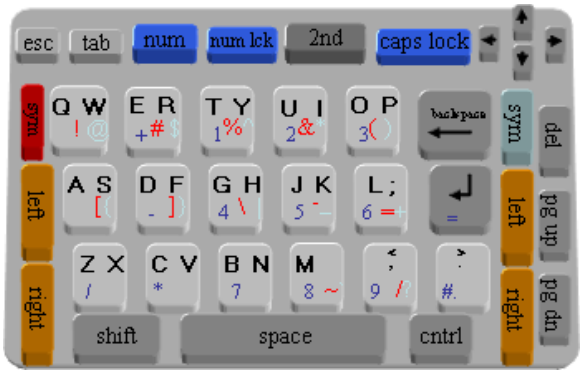
My efforts have resulted in a suite of intellectual property, valued at $3.1 million by an independent party. It was donated by AT&T to Carnegie Mellon University in December of 2003.

I have also pursued research that addresses more general user interface issues. I have formed my own company, Clairgrove LLC, that develops intellectual property in this area.

I would be very interested in exploring collaborations with traditional HCI researchers, where I can contribute unusual innovative skills and experience.

Example I. The input problem on small devices

How to efficiently input text on small devices is a substantially unsolved problem. In theory, the use of large language models, as in speech recognition, may drastically reduce the amount of information that needs to be transmitted by the user. The "key problem", so to speak, is to design a keyboard that takes advantage of this situation, while being efficient and familiar. I have conducted experiments with Michael Riley showing that the letters of the QWERTY keyboard layout can easily be spread over half as many keys as usual:



With this design, a PDA-sized, one-handed QWERTY keyboard, with normal-sized keys is achievable. The clustering principle is similar to that of the telephone keypad but much simpler and totally familiar. We are shown that language models can with striking accuracy disambiguate what is being typed, even in the presence of typing errors. And importantly, flash memory today allows large language models to be stored inexpensively on small devices.

For more on this subject, see my paper in the 2003 EACL Workshop on Text Entry.

Example II. The correction problem

Dictation systems are criticized for the errors they make. But the real problem is not so much accuracy anymore, but a lack of an effective user interface for correction. Solving this problem is fundamental to the success of dictation software.

Surprisingly, typing actually generates more words containing errors than a speech recognizer performing at a 5% error rate (to pick a number). Of course, typing errors are corrected mostly on-the-fly using a couple of editing keys, but this strategy does not work for speech recognition. An information-theoretic analysis of speech recognition errors, as will be known to anybody in the field, shows that the typing forced upon the user by current interfaces is excessive relative to the information that is really needed.

But just as problematic is the massive amounts of information (extracts from the lattice of possible utterances) that is presented when the user tries to correct an error.

Both these problems are *eminently* solvable, I believe—even to a degree that even a novice would immediately be able to correct errors in a way that is much more efficient than the current solution. The approach, however, demands that dictation systems no longer be add-ons, but integral parts of the user interface. In particular, the keyboard—while completely traditional, or perhaps made according to the solution above—must be adapted in minor ways to much better handle uncertain input.

Example III. The lag and undo issues

Most voice-operated systems in deployment are poorly engineered from a most elementary perspective: there is a perceptible and often annoying delay in the sys-

tem's response to user input. (Delays in response time as low as .2 seconds are perceivable.) Whether the subject is dictation systems or interactive, synthesized agents, it is *highly unnatural* that the state of the system, for example the words that appear in the text or the reaction of an avatar, does not in true real-time follow the user's input.

The technical issue to be overcome is that traditional software approaches rely on end-point detection before any events are propagated from the speech recognizer to the user interface system. Up to now, it has been too difficult to build systems that would tentatively react to fickle speech recognizer output in real time. I believe that this issue can be fully solved within a domain-specific program language. In fact, the principles are similar to those of my investigations into a programming language solution (developed from work by in collaboration with Giuseppe di Fabbrizio and Jennifer Beckham, see our Eurospeech 2001 paper) for addressing another persistent and elementary user interface issue: that of the lack of consistent undoability of misinterpreted or erroneous input in speech and IVR (Interactive Voice Response) user interfaces.

