Spoken-Language Access to Multimedia (SLAM): A Multimodal Interface to the World-Wide Web

David House
B.Sc., North Carolina State University, 1993

A thesis submitted to the faculty of the
Oregon Graduate Institute of Science & Technology
in partial fulfillment of the
requirements for the degree
Master of Science
in
Computer Science and Engineering

July 1995

The thesis "Spoken Language Access to Multimedia (SLAM): A Multimodal Interface to the World-Wide Web" by David House has been examined and approved by the following Examination Committee:

Dr. David G. Novick
Associate Professor
Thesis Research Advisor

Dr. Mark Fanty
Research Assistant Professor

Dr. Jonathan Walpole
Associate Professor

Acknowledgments

Above all, I would like to thank David Novick for the arrangements, guidance, ideas and support he provided in all aspects of this research.

I wish to give special thanks Oscar Garcia for his contributions to the remote-recognition model and Mark Fanty for work on the recognition module, as well as his thoughts on a speech-only interface to this system and other feedback. Further thanks go to Jonathan Walpole for his feedback and assistance as a thesis committee member, to Ken Maupin for his valuable help with World-Wide Web-, HTML-, and programming-related issues, and to Ron Cole for his ideas, support and feedback.

Additional thanks to Henry Churchyard for providing the code from htmlchek used for HTML document parsing, and to Michael Mauldin for providing statistics about common link words on the World-Wide Web. Thanks also to Yeshwant Muthusamy for providing a fast text-to-phoneme generator and to James Blakely for his help with creating the Motif code for the type-in version.

Finally, thanks to the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign for its role in the development of Mosaic.

Contents

List of Tables

List of Figures

Abstract

The World-Wide Web (WWW), a global networked information system based on hypertext, has become extremely popular since it became available in 1992. In order to improve the ease of access to the information available on the WWW, as well as to give increased exposure to spoken language systems, we developed Spoken Language Access to Multimedia (SLAM), a spoken language extension to the graphical user interface of the World-Wide Web browser Mosaic.

Although other research has been conducted on speech interfaces to hypertext, including speech interfaces to the WWW, SLAM differs in some key ways. For one, SLAM uses the complementary modalities of spoken language and direct manipulation to improve this interface to information on the Internet. Also, SLAM makes the advantages of spoken language systems available to a wider audience by providing a recognition server available remotely across a network.

This thesis describes previous work related to SLAM, particularly in the areas of multimodality and speech interfaces to hypertext and hypermedia systems, including speech access to the WWW. This thesis also examines the issues and architecture of what is

believed to be the first spoken-language interface to the WWW to be easily run across platforms.

This work is sponsored by a Small Grant for Exploratory Research (number 9069-120) from the National Science Foundation.

Chapter 1

Introduction

1.1 The problem

The World-Wide Web (WWW) (CERN, 1994) is a network-based standard for hypermedia documents that combines documents prepared in HyperText Markup Language (HTML) (NCSA, 1994a) with an extensible set of multimedia resources. The most popular WWW browser with available source code is Mosaic (NCSA, 1994b), a cross-platform program developed and distributed by NCSA, now running in X11-based Unix, Macintosh and PC-Windows environments. As a hypermedia viewer, Mosaic combines the flexibility and navigability of hypermedia with multimedia outputs such as audio and GIF images. The World-Wide Web, especially as viewed with Mosaic, is phenomenally popular. By mid-Spring of 1994, Internet traffic was doubling about every six months. Of this growth, the World-Wide Web's proportional usage was doubling approximately every four months. In absolute volume of traffic, use of the WWW was doubling every two and a half months (Wallach, 1994).

Much of the popularity of Mosaic can be attributed to its mouse-based interface, which can quickly, simply, and directly aid the user in browsing the variety of documents

available on the Internet. However, inherent limitations in mouse-based interfaces make it difficult for users to perform complex commands and to access documents that cannot be reached by the visible links. Speech-based interfaces, on the other hand, perform well on these types of complex, nonvisual tasks, but speech input to computers is not nearly as widespread as other input methods.

1.2 The approach

The SLAM system simultaneously addresses the limitations of mouse-based WWW interfaces and the limited popularity of speech-based interfaces.

By maintaining the full functionality of the mouse-based Mosaic WWW browser while adding the speech input option, a system has been created for which the strengths of each complimentary mode of input (modality) compensate for weaknesses in the other. The SLAM system does not merely add speech input to the existing Mosaic interface, but rather uses speech to allow access to information that was not directly available with the mouse-based system.

