INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 5, 371–388, 2002

A Listening Keyboard for Users with Motor Impairments— A Usability Study [1]

BILL MANARIS
Department of Computer Science, College of Charleston, 66 George Street, Charleston, SC 29424
manaris@cs.cofc.edu

VALANNE MACGYVERS
Department of Psychology, University of Louisiana at Lafayette, P.O. Box 43131, Lafayette LA 70504
macgyver@louisiana.edu

MICHAIL LAGOUDAKIS
Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-0129
mgl@cs.duke.edu

Abstract: Computer users with motor impairments find it difficult and, in many cases, impossible to access PC functionality through the physical keyboard-and-mouse interface. Studies show that even able-bodied users experience similar difficulties when interacting with mobile devices; this is due to the reduced size/usability of the input interfaces. Advances in speech recognition have made it possible to design speech interfaces for alphanumeric data entry and indirect manipulation (cursor control). Although several related commercial applications exist, such systems do not provide a complete solution for arbitrary keyboard and mouse access, such as the access needed for, say, typing, compiling, and executing a C++ program.

We carried out a usability study to support the development of a speech user interface for arbitrary keyboard access and mouse control. The study showed that speech interaction with an ideal listening keyboard is better for users with motor impairments than handstick, in terms of task completion time (37% better), typing rate (74% better), and error rates (63% better). We believe that these results apply to both permanent and task-induced motor impairments. In particular, a follow-up experiment showed that handstick approximates conventional modes of alphanumeric input available on mobile devices (e.g., PDAs, cellular phones, and personal organizers). These modes of input include miniaturized keyboards, stylus "soft" keyboards, cellular phone numberpads, and handwriting recognition software. This result suggests that a listening keyboard would be an effective mode for alphanumeric input on future mobile devices.

This study contributed to the development of SUITEKeys – a speech user interface for arbitrary keyboard and mouse access available for MS platforms as freeware.

Keywords: Assistive technology, motor impairments, speech user interfaces, listening keyboard, usability evaluation.

[1] Partial results from this project have appeared in (Manaris and Harkreader, 1998; Manaris et al., 1999; Manaris et al., 2001).

1. Introduction

Computing devices have become an integral part of people's educational, working, and private lives. Consequently, it is essential to provide interfaces that exclude as few users as possible. In the context of permanent or temporary motor impairments,[2] numerous products are available that address this issue. These products enable users to augment or even replace the available input devices, thereby improving usability and accessibility. Other products, such as ergonomically designed chairs, wrist pads, and mechanisms to customize the position of the keyboard and mouse, aim to provide greater comfort. Finally, some customizations are standard features of the operating system. For instance, Microsoft Windows platforms provide *accessibility options* that allow for keyboard-layout customization, facilitate sequential input of concurrent keystroke sequences, such as `<ALT-CTRL-DEL>` (*StickyKeys*), and provide for control of the mouse through the keyboard (*MouseKeys*).

These products may be classified as follows:

1. Systems that augment motor skills within existing modes of input. Examples include various trackballs, touchpads, extended keyboards, split keyboards, chord keyboards, and other tactile input devices, such as the *Keybowl* (McAlindon and Staney, 1996).
2. Systems that replace existing modes of input with alternate ones. Examples include various on-screen keyboards, such as *WiVik*$_2$ (Prentke Romich, 1997).

Additionally, systems may be further subdivided with respect to whether they provide access to all computer functionality, or sacrifice completeness for usability. Examples of the latter include StickyKeys and commercial speech recognition systems.[3]

1.1 Speech versus Keyboard Input

The effectiveness of speech input as an alternative to the keyboard has been studied extensively. Murray et al. (1983) compared speech and keyboard for controlling cursor movement. They found that the keyboard was twice as fast and more preferred by users. Morrison et al. (1984) compared speech and keyboard for editing text that had been already typed through the keyboard. They discovered that typists preferred the keyboard, whereas non-typists initially preferred speech but eventually switched to the keyboard. In their experiment, because of speech-engine limitations, subjects could speak only commands and had to type any command parameters via the keyboard. Subjects did comment that switching mode of input in mid-task was disruptive and may have contributed to non-

[2] We use the term *temporary impairments* to refer to impairments which do not stem from internal physical disabilities, but are caused by external factors that are temporary in nature. For example, in hands-busy, eyes-busy environments, such as driving an automobile or servicing a jet engine, a task may require interaction with a computing device while prohibiting/inhibiting access to a physical keyboard or mouse.

[3] For instance, StickyKeys cannot handle all possible key combinations that could potentially be used by some interface, e.g., pressing the K, J, and H keys simultaneously.

typists eventually preferring the keyboard. Leggett and Williams (1984) compared speech and keyboard in a language-directed editing environment. Their subjects – 24 knowledgeable programmers – performed better with the keyboard in terms of completion rate, but worse in terms of error rate. Finally, Gould et al. (1983) studied whether keyboard input could be replaced completely by speech input. In this intriguing study, a "listening typewriter" was simulated using a Wizard-of-Oz setup. Specifically, subjects spoke into a microphone to edit a document. In an adjacent room, a skilled typist entered their input using a keyboard. Subjects "typed" more slowly through speech, however they liked the "listening typewriter" concept. It should be noted that these studies employed speech at the level of complete words (e.g., "copy") as opposed to individual spoken keystrokes (e.g., "c", "o", "p", "y"). Also, these studies assumed users without motor impairments.

The above studies suggest that speech input, in general, is slower than keyboard input. However, speech input does not require manual dexterity – only motor control of the vocal apparatus. This observation has led to the development of many assistive technology applications. Danis et al. (1994) developed a speech editor to assist newspaper reporters who suffer from repetitive stress injuries. Fell et al. (1999) used a speech recognizer to screen infants who may be at risk for later communication problems. Malkewitz (1998) developed a user interface that combines speech and head movements to replace the keyboard and mouse. Levitt (1994) provides a somewhat dated but interesting survey of related assistive technology applications. Other related speech applications are presented in Lazzaro (2001), Markowitz (1996), Mostow and Aist (1999), Raman (1996), Roy and Pentland (1998), and Schwartz (2000).

Speech input by dictating individual keystrokes is obviously much slower than dictating complete words. However, it allows entry of arbitrary alphanumeric data. It also provides motor-impaired users with full access to PC functionality – similar to the access available to able-bodied users through the physical keyboard. To perform arbitrary actions on a PC, users with motor impairments manipulate the physical keyboard through various low-tech alternatives, such as mouthstick and handstick. A listening keyboard would provide a complete assistive technology solution for arbitrary keyboard and mouse access, such as the access needed for typing, compiling, and executing a C++ program. [4]

Additionally, a listening keyboard could be used as both an augmentative and an alternative mode of input given the user's motor-control requirements. Obviously, speech input at the keystroke level is not very effective for users with full motor control, as shown in the above studies and verified in the study presented here. However, it is very effective for users with permanent and temporary motor impairments who resort to various alternative input devices.

[4] In a freshman-level "Intro to Programming" class, one of the authors observed a student with motor impairments attempting to compile a program using a handstick. It was clear that the student had an excellent conceptual model of the task, the capabilities of her computer, and the functionality of the user interface. However, she was extremely slow due to her mode of input. In a rare moment of inspiration, the observer realized that a listening keyboard might be a viable solution. This was the original motivation for this project.

Existing speech applications, such as Dragon Systems' *NaturallySpeaking*, do not provide access to all the functionality available through the physical keyboard and mouse; these applications are designed for higher-level tasks, such as dictation and command and control. Consequently, users cannot access through them all the functionality accessible to users of a physical keyboard and mouse. Although users can "program" existing speech applications to perform various operating system commands, this programming requires considerable premeditation and effort. Through a speech user interface, which supports full manipulation of a standard keyboard and mouse, users with motor impairments can perform any action able-bodied users can. This includes running a speech dictation application, such as NaturallySpeaking.

Additionally, speech input for alphanumeric data entry provides users with motor impairments access to any device that requires/supports alphanumeric data entry, such as palmtop PCs, cellular phones, and video camcorders. This also applies to users with task-induced motor impairments, such as users entering alphanumeric data on mobile devices. This is because, as the physical dimensions of the input device shrink, the motor skills of the user become increasingly ineffective. For instance, studies show that users may experience severe reduction in data entry speed (words per minute) when switching from a regular QWERTY keyboard to a mobile-device keyboard alternative. These studies report a 75% reduction on a stylus-based "soft" keyboard (PalmPilot) and a 60% reduction on a telephone keypad (Goldstein et al., 1998; MacKenzie at al., 1999). Given the latest advances in computing power of mobile devices, it will not be long before such a speech user interface could run on a variety of mobile computing platforms (Schwartz, 2000).

2. Usability of Speech for Alphanumeric Input

How effective would a listening keyboard be for users with motor impairments, compared to other available modes of input? To help answer this question we carried out a usability study. This study follows the iterative design methodology used by Kelley (1984) to develop user-friendly natural language applications. Central to Kelly's methodology is an experimental Wizard-of-Oz simulation of the system under development. Specifically, his methodology consists of six steps:

1) *Task analysis:* Subjects are interviewed extensively to understand the application domain.