Example IV. The command and control problem

Another venue that in my opinion lies wide open concerns the segment of partially-enabled users (RSI sufferers) that dictation system vendors have recently pursued more vigorously. Annually, some 1.8 million U.S. workers are affected by musculoskeletar disorders (according to OSHA). Repetitive stress injuries are now the single largest cause of such problems in the United States, with annual costs in the tens of billions of dollars.

The command and control systems offered today in this receptive and motivated market segment suffer from two major deficiencies: (1) the "pause" technique for distinguishing commands and dictation is highly bothersome (every experienced user I've talked to seems to agree on this) and (2) the power of speech recognition is actually vastly underutilized—see my ICASSP 2003 paper on ShortTalk, the constructed editing language. Overcoming both issues, ShortTalk demonstrates that a command language largely distinct from English can be made for the control of computer such that routine work as in editing becomes much faster than by keyboard and mouse. Additionally, we are at or past the point that an array microphone (or perhaps a Jabra microphone embedded in the ear) can capture such a restricted language with near perfection. Moreover, the language can by be made so that word spotting will only very occasionally misinterpret English for commands.

For many professionals, text entry per se is only a minor part of the total amount of operations. Thus, a command and control system as outlined would be *extraordinarily compelling* to a variety of professionals suffering from computer pain. I use the word "extraordinarily" because this solution overcomes each of the major obstacles behind the use of speech recognition: (a) the need to wear a microphone, (b) the need to obsessively take care of the microphone status, (c) the inefficiency of editing with current command languages, and (d) the lack of privacy when using speech recognition (spilling commands to the next cubicle only reveals that one is

working, not one's thoughts).

ShortTalk has not been evaluated by users yet (and indeed awaits well-defined HCI studies); nevertheless, there are several people who have expressed their interest or outright enthusiasm for the approach. Generally speaking, these users are experienced professionals. I am quoting from some of the e-mails I have been sent unsolicited (and from a couple of Web sites):

> *VERY interesting, with much promise. Excellent work.* BJ
>
> *Hallelujah & Congratulations!... I have been thinking along these lines for years now.* EH
>
> *Very exciting video using ShortTalk inside of Emacs...* anonymous (emacswiki glob)
>
> *I would really like to try your ShortTalk product.*" AG
>
> *I got very interested in your approach...* AN
>
> *Interesting research and thinking about input devices.* Paul Marxhausen
>
> *I am very impressed with your work...I have to warn you, I'm really really really hoping I can motivate you to find a way so that I can try ShortTalk.* CE

These comments reinforce my view that speech recognition interfaces have not effectively served the potential of the core recognition technology—and there are lots of leads about how to proceed. I would hope to work with colleagues in HCI on further exploring speech recognition user interfaces, in particular in order to overcome many fundamental user issues.

There are also questions that relate to human cognition springing from my preliminary results in ShortTalk. I would like to pursue questions of language skills and tool use in interdisciplinary collaborations:

1. Is the use of a spoken command modality inherently less effective than the keyboard? Or, does it, as I hypothesize, offer substantially more bandwidth than the keyboard? In other words, when the trained human mind manipulates tools, does it prefer one modality or the other if there were a choice? (The experimental conditions must of course provide a set of operations that can be effectively expressed in both modalities after a period of training.)

2. Could it be that the spoken modality is the preferred one, but not for the reason commonly assumed, that language is more "natural, " but simply because symbolic information expressing operations can be sequenced faster through spoken acts than through movements of fingers?

3. Does it take more or less time to learn tasks that are expressed by voice instead of through keystrokes? (Earlier research may not have provided enough training for subjects to pick up voice skills, thus incorrectly concluding that the keyboard is inherently superior.)

4. Is the learning of a spoken command language facilitated more by it being a variation of a natural language or by operational efficiency?