This research could broaden the market for speech-based interfaces in two ways. By making speech recognition available in a popular product like Mosaic, speech recognition will also become an increasingly popular way to access data. SLAM also enables the user to perform speech recognition on either the local machine or on a remote speech recognition server, so that it is not necessary for the user's client machine to have a speech recognizer in order to access the WWW with speech.

1.3 Overview of thesis

Chapter 2 describes others' research related to the SLAM project, including motivation for creating multimodal interfaces, previous work on input to hypermedia

systems, and research investigating spoken language interfaces to the WWW. Chapter 3 describes the differences in speech- and mouse-based interfaces. Chapter 4 discusses issues involved with building a speech recognition system for the WWW.

Chapter 5 describes the choices that were made in building the SLAM system architecture, and describes user and network testing of the system. Chapter 6 describes open issues and outlines directions for future work with the SLAM project.

Appendix A contains source code and systems requirements developed in the course of this research. Appendix B describes the time trials of the SLAM's current networked recognition client and shows the resulting data.

Chapter 2

Related Work

This chapter discusses previous and ongoing work in fields related to spoken-language navigation of the World-Wide Web. A motivation for research in this field is presented, followed by a brief overview of research comparing speech- and mouse-based input, particularly as used in multimodal systems. An overview of previous research into hypertext and hypermedia systems is given, with a mention of systems using speech access to hypermedia. The chapter concludes with a look at other groups exploring the issue of spoken-language access to the WWW.

2.1 Motivation

The SLAM project combines a variety of emerging technologies and techniques, such as spoken language interfaces, the World-Wide Web, and multimodal access to information systems.

This project is one way to address the important issue of studying multimodal interfaces involving speech. The report of the NSF Workshop on Spoken Language Understanding concluded that performance characteristics of multimodal systems was one of the key research challenges in the field of spoken language research:

> "Interdisciplinary research will be needed to generate novel strategies for designing multimodal systems with performance characteristics superior to those of simpler unimodal alternatives. Among other things, the successful cultivation of such systems will require advance empirical work with human subjects, building a variety of new prototype systems, and the development of appropriate metrics for evaluating the accuracy, efficiency, learnability, expressive power, and other characteristics of different multimodal systems." (Cole, Hirschman et al., 1995, 12)

Development and availability of a spoken-language enhancement to an interface for the World-Wide Web would also increase the availability and visibility of spoken-language technology in the computing community as a whole. This may encourage other researchers and developers to refine and include spoken-language systems technology in future systems. In fact, there may also be a complementary effect, since adding spoken-language input to the World-Wide Web is likely to make the Web more easily used and thus more accessible to the general population. The area of human language technology has been identified as a grand challenge area necessary to support the national information infrastructure technology. A report of the Information Infrastructure Technology Task Group identifies "Intelligent Interfaces" as one of four broad topic areas of the Information Infrastructure Technology and Applications (IITA) program, and states that "Advanced user interfaces will bridge the gap between users and the future National Information Infrastructure... Work in this area includes development of technologies for speech recognition and generation..." (National Coordination Office for HPCC, 1994, 16-17).

The possibilities and practicality of multimodal interfaces to the Web will not be discovered via analytic methods alone. A substantial amount needs to be learned through empirical and experiential methods such as system building. Indeed, the potential interactions involved in multimodal systems are so complex that it may be impossible to discern

their optimal structure without conducting advance exploratory research (Oviatt, 1992). Thus the determination of the important or tractable issues relating to such a project requires development, use, and testing of a spoken-language interface to the World-Wide Web.

Moreover, the availability of even an experimental spoken-language interface would enable the growing population of Web users to address these questions in the very practice of their own day-to-day computing. If a spoken-language interface is used in the "worka-day world" of cooperative computing (Moran, 1990) exemplified by the Web, then we will have (a) empirical evidence of its utility and (b) a fund of varied experiences with the interface that could contribute to improvements. This community of users can tell us what is right and wrong with spoken-language interaction for hypermedia, thereby offering directions for further research in the field. Indeed, a widespread, easily-available spoken-language interface on the Internet could provide results useful to spoken-language systems research as a whole. In short, from a practical standpoint the idea is to make the interface available and see what happens, as in the case of the original Mosaic interface and other WWW browsers.

2.2 Previous work with interface modalities for hypermedia

The graphical user interface (GUI), especially with pointer-based direct manipulation, has become the predominant model for human-computer interaction. Even in innovative settings such as the World-Wide Web, which provides a rich hypermedia environment that includes outputs in hypertext, images and sound, the inputs to the system remain key-board- and pointer-based. (As the most typical pointer is the mouse, we will use the term "mouse-based" interface to refer to pointer-based interfaces generally.)