2) *Application level development:* The application layer of the system is developed to provide all necessary functionality. This functionality is independent of the user interface.

3) *First Wizard-of-Oz session (simulation):* Subjects interact with a system simulated by a human. They believe they interact with a real system. The goal is to refine the task analysis and model the linguistic domain.

4) *Prototype development:* The corpus collected in step 3 is used to develop a functional prototype of the system.

5) *Second Wizard-of-Oz session (intervention):* Subjects interact with the implemented prototype in an attempt to evaluate it and refine it. If the prototype falls short of user expectations in terms of functionality or linguistic coverage, the wizard intervenes to simulate the missing functionality. This provides feedback for iterative refinement of the system.

6) *Summative evaluation:* The final system is evaluated to see how well it performs.

Our study was carried out in three separate phases. The first phase helped refine the experimental methodology; it also helped refine the functional requirements of SUITEKeys (task analysis). This phase combined steps 1 and 3 of Kelly's methodology. It was preceded and followed by the development of several SUITEKeys prototypes (Kelly's steps 2 and 4).

The second phase was a larger-scale version of the first phase. It focused on assessing the effectiveness of alphanumeric data entry via speech for users with motor impairments. The results from the second phase showed that speech input, using an ideal speech recognizer, is much better for alphanumeric data entry compared to handstick input.

The third phase compared handstick to other input devices used in mobile computing for alphanumeric data entry. It showed that the handstick is at least as effective as these devices. Since speech was shown to be much better than handstick, this implies that speech is much better than other input devices used in mobile computing for alphanumeric data entry.

Around the time of the experiment (summer of 1998), speech recognition technology had experienced several performance breakthroughs related to accuracy, speed, and vocabulary size. We were concerned that better speech engines might soon replace the state-of-the-art engines at our disposal. If we had used these speech engines in our study, the results would have not reflected the potential of speech as a mode of input for alphanumeric data entry – they would have only reflected the capabilities of the state-of-the-art in speech recognition and the potential limitations of our implementation. For this reason, similarly to Gould et al. (1983), in the second phase of the study we assumed that the simulated system represented an ideal implementation of the system under development.

Incidentally, we feel that some earlier studies assessing the effectiveness of speech input suffer from this flaw – they implicitly assume that the then state-of-the-art represents an upper bound of speech recognition performance. Or at least that's how many readers may have interpreted them. Given the developments in speech recognition within the last decade, one cannot help but wonder how the results of such experiments would turn out today. Calverley (1999) introduces the intriguing possibility that even our human-like speech recognizer may be a conservative upper bound of the effectiveness of speech for alphanumeric data entry, as computer speech recognition may eventually surpass human capabilities.

3. Pilot Study: Listening Keyboard as an Alternate Input Mode

The first experiment served as a pilot study to help refine functional requirements, as well as to assess the methodology for the main evaluation study. The specific design of this experiment was based on Napier et al. (1989), Smith et al. (1996), and Trewin (1996).

In this study, subjects with permanent motor disabilities were asked to perform an alphanumeric data-entry task on an MS Windows platform using two alternative modes of input: (a) their preferred mode of input, and (b) a simulated listening keyboard. A member of our team simulated the speech interface by listening and typing as the subjects uttered alphanumeric data. Given the pilot nature of the study, we did not hide the simulated nature of the experiment.

Subjects: Three volunteers served as subjects in the pilot study. All subjects had upper-body motor impairments that interfered with the use of a standard keyboard and mouse. All subjects were university students (two undergraduates, one graduate) and experienced computer users. Two of the subjects were female, one male. None of the subjects had significant speech impediments.

Task: A simple document creation task was introduced, namely to type a short paragraph into a text editor and save the document. Each subject performed the task using two different source paragraphs, each with a different method of input (the user's preferred method and the simulated listening keyboard). The subjects were instructed to enter text using the keys on the keyboard as opposed to dictating them to a human.

Measures: The dependent measures were *completion rate* (percentage of task completed), *typing rate*, and *error rate*. The independent variable had two values: subject's preferred mode of input and speech input.

- *Completion Rate*: This was calculated as the number of correct characters typed per number of characters in the target, or proportion correct.
- *Typing Rate*: This was calculated as the number of correct characters typed per task completion time (total time), and reported as key presses per second.
- *Error Rate*: This was calculated as the number of erroneous keystrokes per total number of keystrokes, or proportion of erroneous keystrokes.[5]

Apparatus: The computing environment used to type in the paragraph was Microsoft Windows NT Workstation 4.0 running on an 180MHz Pentium PC. The only additional input equipment was a mouthstick used by one of the subjects. The text editor used in this experiment was Microsoft's Notepad. Four methods were used to collect data during the pilot study. The computer's display was recorded by VCR (JVC BR-S822U) through a screen capture device (AITech Pro PC/TV). The room audio was also recorded by the VCR. User keystrokes and mouse button

[5] This attempts to capture how error prone each mode of input is, regardless of whether an error is generated by the subject or the system interpreting the subject's input.

clicks were recorded with a custom-made software logger (Winlogger). The logger recorded the time (to the nearest millisecond), the nature of the hardware event (i.e., key press, key release, mouse button press, or mouse button release), and the key or mouse button that generated the event. Finally, the saved document created by the user served as a source of measurable data.

Procedure: The subjects were tested independently. Each subject participated in a first trial involving their preferred means of input, a training session with the simulated system, and a second trial making use of this system.

Trial 1 (Preferred Input Method): The first trial consisted of entering a 38-word (238 characters, including punctuation and spacing) paragraph followed by saving the created document. Initially, two windows running the Notepad text editor were displayed on the computer screen, one window above the other. The top window fully displayed the target paragraph.[6] The bottom window displayed a blank document. The bottom window had focus.

The subjects employed their preferred input method (using the keyboard and mouse, without the assistance of speech system) to type the text displayed in the top window into the bottom window and to save the resulting document. The preferred methods of the three subjects were as follows: subject 1 typed single-handedly; subject 2 was unable to manipulate directly the keyboard or mouse – she verbally directed an assistant (not a member of the research team) to carry out the desired actions; and subject 3 employed a mouthstick to manipulate the keyboard, making use of the StickyKeys functionality available in Microsoft Windows.

Training: After Trial 1, the subjects were trained in using the simulated listening keyboard. This system performed actions such as keystrokes, moving the mouse, and clicking the mouse buttons based on verbal commands from the subject. It beeped or "froze-up" when the subject's commands were unintelligible or the subject spoke too rapidly. Training consisted of retyping the paragraph from Trial 1.

Trial 2 (Speech Input Method): The second trial consisted of entering a 41-word (257 characters, including punctuation and spacing) paragraph followed by saving the created document using the simulated system. Other conditions were identical to Trial 1.

3.1 Pilot Study Results

This preliminary study was conducted to study the effectiveness of speech as an alternate mode of input for users with motor impairments. It gave us a better understanding of the factors involved in assessing the effectiveness of various modes of computer input for users with motor impairments. It also helped us refine our experimental methodology for use in future, larger-scale experiments. For instance, it became clear that, for a larger-scale study,

[6] A potential drawback of this task is that it does not capture the difficulty of someone attempting to spell out from his or her head. This is significant when typing words, but not as significant when typing arbitrary alphanumeric strings – the latter are already kept as individual letters/numbers in someone's head. Another possibility is to ask subjects to type the words to a familiar song or a memorized passage. This, however, may overcompensate in terms of difficulty as it adds a considerable memorization/recall load – in addition to spelling out from his or her head, the subject has to recall the memorized words.

we would need to simulate a permanent motor impairment with able-bodied subjects. This is because we were able to locate only four people in our area with relevant motor impairments capable of participating in such a study.

This study also allowed us to observe how users chose to communicate with a keyboard that understands natural language. This directly affected the mindset used in developing SUITEKeys. In particular, all three subjects were comfortable with the concept of a listening keyboard. All three subjects converged to "typing" by spelling out words, such as "a", "n", "d"; they quickly stopped using prefixes, such as "press a", or suffixes, such as "a key". Also, given the functional simplicity of a keyboard, the input generated by the subjects was very simple grammatically, such as "backspace three times." This information was incorporated into the linguistic model of SUITEKeys. Additional information on how this study contributed to the design of SUITEKeys is available in Manaris and Harkreader (1998).

Table 1 summarizes the quantitative results of the pilot study. Subjects were able to achieve good performance through speech with minimal training (less than ten minutes). It should be noted that these subject had considerable training on their preferred input modality. Therefore, it is perhaps not surprising that (other than the first subject who was able to type with one hand) subjects achieved comparable results between speech and their preferred mode of input. We interpreted this as an indication that speech is an effective mode of input for users with permanent motor impairments. One would expect that this would be even more pronounced for motor-impaired users that have not yet established a preferred mode of input, as opposed to our three subjects. In particular, we felt that, considering the effort associated with rehabilitation and computer training for users with motor impairments, systems such as SUITEKeys could prove to be a valuable assistive technology. We decided to explore this further in a larger-scale study (see next section).