The mouse-based direct-manipulation interface (Shneiderman, 1983) provided a rational and innovative means of interaction with computer systems. While physical pointing and bitmapped displays solved many of the problems with character-and-keyboard-based interfaces, direct manipulation based on physical pointing did not make use of the full range of expressive capabilities of human users. This omission was, no doubt, mostly a consequence of the relatively poor state of other means of expression as input modalities; spoken-language systems have made immense progress since 1983 (Cole, Hirschman et al., 1995).

Adding spoken-language capabilities to hypermedia holds the promise of extending users' abilities in ways they find appealing. Empirical studies of multimodal interfaces have looked at user preferences for different kinds of inputs. For example, Rudnicky (1993) showed that users preferred speech input, even if it meant spending a longer time on the task, when compared with using a keyboard and a direct manipulation device. Oviatt and Olsen (1994) found that users of multimodal interfaces displayed patterns of use that reflected the contrastive functionality of the available modalities.

Other researchers have investigated the comparative advantages of multimodal interfaces, including Cohen (1992) and Oviatt (1992, 1994). One of the goals of this research has been to attempt "to use the strengths of one modality to overcome for the weaknesses of another" (Cohen, 1992, 143), who proposed a framework for this analysis. Cohen's analytical framework involves comparing the strengths and weaknesses of modalities with respect to factors such as:

- intuitiveness,
- consistency of "look and feel,"
- whether options are apparent,

- safety,
- feedback,
- "direct engagement" with an entity,
- ability to describe,
- use of anaphora,
- establishing and maintaining context, and
- use of temporal relations.

Cohen studied multimodal interfaces in general terms, without specific consideration of interfaces for hypermedia. For multimodal interfaces in general, then, he observed that the advantages of pointer-based interfaces are that they are generally intuitive, unambiguous, and, if well-designed, can have a consistent "look and feel." Drawbacks to using such interfaces include difficulty in selecting items not currently visible, poor support for temporal relations, and difficulty using context to specify relations. Natural language systems overcome some of the weaknesses of pointer-based interfaces by allowing the specification of context, temporal relations, and unseen objects. On the other hand, language has the problem that the user may not know the vocabulary of the recognizer. Spoken language systems are also prone to other problems such as ambiguity and other causes of recognition errors. (Cohen, 1992).

2.3 Previous work with speech access to hypermedia systems

Interactive hypertext systems have been proposed for fifty years; a useful survey is provided by Arons (1991). Such systems have a number of advantages for information retrieval over traditional databases, including that there is no need for training the user on the system and users do not require knowledge of a topic before searching for information. Some disadvantages of such systems are that users will have difficulty in actually getting

specific information, and are likely to encounter the well-known "lost in hyperspace" effect (Domel, 1994; Whalen, 1989) during which users get sidetracked and lost while navigating through a hypermedia environment.

One system (Stock, 1991) combines natural language and hypermedia to explore Italian frescoes. This system uses the hypermedia aspect to organize unstructured information, and used the natural language aspect to help alleviate the problems of disorientation and the cognitive overhead of having too many links.

Other groups have investigated using speech with hypermedia systems. One hyperspeech system (Arons, 1991) enabled the user to navigate in an audio environment without a visual display; speech recognition was used to maneuver in a database of digitally recorded speech. This system was similar to a speech-only WWW browser in that the speech interface was goal-directed; the speech provided a form of direct addressing that is difficult to capture in other interfaces, so that the user felt that they were navigating and in control. Arons acknowledged that "representing and manipulating a hypermedia database becomes much more complex in the speech domain than with traditional media." Related systems include those described by Resnick (1990) and Muller (1990), both cited by Arons (1991).

2.4 Previous work with spoken-language extensions to WWW browsers

Many groups around the country, and presumably around the world, are working on projects that are similar in many ways to OGI's SLAM system.

Earlier versions of MacMosaic had been compiled with speech recognition enhancements, but those compilations are no longer being performed, although they could be activated with some code changes (Stephenson, 1994).

MIT's Spoken Language Systems group has been working on GALAXY (Goddeau, 1994), a distributed system for on-line information that handles the natural language aspects of the system at a remote-recognition server. While the current focus of the GALAXY system is the travel domain, MIT is also believed to be applying this technology to creating a speech interface to the WWW as well.

Raman at DEC has begun work on a spoken language extension to Mosaic called RETRIEVER (Raman, 1995) that focuses on allowing easier access to the Web to people with disabilities. Paciello at DEC is working with Hardin at NCSA on the Mosaic Disability Project (Paciello, 1995), one aspect of which is speech recognition.

Hemphill at Texas Instruments has completed a prototype speech-enabled Mosaic (Hemphill, 1995) that allows for associating extended grammars and dialog states with links and hotlist items. Arbash at SRI developed a speech interface to Mosaic based on work related to the Xtalk project (Arbash, 1994). Both of these are based on local speech recognition, unlike OGI's remote-recognition system.

Chapter 3

A Comparison of Speech- and Mouse-based Interfaces to Hypermedia

This chapter examines issues involved in creating a spoken language extension to a hypermedia system, in particular the Mosaic World-Wide Web browser. I discuss general issues involved in creating a multimodal interface with spoken language and mouse-based systems.

Cohen's analytic framework (discussed in Chapter 2) will now be particularized and extended to deal specifically with the comparison of speech-based and mouse-based interfaces for hypermedia. This will be done in two steps, by looking at mouse-based and then speech-based interfaces in terms of their respective advantages and disadvantages for hypermedia systems.

3.1 Mouse-based interfaces to hypermedia: advantages and disadvantages

The physical pointing involved in mouse-based interfaces is the source of both advantages and disadvantages for this modality. From the user's perspective, pointing has the

traditional advantage of direct manipulation, namely reference specified through a combination of action and location, as in double-clicking an icon to start a program. Moreover, the interface generally provides immediate feedback to the user that the reference was successful, typically by highlighting the selected entity. From the point of view of the author of a WWW document, mouse-based pointing has the advantage that the reference can be completely specified: the label of a link will appear exactly as the author wrote it. Additionally, physical pointing in this context has no referential ambiguity; when the user clicks a mouse button, the user and the author both know exactly to which entity the user is referring.

Mouse-based interfaces also have a number of disadvantages, particularly of the "lost-in-hyperspace" variety. This well-known problem was identified for hypertext systems by Whalen and Patrick (1989), who proposed a text-based natural-language solution.

> "Users can have trouble actually getting to the specific information required. They may have to navigate through a large number of paragraphs to get to the desired goal. Along the way, users are likely to get sidetracked and lost." (289)

When reference is based on physical pointing to a graphically-represented entity, the absence of such an entity on the screen means that the user *cannot* refer to it. In other words, the act of reference depends on the physical location of the referent's presentation, which in hypermedia may be pages and documents away.

Hypermedia interfaces typically have standard features such as a "hotlist" and history windows in order to give users a place that contains references they might want and that are otherwise not displayed. But the user might also prefer to refer to an entity by a name other than that specified by the author; the only way the user has to specify an entity is to click on it. Finally, the spatial nature of the interface limits the set of things to which the

Table 3.1: Mouse-Based Interaction with Hypermedia

Advantages	Disadvantages
1. Deictic reference and combination of action and reference	1. Reference depends on location of referent
2. Author completely specifies the representation of the entity	2. (a) User might prefer another representation and (b) no other representation possible
3. No referential ambiguity	3. Vocabulary of references limited to those with visible links
4. Generally gives immediate feedback that user's action was understood	4. Difficult to express complex acts

user can refer. Users cannot *describe* entities (Cohen, 1992) instead of pointing to them. Similar problems exist with respect to actions. Because they are typically accomplished by selecting a command from a menu or by clicking on an icon, it is difficult to express complex actions other than as a perhaps tedious series of primitives. The advantages and disadvantages of mouse-based interfaces for hypermedia are summarized in Table 3.1.

3.2 Spoken-language-based interfaces to hypermedia: advantages and disadvantages

Many of the advantages and disadvantages of spoken-language-based interfaces for hypermedia turn out to be complements of those for mouse-based interfaces. From the user's standpoint, the ability to refer to an entity no longer depends on the location of its graphical representation.

Indeed, *all* referents are potentially available because the user can simply say the name of the referent without having to see it displayed. A related advantage is that the user can now have a number of different ways in which to refer to entities. Similarly, multiple action primitives could easily be combined into a single complex action that could include temporal and other sophisticated concepts that are not expressible in mouse-based inter-

faces. Another advantage is that the user's hands are freed for other activities. Indeed, it might be possible to build a spoken-language-only interface to hypermedia that could serve users by telephone instead of requiring a GUI.

Speech input to hypermedia also has characteristic disadvantages that are often reciprocal consequences of its advantages. For example, because references no longer depend on physical location, references may become ambiguous: a "hotlink" may be uniquely accessible via the mouse but ambiguously accessible via speech because another hotlink might have the same label. This strongly suggests that designers of hypermedia interfaces should avoid multiple uses of "Click here" as a hotlink label; rather they should use a lexically meaningful label that refers to the semantics of the linked entity (Conte, 1994). Similarly, there could be confusion between names of labels and names of actions; for example, the word "back" could be used both as a command to display the previously viewed document or a link label to a different document. Although speech interfaces make all referents available, the user may not know all available referents. However, the user is no worse off than in the mouse-based case, where it is not even possible to refer to other entities directly.