Finally, subject 1's scores using speech (trial 2) were considerably lower than those of the other subjects because, after an erroneous start where she tried to speak complete words to the simulated system, she started over using discrete (as opposed to continuous) speech. This made us realize that (a) our instructions may have been vague or incomplete, and/or (b) that the subject, under the stress of the experiment, might have forgotten our instructions. As a result, we eventually extended the design of SUITEKeys to include a set of training "cards" to teach system essentials to the user. These cards are displayed automatically for every new user. Each card corresponds to a

	Subject 1		**Subject 2**		**Subject 3**		**All**			
	Preferred	**Speech**	**Preferred**	**Speech**	**Preferred**	**Speech**	**Preferred**		**Speech**	
							mean	***SD***	***mean***	***SD***
TotalTime (secs)	98	332	317	293	205	212	206.67	109.51	279	61
CompletionRate	1.0	1.0	.996	1.0	.983	.977	.993	.01	.992	.01
TypingRate (chars/sec)	2.43	.77	.75	.88	1.14	1.18	1.44	.88	.95	.21
ErrorRate	.029	.054	.046	.038	.023	.018	.033	.01	.037	.02

Table 1. Selected results of pilot study.

training exercise on a subset of SUITEKeys functionality. Each training exercise is associated with a very small linguistic model. Training cards are designed so that a user may proceed to the next lesson only by mastering the concept being presented (e.g., controlling the mouse with speech) (Manaris et al., 2001).

4. Main Study: Listening Keyboard and Users with Motor Impairments

The second study focused on comparing the listening keyboard to the handstick. The handstick was chosen because it resembles several "standard" modes of input used by users with motor impairments, namely single-finger typing, pencil-assisted typing, and the mouthstick. Moreover, when used with able-bodied subjects, it effectively simulates a motor impairment. As shown in the follow-up study (Section 5), interestingly, the handstick also approximates a range of alternative input devices that

(a) have decreased physical input area (e.g., miniaturized keyboards),

(b) have increased visual scan time (e.g., stylus-type "soft" keyboards), and

(c) add a third dimension through time (e.g., handwriting recognition devices).

Our hypothesis was that speech would prove more effective than handstick.

As mentioned above, for this phase we chose to test against an ideal speech recognizer simulated by a human. If the hypothesis did not hold for an ideal speech recognizer, we felt that this mode of input was not worth pursuing. Alternatively, if the hypothesis held for an ideal speech recognizer, but did not hold for the then state-of-the-art speech recognizers, this mode of input could still be explored at a later time (as permitted by advances in speech recognition). This assumption – that the wizard represented an ideal listening keyboard – allowed us to collect specific performance data comparing speech to handstick.

Subjects: The subject pool consisted of 43 able-bodied undergraduate psychology students (14 male, 29 female, of variable age) who participated as part of their course requirements. Of these, 11 subjects classified themselves as novice computer users, 12 intermediate, and 20 expert.

Task: Subjects were asked to type and save a one-paragraph document. Two linguistically similar paragraphs were used as sources (one for each mode of input). A separate paragraph was used for training purposes. The paragraph used for the handstick input (session A) was:

```
The language PLANNER, referred to here, is an AI language whose principal feature
is that some of the operations necessary for problem reduction are built in--
namely, the recursive process of creating a tree of subgoals, subsubgoals, etc.
```

The paragraph used for the speech input (session B) was:

```
What this means is that such processes, instead of having to be spelled out time
and time again by the programmer, are automatically implied by so-called GOAL-
statements.  Someone who reads a SHRDLU program will see no explicit reference to
such operations.
```

Both paragraphs were taken from Hofstadter (1989). For the handstick condition, subjects where given an unsharpened pencil with eraser, and were asked to hold it with both hands (eraser side towards keyboard), keep hands locked and held under their chin, and depress keys using their neck and torso muscles as opposed to wrist movements (see Figure 1). Figure 2 shows the setup for the speech condition.

Measures: As in the pilot study, we measured *completion rate* (percentage of task completed), *typing rate*, and *error rate*. Additionally, we collected data on how the subjects reacted to an implemented version of SUITEKeys through a post-experiment survey (see Section 4.1).

Apparatus: Two platforms (equivalent to the one in the pilot study) were used in two separate rooms to facilitate two parallel sessions. In both sessions, the environmental conditions were identical in terms of climate control, seating, privacy, and background noise. For the speech condition, we used a Wizard-of-Oz setup – subjects assumed they interacted with an implemented prototype of SUITEKeys whereas they actually interacted with a system simulated by a human. The wizard used a third platform that was "hidden" in an adjacent room (see Figure 3). Subjects spoke into a microphone that was connected to a set of headphones worn by the wizard. He used a separate keyboard and mouse to simulate recognition of the subjects' input. The wizard was instructed to accept any reasonable input within the application domain – control of a listening keyboard and mouse – and reject any other input. To improve the accuracy of the simulation the wizard could not see the results of his actions – these were displayed only on the subjects' monitor. Thus the wizard could not anticipate user input, as it was extremely hard, if not impossible, to keep track of the state of the interaction.

For each subject we collected the following data: generated documents (handstick, speech), keystroke-event logs (handstick, speech), video capture of computer display without audio (handstick), video capture of computer display with room audio (speech), and survey data. In the speech condition, we captured the data generated by the wizard as he "blindly" carried out the subjects' speech commands. In addition to the data collection methods used in the pilot study, we conducted post-experiment surveys. Although we provided for mouse input, we did not study the effectiveness of a listening mouse.[7]

Procedure: The subjects were tested independently (in two parallel sessions). They were not aware of either the purpose for or the Wizard-of-Oz nature of the experiment. All subjects were tested under both conditions (each consisting of training and an actual run). Subjects were randomly assigned as to which condition they performed first. Detailed written protocols were used to standardize the presentation of instructions to the subjects. Following both tasks, subjects completed a brief questionnaire assessing their impressions of the two modes of input, as well as

[7] Pausch and Leatherby (1991) found that combining speech and mouse input decreases task completion time by 21% in certain graphical editing tasks. Karl et al. (1993) showed that using speech instead of mouse decreased task completion time by 18.67% in certain word processing tasks. On the other hand, Christian et al. (2000) found that using speech instead of mouse increased task completion time by 50% in certain web browsing tasks. These studies indicate that a listening mouse is an effective mode of input at least for some selection tasks.

whether or not they had suspected the Wizard-of-Oz nature of the experiment. Only 7 of the 43 reported a suspicion, and only 3 of those suspicions were based on rational evidence.

4.1 Main Study Results

The results of the main study indicate that speech is much better for alphanumeric data entry than handstick. Table 2 and Figure 4 present the group means and statistics. Specifically, subjects performed best in the speech condition in that they took less time, typed faster, were more complete, and made fewer errors. Note that while the means in *CompletionRate* are very similar, the variance is also very small, so those differences are significant.

A significant main effect of User Level (novice, intermediate, and expert computer users) was found for *TotalTime* and *TypingRate*. This indicates that novice users took longer to complete the task and were slower typists. Examination of means shows that this was truer in the handstick condition than in the speech condition.

It should be noted that one significant order effect was found. Participants made more errors in the speech condition when it was preceded by the handstick condition ($F(1,41) = 5.31$, $p = .03$), which we interpreted as a fatigue effect. This conclusion is supported by the questionnaire data.

The post-experiment questionnaire presented eight statements about the two systems (conditions). To each statement, participants indicated their level of agreement using a six-point scale, where "1" indicated strong

Fig. 1. Handstick input condition

Fig. 2. Speech input condition

Fig. 3. Wizard-of-Oz setup

disagreement and "6" strong agreement. The first item stated, "The voice controlled interface was easier for me to use than the handstick." The mean level of agreement, *M*, was 5.35, supporting the fatigue interpretation.

The results of the analyses of variance suggest that a listening keyboard would be easily learned by users, especially when compared to the labor-intensive alternative. Each of the four summative measures showed a significant main effect of condition favoring the speech condition. These effects are most pronounced for novices.

Participants also indicated that the speech condition worked well (*M* = 5.58) and that they would want to purchase a program like this (*M* = 4.38), especially if they had a physical disability (*M* = 5.67). Further, the participants felt they would prefer a speech-activated system to a handstick- or mouthstick-activated one if they had a disability (*M* = 5.51). Finally, users generally disagreed with statements suggesting that the speech-activated system was not user-friendly (*M* = 1.91), too confusing to work with (*M* = 1.72), and harder to learn than the handstick (*M* = 2.14).