Although the hands-free nature of spoken-language interfaces is appealing, early implementations of spoken-language interfaces to hypermedia may have to rely on "touch-to-talk" methods so that the recognizer is not confused by extraneous speech (Lunati & Rudnicky, 1990), or by prefixing the utterance with a keyword, as in some commercial interfaces. Similarly, while spoken-language understanding could possibly provide a speech-only interface, there would be a number of problems with unimodal application of speech to hypermedia, including (a) straining user tolerance in getting input

Table 3.2: Spoken-Language Interaction with Hypermedia

Advantages	Disadvantages
1. References not dependent on location	1. Possible ambiguity
2. All referents are available	2. User may not know all available referents
3. Hands free	3. Might have to use touch-to-talk to avoid extraneous sounds and speech
4. Could provide access to information when GUI not available	4. Problems with audio-only: (a) too much text (b) pictures (c) navigation (d) presentation of links
5. More direct expression	5. Unknown words, unlimited vocabularies
6. More than one way to refer to an entity	6. Multiple links may have same key words; or link and command may be the same
7. Can express more complex action	7. Difficult to refer to graphics such as bitmaps, icons and pictures

through extended synthesis of text, (b) removing meaning from images, (c) making navigation more difficult, and (d) not providing the user immediate knowledge of the names of new links. Indeed, consideration of these factors suggests that the application of spoken-language technology as a multimodal extension to a hypermedia browser would likely be more immediately useful than development of a unimodal, speech-only interface. The advantages and disadvantages of spoken-language input for hypermedia are summarized in Table 2.

From this comparison, I conclude that mouse-based and speech-based modalities have a high degree of complementarity that could improve the usefulness of hypermedia systems. This could lead to a *synergistic* interaction style (Lefebvre et al., 1993; Nigay & Coutaz, 1993) that allows multiple modalities to perform a task.

Chapter 4

Issues in Creating a Speech-Enabled WWW Browser

This chapter focuses on the techniques and components required to create a spoken-language extension to a hypermedia browser. The options available in choosing which parts of the system to make accessible by speech are examined, as well as the different forms of ambiguity between the components. Speech input issues such as variability among microphones, touch-to-talk systems, and recording format are also discussed. Furthermore, a variety of speech recognition models, including distributed systems and vocabulary- and speaker-independent models, are compared with their alternatives.

4.1 Components accessible by speech

In building a spoken-language extension to a hypermedia system, one needs to decide what aspects of the system to enable the user to access with speech. Hyperlinks, commands, and lists of previously-stored URLs (known in Mosaic as "hotlists") are some of the system components that could be made accessible by speech. This section looks at the advantages and disadvantages of adding these functionalities. One disadvantage of adding speech to a hypermedia system is that spoken ambiguities can occur between these

system components. An advantage of using speech in the makes it possible to offer multiple names for an individual component.

4.1.1 Speech access to hyperlinks

The development a spoken-language extension to a hypermedia system would presumably enable the user to access the hypertext links within the body of the World-Wide Web document with speech. It would need to be decided if the user could only access links that are visible to them, or, since speech allows access to non-visible referents, if the user could access any link within the current document.

The more items the recognizer has to choose from, the less accurate the overall recognition rate will be in general. Only allowing speech access to links currently displayed on the screen would restrict the number of links that would need to be recognized and should improve recognition rates on long pages with many links. However, allowing access to links on the entire page spares the user from scrolling through a long page looking for an off-screen link if they know the link is there and could more easily access it by using speech.

The user may want the option of selecting this mode, and may want to change modes during a session, so the system designer could add this option.

4.1.2 Speech access to hotlists

The user may also wish to access a list of links stored from previous sessions (known as a hotlist in Mosaic) with speech. In Mosaic versions 2.4 and earlier, hotlists were implemented as linear lists that are slow to search via scrolling in a mouse-based system. Even with the nested hotlists (NCSA, 1994d) implemented in Mosaic 2.5 and beyond, it could still be time-consuming for the user to search their hotlist for a particular item. Adding

speech could allow the user faster access by not requiring that they user hunt for this information; if the user could remember the exact name of the item, or the system could assign multiple names to a single entity, the user would not even have to bring up the hotlist information if they were to use speech to search it.

If users are to be provided access to hotlist information with speech, one must decide whether to use the existing hotlist structure of Mosaic or to create a separate hotlist of information available only for speech. The advantage of using Mosaic's list is that less additional structure will need to be added to the system (with fewer changes and less maintenance as well); the existing interface should be more stable than newly-designed code, and it is already familiar and easily accessible to the user. On the other hand, users may have different needs and intentions for how they use speech for Mosaic and how they use the existing hotlist function, so some way could be devised to handle hotlists for speech in a way that is separate from the way Mosaic currently handles hotlists, perhaps as a completely separate structure or (for Mosaic 2.5 and beyond) as a separate subdirectory within the nested hotlist feature.

The system designer may also want to decide if the system will update these speech-accessible hotlists dynamically from the browser during a WWW browsing session, or if all of the speech-accessible hotlist information will be enabled for speech ahead of time. If the pronunciation models for the hotlist are compiled ahead of time, then large files will need to be generated and stored. For a distributed recognition system, these files would need to be stored at the recognizer's site to avoid larger network delays that would result from passing this information each time. Precompiling this information would result in faster processing times at the cost of increased network delays (if stored at the user's site) or storage and maintenance problems (if stored at the server's site).

4.1.3 Speech access to commands

System developers could decide if they wish to implement spoken commands (such as "back," "forward," and "home") within the system. Arguments against implementing these commands are that they are readily available from the mouse based interface, by (1) a bottom row of buttons, (2) a keyboard press when the mouse is focused within the Mosaic window, and (3) from the top menu bar. Adding spoken commands enhances access by disabled users, and would allow for greater consistency of input modality for the user (instead of being required to constantly switch between mouse- and speech-based input.

If speech access to Mosaic commands is allowed, it remains to be determined which commands (if not all) are to be made available to the user. The concept of allowing the user to use spoken commands also brings up the question of whether to allow the user to use new spoken commands (such as "back 3 pages.") or to only allow the use speech to access existing commands (such as "back" and "home").

4.1.4 Ambiguity in alternative outputs

Ambiguity can occur when using a spoken language interface to the WWW. There can be confusion between link labels (for example, the user may have more than one link on a page labeled "Click Here"; if speech was used to access that page, how would the system know which link was referred to by the word "here"? Examples of WWW pages with this problem include "Hartsfield School's Home Page" (Figure 4.1) (Olary, 1995) and "Faculty of Dentistry Home Page" at Kyushu University (Figure 4.2) (Saito, 1995). Goddeau's "Recent Publications" page (Figure 4.3) (Goddeau, 1995) would produce spoken ambiguity because of the multiple occurrences of the word "Postscript."

A similar problem would be homophones appearing on the same page. For example, in

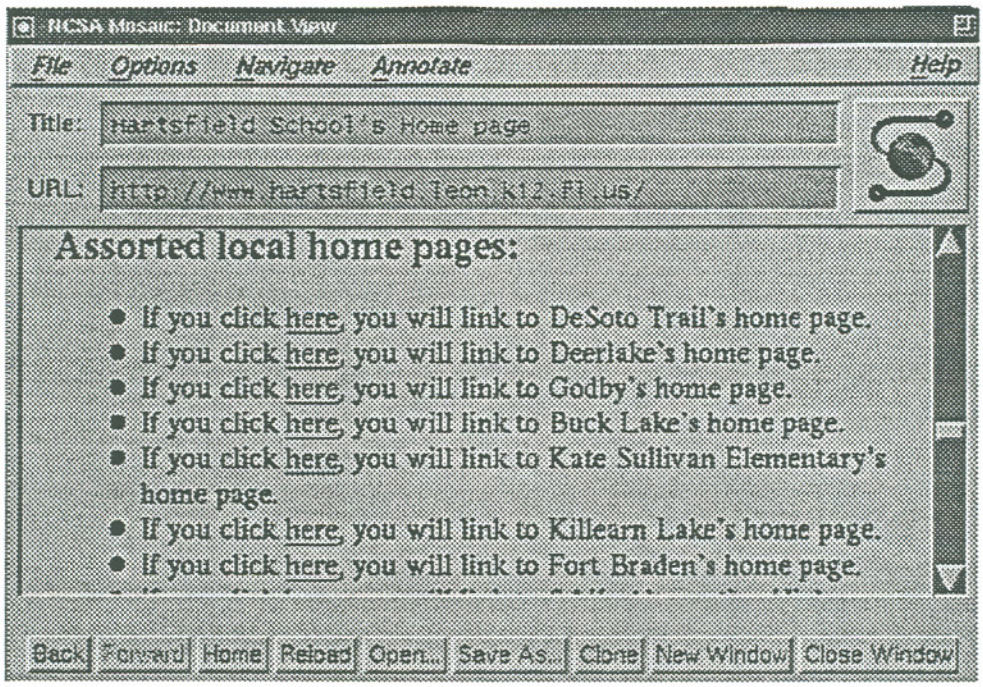

Figure 4.1. Hartsfield School's Home Page (overuse of "here" link)