Finally, we were able to further refine the SUITEKeys linguistic model by observing the interactions during the experiment. In some cases, we verified observed patterns by referring to captured data – generated documents, keystroke-event logs, and videotapes. We tried to balance the trade-off between (a) including all interactions observed during the experiment and (b) minimizing the complexity of the SUITEKeys grammar.[8]

Variable	User Level (n)	Means Handstick Condition	Means Speech Condition	ANOVA Main Effects	p(F)
TotalTime (seconds)	Novice (11)	359.1	207.6		
	Intermediate (12)	329.0	219.6	User Level F (2,40) = 5.89	.0057
	Expert (20)	295.5	187.3		
	All (43)	321.1	201.5	Condition F (1,40) = 110.42	.0001
CompletionRate (proportion of correct chars)	Novice (11)	.984	.996		
	Intermediate (12)	.994	.999	User Level F (2,40) = 2.15	.13, ns
	Expert (20)	.992	.998		
	All (43)	.991	.998	Condition F (1,40) = 6.90	.0122
TypingRate (chars per second)	Novice (11)	.682	1.290		
	Intermediate (12)	.749	1.203	User Level F (2,40) = 4.85	.0131
	Expert (20)	.821	1.440		
	All (43)	.766	1.336	Condition F (1,40) = 127.82	.0001
ErrorRate (proportion of erroneous keystrokes)	Novice (11)	.083	.034		
	Intermediate (12)	.104	.021	User Level F (2,40) = 0.05	.949, ns
	Expert (20)	.086	.040		
	All (43)	.090	.033	Condition F (1,40) = 23.8	.0001

Table 2. Selected results of main study.

[8] A thorough description of the refined linguistic model is available in the Quick Reference Guide at www.suitekeys.org .

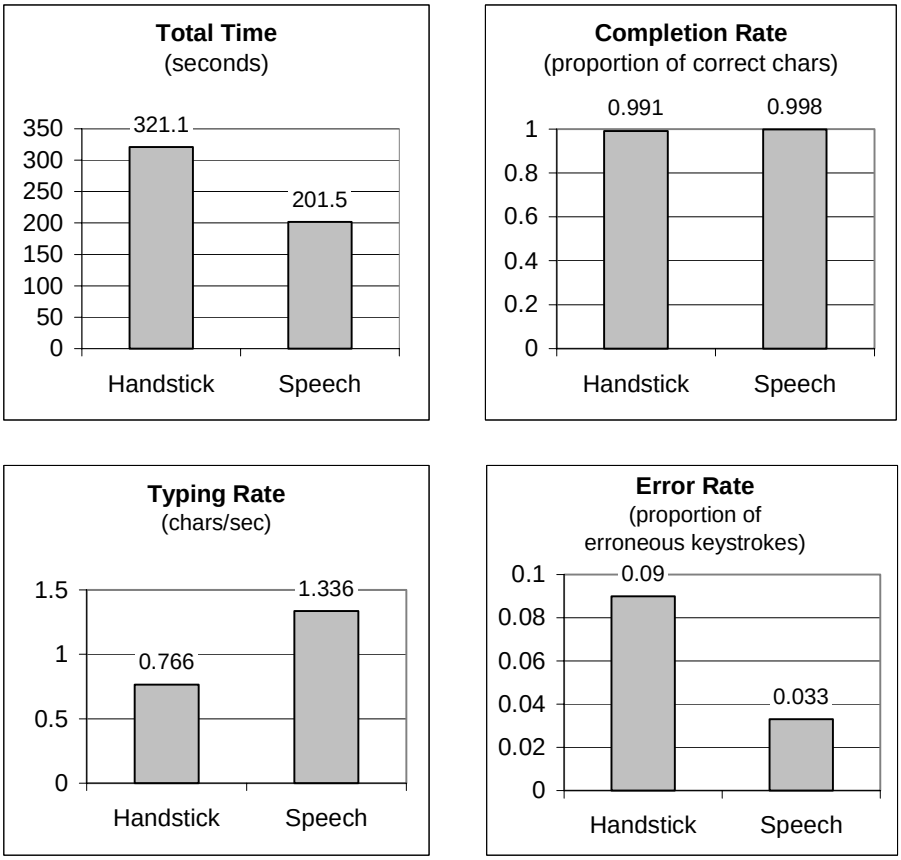

Fig. 4. Mean values for summative variables (main study).

5. Follow-up Study: Listening Keyboard and Mobile Devices

The goal of the follow-up study was two-fold. First, we wanted to show that the handstick approximates the modes of input for alphanumeric data entry on mobile devices (e.g., PDAs, cellular phones, and personal organizers). Then, given the main study's results, we wanted to show by inference that speech is superior to user interfaces available on such mobile devices.

Specifically, having already established the usefulness of a listening keyboard for users with permanent motor impairments, the goal of the third study was to assess the effectiveness of this mode of input for temporary (task-induced) motor disabilities. Our hypothesis was that the handstick condition is a good approximation of mobile input devices characterized by decreased physical input area, increased visual scan time, and/or increased character specification time. Examples of such devices include miniaturized keyboards, stylus "soft" keyboards, and handwriting recognition software. In this experiment, we compared the handstick to the following input devices: standard QWERTY keyboard (*QWERTY*), miniaturized QWERTY keyboard (*minikey*), stylus "soft" keyboard

(*stylus*), T9-like telephone keypad (*T9*), and a handwriting recognition system (*handwrite*).[9] A simple repeated-measures design was used; subjects would enter a short string using six different data entry methods.

Subjects: A convenience sample of 14 adults was used. Eleven of the subjects were undergraduate or graduate students in computer science and psychology. Three subjects were faculty/staff members in Computer Science. All subjects had computer expertise and were comfortable QWERTY typists. One subject was an expert with the handwriting recognition program. Other subjects had little or no familiarity with the input devices used (other than QWERTY). All were volunteers.

Task: Subjects were asked to type the following string, without capitals or punctuation:

```
the quick brown fox jumps over the lazy dog
```

This string was chosen since it contains all the alphabet keys. Without errors, it takes 43 keystrokes to type it.

Measures: For each condition, we recorded five pieces of data. Task completion time (measured with a digital stopwatch), number of times the "backspace key" was pressed, number of transposed characters (no subject transposed characters), number of missing characters, and the number of wrong characters.

Four summative variables were computed. The *TotalTime* to enter the 43 keystrokes was recorded in seconds. The *TypingRate* was computed as (43 – *MissingCharacters*) / *TotalTime*, giving characters per second. *CompletionRate* was computed as (43 – (*MissingCharacters* + *WrongCharacters*)) / 43. *ErrorRate* was computed as (*MissingCharacters* + *WrongCharacters* + *Backspaces*) / 43.[10]

In addition, subjects completed a short questionnaire focusing on ranking the six data entry conditions in terms of ease of use and ease of learning. They were also asked to rate the four input devices (minikey, T9, stylus, handwrite) in terms of how well they worked, how user-friendly the corresponding user interfaces were, and whether they would want to purchase a system incorporating such an input device.

Apparatus: A Windows NT workstation with a QWERTY keyboard running Notepad was used for two of the conditions, QWERTY (regular typing) and handstick. As mentioned in Section 4, in the handstick condition, subjects held an unsharpened pencil in both hands, with the back of their fingers pressed under their chin and their wrists held rigid; the subjects had to utilize neck and torso muscles to press the keys with the pencil.

A small personal organizer (Sharp EL-6620 Electronic Organizer) was used for the minikey condition. A 3Com PalmPilot was used for two of the conditions: stylus (the subjects used a pencil-like stylus to press keys on a touch-

[9] The T9 software, developed by Tegic Communications, Corp., optimizes text entry via telephone keypad by applying text statistics to minimize the number of keystrokes required to enter a specific letter (Comerford, 1998).

[10] This formula captures accuracy in process – how far the total keystrokes are from the optimal 43 keystrokes. For example, an error rate of 50% means that, given a sentence with optimal length 30, the subject will make about 15 mistakes. An error rate of 300% simply means that the subject will make about 90 mistakes.

sensitive screen), and handwrite (subjects utilized the PalmPilot's Graffiti handwriting recognition program). In addition to the PalmPilot, subjects were also provided with a 'cheat-sheet' detailing how to input characters into the Graffiti program.[11] Finally, a Nokia cellular phone was used for the T9 condition. Specifically, the T9 interface was simulated by subjects pressing the key corresponding to each letter only once.[12] The experimenter recorded these key presses as numbers. The # key was used as a space key. The phone had a 'clear' key that acted as a backspace key. The recorded number sequence was compared after the experiment to the optimal number sequence (the smallest possible sequence of keystrokes without errors).