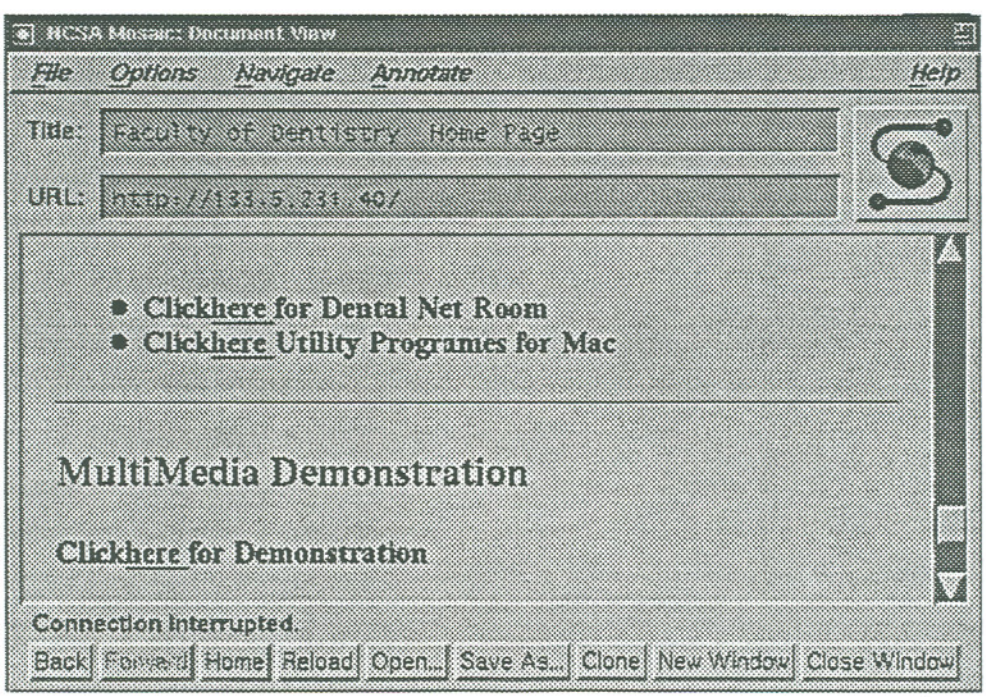

Figure 4.2. Faculty of Dentistry Home Page (overuse of "here" link)

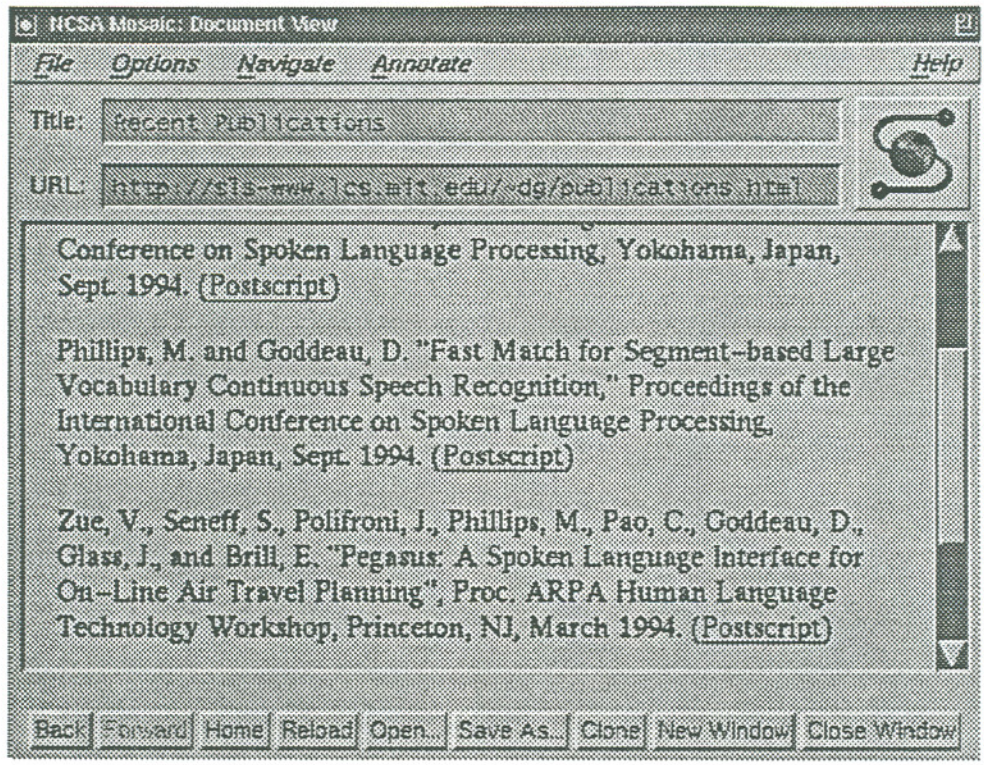

Figure 4.3. Recent Publications page (overuse of "Postscript" link)