Procedure: All subjects were informed of the purpose of the study and volunteered to participate. Each subject entered the character string using each of the six data entry devices in a fixed order: QWERTY, minikey, stylus, T9, handstick, handwrite. This order placed modes of input in an increasing order of presumed "difficulty" in an attempt to counteract potential effects of training bias (since a single character string was being used). Before each condition, the data entry process was explained to the subjects, and they were allowed to study the device before they began. A sheet of paper showing the target input string was available at all times. Subjects were instructed to situate themselves comfortably. The experimenters arranged themselves so as to see the keyboards. For the smaller devices (minikey, stylus, handwrite and T9), subjects were asked to say "Oops" if they needed to press the "backspace key". This helped verify the experimenters' visual observations, which were constrained because of the small input device area. We opted against videotaping, since it has been reported as not very effective in a similar experiment using a PalmPilot (Goldstein et al., 1998). Experimenters also recorded all the key presses in the T9 emulation, since the phone was not able to maintain all 43 characters in its buffer. After subjects finished all six conditions, they completed the questionnaire.

5.1 Follow-up Study Results

The study results show that handstick is a good approximator to many alphanumeric modes of input available in today's mobile devices. Given the nature of the hypothesis, we used simple Student's *t*-tests to analyze the data. Table 3 and Figure 5 present the means for each condition, for each of the four summative variables. In comparing the means, as was expected, touch typing (QWERTY) was faster than *handstick*, as measured by *TotalTime*, $t(13) = -7.63$, $p < .0001$; and by *TypingRate*, $t(13) = 9.77$, $p < .0001$. However, touch typing was not more accurate, as indicated by the *CompletionRate*, $t(13) = .26$, *ns*, and the *ErrorRate*, $t(13) = 1.34$, *ns*.

The minikey means suggest that the subjects' performance is better using the miniaturized keyboard than the handstick. The miniaturized keyboard afforded higher typing speed, as indicated by *TotalTime*,

[11] Graffiti input resembles regular handwriting in that users use a stylus to "draw" the outline of each character they wish to input. However, Graffiti character outlines are, in many cases, different from the ones used in regular handwriting. This means users have to learn a new way to "write" most characters, hence the provided 'cheat-sheet'.

[12] To enter a single character on a phone without the T9 software subjects may have to press a key up to three times. Keystrokes cycle through the letters associated with a given key (e.g., pressing #2 cycles through 'A', 'B', and 'C').

t(13) = 3.88, $p < .002$, and by *TypingRate*, *t(13)* = –4.64, $p < .001$. However, the miniaturized keyboard was not more accurate than the handstick, as indicated by the *CompletionRate*, *t(13)* = –0.52, *ns*, and by the *ErrorRate*, *t(13)* = .32, *ns*.

The handstick was not significantly different from the stylus "soft" keyboard (stylus) on any of the measures, namely *TotalTime*, *t(13)* = –.64, *ns*, *TypingRate*, *t(13)* = .81, *ns*, *CompletionRate*, *t(13)* = –.94, *ns*, and *ErrorRate*, *t(13)* = –.86, *ns*. This indicates that the handstick is a good approximation for this device. The handstick afforded higher typing speed than the T9 emulation, as indicated by *TotalTime*, *t(13)* = –3.37, $p < .005$, and *TypingRate*, *t(13)* = 4.31, $p < .001$. The handstick also afforded higher typing speed and was more accurate than handwriting recognition (handwrite), as indicated by *TotalTime*, *t(13)* = –5.98, $p < .0001$, *TypingRate*, *t(13)* = 3.16, $p < .01$, *CompletionRate*, *t(13)* = 1.96, $p < .05$; and *ErrorRate*, *t(13)* = –4.49, $p < .001$.

Combined Minikey, Stylus, T9 and Handwrite. Another way to show that handstick is a good approximator is to combine the four non-QWERTY conditions and to compare the combined data to the handstick condition. The combined variable is called *MSTH* (Minikey, Stylus, T9 and Handwrite). Table 3 displays the means for the two conditions. From this perspective, handstick is actually more effective than MSTH in that it is faster and more accurate, as indicated by TotalTime *t(13)* = –3.59, $p < .001$ and *ErrorRate*, *t(13)* = –2.43, $p < .03$.

Combined Minikey, Stylus, and T9. All subjects, except one, found the handwriting recognition condition extremely difficult because they were untrained in the specialized user interface.[13] The one subject who was already familiar with that interface exhibited data entry speeds approximating the minikey performance. Assuming that once subjects become expert with the specialized interface they would all approximate minikey, we decided to do another analysis comparing handstick to the combined data from minikey, stylus and T9 conditions only. This was done in an attempt to get a more accurate estimate of the effectiveness of the handstick as an approximation to these input devices. This new variable is called *MST* (Minikey, Stylus, and T9). As shown in Table 3, the means for the two conditions are nearly identical. Specifically, results show that there are no significant differences between handstick and MST on

Variable	Means QWERTY Condition	Means Handstick Condition	Means Minikey Condition	Means Stylus Condition	Means T9 Condition	Means Handwrite Condition	Means MSTH Condition	Means MST Condition
TotalTime	13.50 (4.45)	46.86 (14.61)	26.44 (10.13)	50.20 (10.59)	73.57 (23.45)	133.84 (54.23)	71.01 (19.47)	50.07 (11.34)
CompletionRate	.99 (.01)	.99 (.04)	.99 (.01)	1.0 (.01)	1.0 (.01)	.95 (.06)	.98 (.01)	1.0 (.01)
TypingRate	3.43 (.87)	.97 (.22)	1.81 (.62)	.90 (.22)	.64 (.19)	.49 (.61)	.96 (.33)	1.12 (.31)
ErrorRate	.038 (.047)	.015 (.037)	.012 (.018)	.030 (.045)	.007 (.011)	.145 (.104)	.05 (.034)	.016 (.018)

Table 3. Means for each condition of follow-up study (standard deviations appear in parentheses).

[13] The Graffiti handwriting recognition program requires users to learn a new way to write letters.

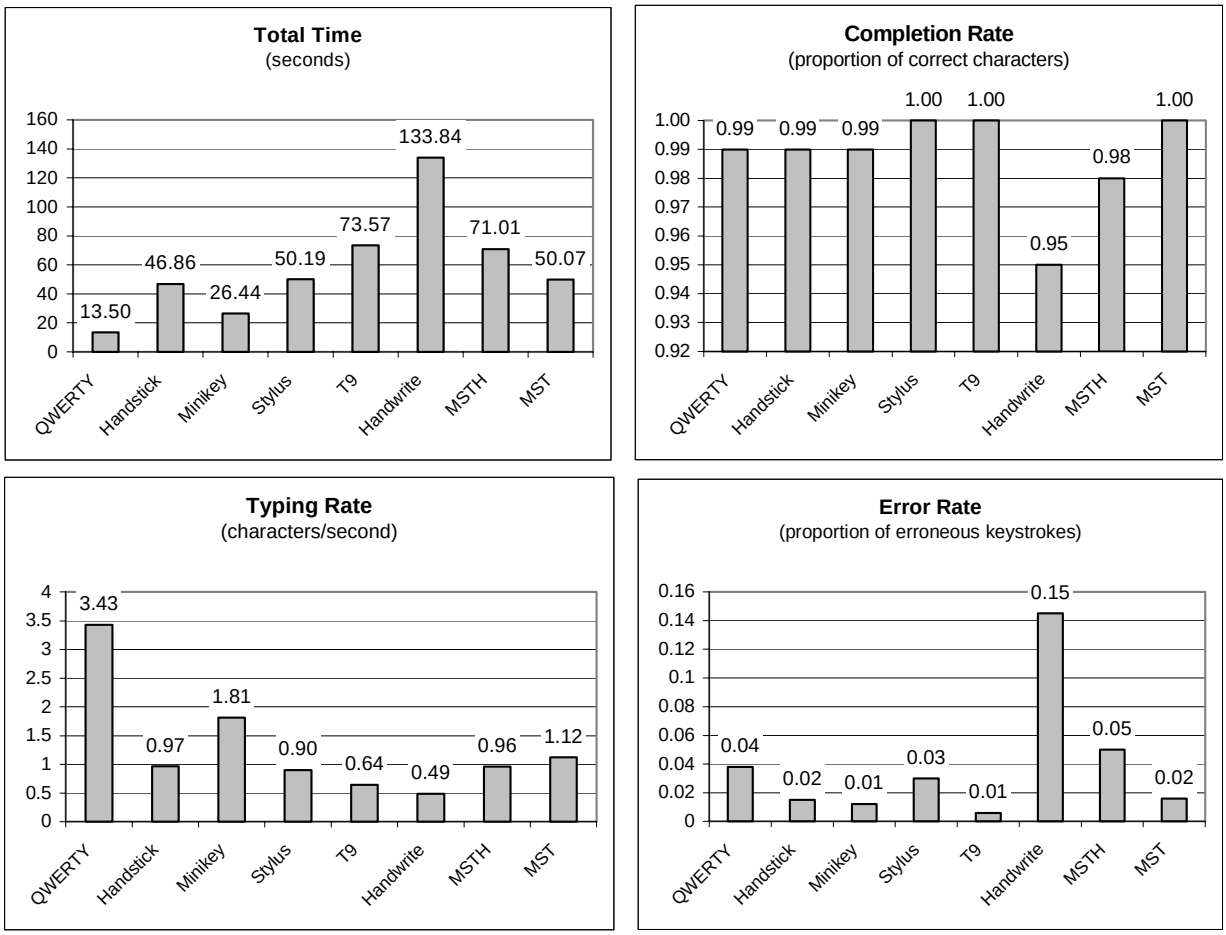

Fig. 5. Mean values for summative variables (follow-up study).

any of the summative measures, as indicated by *TotalTime*, $t(13) = -0.58$, *ns*, *TypingRate*, $t(13) = -1.43$, *ns*, *CompletionRate*, $t(13) = .07$, *ns*, and *ErrorRate*, $t(13) = -0.09$, *ns*.

Questionnaire Data. The questionnaire data provide additional support to these findings. In general, we see that the handstick condition is ranked in a manner similar to the other conditions, particularly the minikey, stylus and T9 (see Figure 6). The handwriting recognition interface was considered difficult to learn or use by the majority of the subjects. It is interesting to compare the subjective ranking, with respect to usability, with the objective ranking indicated in Table 3. If we accept *TypingRate* as indicative of usability, handstick is perceived to be far less usable than it actually is. This might be related to the fatigue effect discussed in Section 4.1.

5.2 Discussion

Overall, these results support the assumption that standard, non-speech modes of input challenge the user's motor-control skills because of the reduced physical area available for data entry. They also strongly indicate that speech is in general a much more effective mode of input for alphanumeric data entry compared to other alternatives.

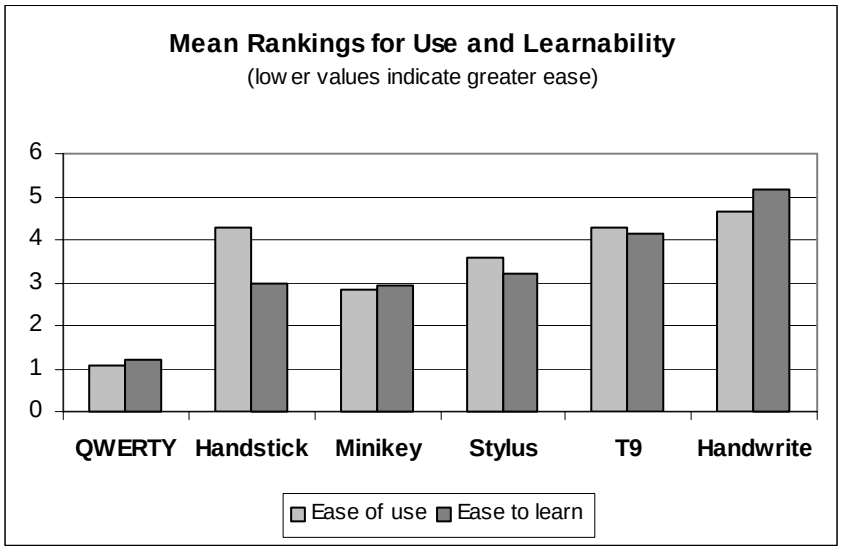

Fig. 6. Mean rankings for usability and perceived learnability.

Specifically, the results of the follow-up study provide compelling evidence that the handstick condition is a good approximation of many of the input user interfaces available on mobile computing devices. What is central here is how this interpretation relates to the results of the main study. Specifically, the main study indicated that a listening keyboard is far more effective than handstick input. Since handstick input is at least as effective as most of the examined alternative input devices, by transitive reasoning, speech input is also far more effective than the following alphanumeric input devices: stylus "soft" keyboards, handwriting recognition systems, and telephone keypads, at least for the novice user.

Moreover, we suspect that these results apply to any alphanumeric input device that challenges the user's motor skills through decreased physical input area, increased visual scan time, and/or increased character specification time.

6. SUITEKeys Speech User Interface

This section describes an implementation of a listening keyboard, called SUITEKeys. Version 1.0 has evolved through several prototypes. It is available as freeware for MS Windows at www.suitekeys.org. The study presented here helped refine its functional and linguistic requirements in various ways. For instance, it validated the need for its development – a listening keyboard was shown to be an effective mode of input for users with motor impairments. It also helped refine its linguistic model through identifying interaction patterns used by subjects that were not modeled in earlier prototypes.[14]

[14] Theoretical foundations and prototype development have been presented elsewhere (Manaris and Dominick, 1993; Manaris and Harkreader, 1997; Manaris and Harkreader, 1998; Manaris et al., 1999).

SUITEKeys allows users to transcribe sequences of keystrokes and mouse actions using spoken language. It assumes that the user speaks English and has minimal or no speech impediments. It models an 84-key keyboard (no numeric pad) and two-button mouse functionality. Other applications, including the operating system, treat the generated keyboard and mouse events as having originated from the corresponding physical devices.

Other speech applications, such as NaturallySpeaking and Microsoft Voice, do not provide access to all the functionality available through the physical keyboard and mouse. This is because they are designed for higher-level tasks, such as dictation, and command and control. They allow input of keystrokes either by requiring a fixed keyword prefix (e.g., "Press a"), or by entering a special spell mode. Moreover, they do not model all 84 keys. Finally, they send recognized keystrokes directly to the active window, bypassing the operating system. This prohibits the operating system and other applications from attaching arbitrary semantics to sequences of keystrokes (e.g., ALT-CONTROL-DEL), and intercepting them.

SUITEKeys 1.0 has been implemented using MS Visual Studio, SAPI, and `lex` and `yacc`. Minimum system requirements include: Pentium 120 processor (preferred Pentium 200 and up); 16MB RAM for Windows 95 & 98, 24MB RAM for Windows NT, and 64MB RAM for Windows 2000; 18MB disk space; sound card and sound card device driver supported in Windows; microphone (preferred noise-reducing, close-talk headset), and a 16 kHz/16 bit or 8kHz/16 bit sampling rate for input stream.

6.1 Domain Limitations

Although the system incorporates a continuous, speaker-independent engine, this capability could not be exploited in all cases. This is due to speech ambiguities in the linguistic domain which sometimes confuse even human listeners. For instance, the linguistic domain includes several near-homophone letters, such as "b" and "p", or "d" and "t". [15] For this reason, a few tasks have been modeled only through discrete speech.

Specifically, we use discrete speech mainly when transcribing regular letters (e.g., "a", "b"). We also use it in a few other cases – where our usability study showed that users preferred it, such as entering function keys (e.g., "Function Eleven", "Function Twelve") and certain special keys (e.g., "Insert", "Home", "Print Screen"). We use continuous speech in all other cases, such as entering letters using the military alphabet, numbers, repeatable keys (e.g., "Up Arrow", "Tab", "Back Space"), mouse commands, and system commands.

6.2 System Architecture

SUITEKeys incorporates a general-purpose speech processing architecture developed in the SUITE project at the University of Louisiana at Lafayette (Manaris and Harkreader, 1997). This architecture of SUITEKeys integrates speech recognition and natural language processing components. It processes input that originates as speech events and converts it to operating system keyboard and mouse events. The complete system runs on top of the operating

[15] Hence the invention of the military alphabet.

system like any other application. Other applications are unaware that keyboard/mouse events originated as speech input.

The architecture is subdivided into (a) the language model and (b) language processing components. Speech events are processed in a pipelined fashion to enable real-time response. Processing is performed by the following components:

- *Dialog Management:* SUITEKeys incorporates a stack-based dialog manager. This subsystem utilizes a dialog grammar containing dialog states and actions. Each top-level state corresponds to a specific language model (e.g., lexicon, grammar). The overall linguistic domain has been divided into thirteen such states. The dialog manager loads, in real time, the appropriate dialog state based on the current state of the interaction. Given the ambiguity of the linguistic domain, this allows processing components to focus on specific subsets of the overall domain, and thus improve recognition accuracy and performance.

- *Speech Processing:* SUITEKeys evolved through several architectures following the rapid developments in speech recognition within the last six years. Currently, it implements the Microsoft Speech API (SAPI). The released version is distributed with the MS Speech Recognition engine (freeware), but could also work with other SAPI-compliant engines, such as Dragon NaturallySpeaking.

- *Natural Language Processing:* The architecture incorporates a left-to-right, top-down, non-deterministic parser. This parser supports multiple parse trees, thus providing for ambiguity handling at the semantic level. It incorporates semantic actions for constructing semantic interpretations of user input.

- *Code Generator:* This module converts semantic interpretations – the output of the parser – to low-level code understood by the operating system.

- *Other Components:* Other components of the SUITE architecture include a knowledge-base manager, speech generator, lexical analyzer, and pragmatic analyzer. However, our SUITEKeys linguistic model did not have significant lexical, pragmatic, and speech generation aspects.

6.3 Graphical User Interface

Although SUITEKeys is primarily a speech user interface, it includes a graphical user interface for visibility and feedback purposes. Specifically, the user interface identifies available functionality and how to access it (visibility); it also provides information about the effects of user actions (feedback).

The *SUITEKeys* graphical user interface has two main views: maximized and minimized.

The maximized view is designed to provide maximum feedback to the user. As shown in Figure 7, this view allows the user to focus on SUITEKeys and its operation. This view provides several forms of feedback (left-to-right, top-to-bottom):

- **Command History:** a list of recognized commands in reverse chronological order. Unrecognized commands are denoted by "???".

- **Speech Processing:** superimposed animations provide feedback for two independent yet related events: voice input level corresponds to raising and dropping rings in green, yellow, and red; speech engine processing corresponds to a revolving “SK”.
- **Toggle Keys:** on/off indicators for CAPS LOCK and SCROLL LOCK.
- **Generated Keystrokes:** a keyboard that visually “echoes” generated keystrokes.

Additionally, this view identifies all system commands through its menu structure. Since some system commands disconnect the speech engine from the microphone (e.g., Microphone Calibration), such commands are not accessible through speech. This is to prevent users from entering system states from which they may not be able to exit.[16]

However, effective user interfaces should be invisible to the user. According to Weiser (1994),

> *[a] good tool is an invisible tool. By invisible, I mean that the tool does not intrude on your consciousness; you focus on the task, not the tool. Eyeglasses are a good tool — you look at the world, not the eyeglasses.*

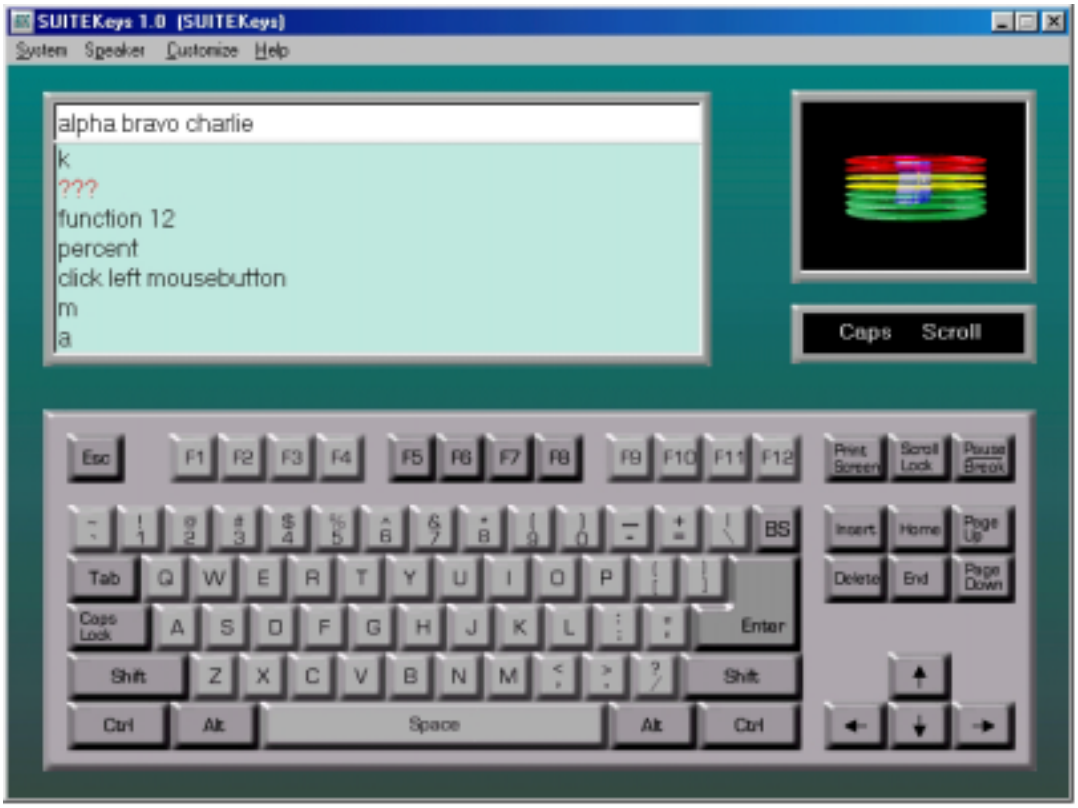

Fig. 7. SUITEKeys 1.0 maximized view.

[16] These commands are only accessible to users with some motor control via the physical keyboard/mouse interface.

For this reason, SUITEKeys provides a minimized view. This view is intended for when the user wants to concentrate on the task at hand, i.e., using the listening keyboard to control the PC (see Figure 8). This view always stays on top and provides minimal feedback, namely the last-recognized input, which modifier/toggle keys are pressed, the volume indicator, and the speech engine processing status.

SUITEKeys follows the "metaphor as model" approach (Preece et al., 1994). It is a listening keyboard. Since users presumably have a good conceptual model of the physical keyboard's functionality, they can easily learn/discover the listening keyboard's linguistic domain, e.g., keys may be pressed, released, held, etc. Given that the linguistic primitives of speech user interfaces are invisible (as opposed to the primitives of well-designed graphical user interfaces), this may significantly improve the system's learnability and usability.

Additional functionality—that is functionality beyond the interface metaphor—is incorporated by making it visible through the *SUITEKeys* graphical user interface. The user may simply read aloud menu entries and dialog-box elements to select them. Further support for learning and retaining how to use the system is provided through context-sensitive help (i.e., "What Can I Say") and progressive disclosure (i.e., "More Help").

This relatively low-level approach has a potential drawback compared to speech user interfaces employing a higher-level approach, such as NaturallySpeaking. In many cases, a user may be required to "synthesize" higher-level actions (e.g., Drag and Drop) through a sequence of low-level actions. This is especially "painful" in the context of dictation. Although SUITEKeys does not claim to be a dictation package, it addresses this drawback through its capability to relinquish control to other speech-enabled applications.

6.4 Usability of SUITEKeys

How does SUITEKeys 1.0 relate in terms of usability to the ideal system tested in our study? We are currently planning a large-scale survey of actual users of the system – there have been more than 300 registered downloads of SUITEKeys. So far we have conducted one small-scale experiment to answer the above question. For this experiment we tested one motivated, expert user (one of the authors) using the speech condition setup from the main study. This allowed us to approximate an upper bound of performance for the current implementation. The results indicate that, even in a best-case scenario, SUITEKeys 1.0 is far from ideal (see Table 4).

Fig. 8: SUITEKeys minimized view.

6.5 Discussion

Although far from ideal, the current implementation of SUITEKeys may still be a useful application for users with motor impairments. Although it is not as effective as handstick (at least in terms of data entry speed and task completion time), we feel it is a workable alternative at least for users with severe motor impairments.

As mentioned above, although the system incorporates a continuous speech engine, this capability was not exploited in all cases. Discrete speech slows down data entry considerably. We feel that its less-than-ideal performance is related to limitations in (a) our particular implementation, and (b) the current state-of-the-art in speech recognition. We have several ideas on how to improve performance, including a T9-like disambiguation algorithm for keystroke recognition that learns through user error recovery.

The modular design of SUITEKeys makes it straightforward to attach newer speech engines, as they become available, thus improving its performance.

There are two obvious ways to interpret these results:

a) Using an ideal human-like simulation to assess the effectiveness of speech in a particular application domain may be utopian, given the technological limitations of speech recognition. The results of such studies may mislead developers into wasting time and resources on unattainable goals.

b) Using an implemented system to assess the effectiveness of speech in a particular application domain is methodologically flawed. The results of such studies mainly reflect the limitations of the then state-of-the-art – not limitations of speech as a mode of input.

As mentioned above, we are most comfortable with the second interpretation. We also believe that several widely referenced studies comparing speech against other modes of input should perhaps be revisited. This is because their use of the then state-of-the-art in speech recognition was, in some cases, presented/misunderstood as reflecting limitations of speech as a mode of input. Unfortunately, this interpretation may have inhibited research and development along potentially promising directions.

Variable	Ideal Speech (Main Study)	Handstick (Main Study)	Speech (SUITEKeys)
TotalTime (secs)	201.5	321.1	346.8
CompletionRate (%)	.998	.991	1.0
TypingRate (chars/sec)	1.336	.766	.695
ErrorRate (%)	.033	.090	.057

Table 4. Selected results from main study (ideal speech, handstick), and preliminary results of SUITEKeys 1.0 effectiveness.

7. Conclusion

This paper presented a usability study evaluating the effectiveness of speech input for alphanumeric entry against alternative input devices. This study contributed to the design of a listening keyboard and mouse, called SUITEKeys. This speech user interface provides access to all available functionality of a PC by modeling interaction at the level of a physical keyboard and mouse. Although this work was originally targeted towards computer users with permanent motor impairments, it may also benefit able-bodied users with temporary (task-induced) motor impairments, such as users performing alphanumeric data entry through a cellular phone keypad.

The study reported here indicates that speech interaction with a listening keyboard is a very effective means of input in terms of user data entry, task completion, and error rates. Moreover, it suggests that this input method is far better than alternative user interfaces used in mobile devices that require physical manipulation of a device component for alphanumeric data entry. Such interfaces are characterized by decreased physical input area, increased visual scan time, and/or increased character specification time (e.g., handwriting recognition). A listening keyboard would be relatively easy to learn and to use, particularly for people with motor impairments and/or computer illiterate users. Anecdotal evidence from the novice subjects of the study suggests that this user interface is far less intimidating than others.

Although speech is not the best user interface for all human-computer interaction tasks, when delivered at the level of keyboard and mouse, it allows for universal access to computing devices by people with motor impairments. This access is functionally similar to the one available through a standard QWERTY keyboard and mouse, although there remain a few tasks, such as ‘freehand’ drawing with the mouse that are still difficult to perform. The listening keyboard concept studied here has the potential to complement or even replace existing user interfaces for mobile computing. Since it does not require much physical device area for alphanumeric data entry (only a microphone and, perhaps, a speaker for feedback), the physical device may shrink as much as advances in microelectronics will allow. We believe this result is of importance. It’s only a matter of time (perhaps a few years) before new delivery platforms for computing applications may be successfully exploited, such as eyeglass frames, watches, and perhaps even body implants (e.g., tooth crowns). Although the latter raises significant ethical issues, it will also provide for innovative solutions to a variety of problems faced by users with or without motor impairments.

Acknowledgements

This research was partially supported by the Louisiana Board of Regents (grant # BoRSF-(1997-00)-RD-A-31). The authors acknowledge the contribution of Renée McCauley, Rao Adavikolanu, Huong Do, Corey Gaudin, Garrick Hall, Alan Harkreader, William Jacobs, Jonathan Laughery, Eric Li, Adi Sakala, and the Spring 1999 Human-Computer Interaction class at the University of Louisiana at Lafayette. The authors would like to thank Najwa Dibbs, Jay Jackson, Ursula Jackson, Page Salley, Nona Wilson, and the subjects for their contribution to the usability study.

References

Calverley, B. (1999). Machine demonstrates superhuman speech recognition abilities. USC News Service, news release 0999025. http://uscnews.usc.edu/newsreleases .

Christian, K., Kules, B., Shneiderman, B., and Youssef, A. (2000). A comparison of voice-controlled and mouse-controlled web browsing. Proceedings of The Fourth International ACM Conference on Assistive Technologies (ASSETS 2000) (pp. 72-79). New York: ACM Press.

Comerford, R. (1998). Pocket computers ignite OS battle. *IEEE Spectrum*, *35*, 5:43-48.

Danis, C., Comerford, L., Janke, E., Davies, K., DeVries, J., and Bertrand, A. (1994). StoryWriter: A speech-oriented editor. Proceedings of Human Factors in Computing Systems (CHI ’94) – Conference Companion (pp. 277-278). New York: ACM Press.

Fell, H.J., MacAuslan, J., Ferrier, L.J., and Chenausky, K. (1999). Automatic babble recognition for early detection of speech related disorders. *Behaviour & Information Technology*, *18*, 1:56-63.

Goldstein, M., Book, R., Alsio, and G., Tessa, S. (1998). Ubiquitous input for wearable computing: QWERTY keyboard without a board. Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices, Glasgow, Scotland. www.dcs.gla.ac.uk/~johnson/papers/mobile/HCIMD1.html .

Gould, J. D., Conti, J., and Hovanyecz, T. (1983). Composing letters with a simulated listening typewriter. *Communications of the ACM*, *26*, 4:295-308.

Hofstadter, D. R. (1989). *Gödel, Escher, Bach: An Eternal Golden Braid*. New York: Vintage Books.

Karl, L., Pettey, M., and Shneiderman, B. (1993). Speech-activated versus mouse-activated commands for word processing applications. *International Journal of Man-Machine Studies*, *39*, 4:667-687.

Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Office Information Systems*, *2*, 1:26-41.

Lazzaro, J. J. (2001). Speech-enabling applications. *Byte Magazine*, April 4, 2001, www.byte.com/column/BYT20010404S0005 .

Leggett, J. and Williams, G. (1984). An empirical investigation of voice as an input modality for computer programming. *International Journal of Man-Machine Studies*, *21*, 1:493-520.

Levitt, H. (1994). Speech processing for physical and sensory disabilities. In D. B. Roe, and J. G. Wilpon (Eds.), *Voice Communication Between Humans and Machines* (pp. 311-343). Washington, DC: National Academy of Sciences.

MacKenzie, I. S., Zhang, S. X., and Soukoreff, R. W. (1999). Text entry using soft keyboards. *Behaviour & Information Technology*, *18*, 4:235-244.

Malkewitz, R. (1998). Head pointing and speech control as a hands-free interface to desktop computing. Proceedings of The Third International ACM Conference on Assistive Technologies (ASSETS ’98) (pp. 182-188). New York: ACM Press.

Manaris, B. and Dominick, W. D. (1993). NALIGE: A user interface management system for the development of natural language interfaces. *International Journal of Man-Machine Studies, 38,* 6:891-921.

Manaris, B. and Harkreader, A. (1997). SUITE: Speech understanding interface tools and environments. Proceedings of 10th International Florida AI Research Symposium (FLAIRS-97) (pp. 247-252). Menlo Park, CA: AAAI Press.

Manaris, B. and Harkreader, A. (1998). SUITEKeys: A speech understanding interface for the motor-control challenged. Proceedings of The Third International ACM Conference on Assistive Technologies (ASSETS '98) (pp. 108-115). New York: ACM Press.

Manaris, B., MacGyvers, V., and Lagoudakis, M. (1999). Universal access to mobile computing devices through speech input. Proceedings of 12th International Florida AI Research Symposium (FLAIRS-99) (pp. 286-292). Menlo Park, CA: AAAI Press.

Manaris, B., McCauley, R., and MacGyvers, V. (2001). An intelligent interface for keyboard and mouse control – Providing full access to PC functionality via speech. Proceedings of 14th International Florida AI Research Symposium (FLAIRS-01) (pp. 182-188). Menlo Park, CA: AAAI Press.

Markowitz, J. A. (1996). *Using Speech Recognition.* Upper Saddle River, NJ: Prentice Hall PTR.

McAlindon, P.J., and Staney, K.M. (1996) The Keybowl: An ergonomically designed document processing device. Proceedings of The Second International ACM Conference on Assistive Technologies (ASSETS '96) (pp. 86-93). New York: ACM Press.

Morrison, D. L., Green, T. R. G., Shaw, A. C., and Payne, S. J. (1984). Speech-controlled text-editing: Effects of input modality and of command structure. *International Journal of Man-Machine Studies, 21,* 1:49-63.

Mostow, J. and Aist, G. (1999). Reading and Pronunciation Tutor. U.S. Patent 5,920,838. (Also see Mostow, J., Roth, S.F., Hauptmann, A.G., and Kane, M. (1994). A prototype reading coach that listens. Proceedings of 12th National Conference on Artificial Intelligence (AAAI-94) (pp. 785-792). Menlo Park, CA: AAAI Press.)

Murray, J. T., Van Praag, J., and Gilfoil, D. (1983). Voice versus keyboard control of cursor motion. Proceedings of the 27th Annual Meeting of Human Factors Society (p. 103). Santa Monica, CA: Human Factors Society. (cited in Shneiderman, 1997).

Napier, H. A., Lane, D. M., Batsell, R. R., and Guadango, N. S. (1989). Impact of a restricted natural language interface on ease of learning and productivity. *Communications of the ACM, 32,* 10:1190-1198.

Pausch, R. and Leatherby, J. H. (1991). An empirical study: Adding voice input to a graphical editor. *Journal of the American Voice Input/Output Society, 9,* 2:55-66.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., and Carey, T. (1994). *Human-Computer Interaction.* Reading, MA: Addison Wesley.

Prentke Romich, Co. (1997) $WiVik_2$ on-screen keyboard programs for Windows. www.prentrom.com/access/wivik.html .

Raman, T. V. (1996). Emacspeak–direct speech access. Proceedings of The Second International ACM Conference on Assistive Technologies (ASSETS '96) (pp. 72-79). New York: ACM, Press.

Roy, D. and Pentland, A. (1998). A phoneme probability display for individuals with hearing disabilities. Proceedings of The Third International ACM Conference on Assistive Technologies (ASSETS '98) (pp. 165-168). New York: ACM Press.

Schwartz, E. (2000). PDAs Learn to Listen Up. *Info World*, *22*, 6:1&10.

Smith, A., Dunaway, J., Demasco, P., and Peischl, D. (1996). Multimodal input for computer access and augmentative communication. Proceedings of The Second International ACM Conference on Assistive Technologies (ASSETS '96) (pp. 80-85). New York: ACM Press.

Shneiderman, B. (1997). *Designing the User Interface*, 3rd ed. Reading, MA: Addison-Wesley.

Trewin, S. (1996). A study of input device manipulation difficulties. Proceedings of The Second International ACM Conference on Assistive Technologies (ASSETS '96) (pp. 15-22). New York: ACM Press.

Weiser, M. (1994). The world is not a desktop. *Interactions*, *1*, 1:7-8.