addition to "Click Here" links, there could be a link such as "Would you like to hear about our program?" Again, how would this ambiguity be resolved? Related confusions can arise with items labeled the same way within the user's hotlist. Furthermore, there can also be confusions between link labels, system commands, and items from the hotlist. It is conceivable, for example, that there could be labels on both the current page and in the user's hotlist with the label "Back," and this word could also relate to the command for the system to return to the previous page. Even if the link labels do not sound identical, there could be problems if the words sounds similar. For example, a link labeled "ohm" could be confused with the command "home."

One way to resolve this problem would be to plan for it ahead of time to avoid it; when a user's hotlist or current page is read into the system, the links could be examined to find conflicts in the form of repeated words or homophones. However, the system designer

may not want this processing delay before each page is shown to the user. Another solution may be for the system to begin some type of dialogue with the user, to find out which context they meant for the conflicting word. For example, the system could highlight a particular reference relating to that word and ask "Is this the reference to 'here' you meant?" Yet another solution would be to pick one of the alternatives, and let the user press the "back" key or some other option to repeat their entry so that the user is not bogged down with constant bombardment of dialogue boxes.

4.1.5 Handling multiple pronunciations

Enabling the HTML document author to provide the user with multiple alternative pronunciations for given link labels has several advantages over only allowing one pronunciation per label.

Using multiple pronunciations seems to be a natural approach for link labels that have multiple pronunciations, such as GIF, that can be spelled out letter-by-letter, or pronounced with a hard or soft "G". It may also be worthwhile to create multiple pronunciations for link-label words that are likely to be unfamiliar to people. For example, there may be some value is putting in the pronunciation "mowpin" for the link label containing the last name Maupin, even though the correct pronunciation is "mawpin." It may also be possible to add multiple pronunciations of words for the sake of conveniences: for example, allowing the user to say "O G I" for a link labeled "Oregon Graduate Institute."

However, there are several reasons not to add multiple pronunciations to link labels. It is not clear how the user will be aware of the existence of the multiple pronunciations, except by possibly seeing these pronunciations in the HTML source code. Also, these multiple pronunciations may add confusion to the recognizer more often than actually being useful as a feature. These potential problems of adding multiple pronunciations to

hyperlink labels indicate that developers should use caution when adding this capability to a speech-enabled page.

4.2 Microphone variation

Users may try accessing a speech-enabled hypermedia system with a variety of input devices, from poor-quality telephones to high-quality microphones. Designers of such a system may wish to have some mechanism in place for handling this variability in the input device. This mechanism could take a variety of forms, from an explicit statement telling the recognizer what the input device is, to an internal system at the server that is flexible enough to handle this variation (although this flexibility may result in less overall accuracy).

4.3 Input style: Open microphone or touch-to-talk

The two main choices for recording the user's speech for such a system are by an "open microphone," for which the user is able to speak to the system at any time (perhaps by prefixing the speech with an attention-getting command word like "computer"), or by "touch-to-talk" for which the user controls when the speech gets recorded by pressing a system component to activate (and perhaps again to deactivate) the recording device.

Advantages of the open microphone method are that truly hands-free input is available, which would be especially valuable for disabled users or users simultaneously performing hands-busy tasks. Advantages of closed microphone (touch-to-talk) systems are that they are more reliable than open microphone systems. Also, there may be system considerations (such as a noisy environment or distributed recognition components) that could make end-of-utterance detection and other speech processing techniques difficult or impossible, thereby making use of the touch-to-talk method necessary.

4.4 Recording format

Local systems interacting with a remote-recognition system might use a variety of recording formats. IBM-PC's, Macintoshes, and Unix systems all have different recording formats associated with their systems. Designers of a speech-enabled hypermedia system would need to decide if their system will accept a variety of formats, or merely restrict the user to one standard recording format. If different formats were being used, a mechanism for notifying the server of the varying formats and allowing for the conversion to a standard format would need to be established.

4.5 Location of recognition

There are two models for the location where a SLAM system could perform the speech recognition. The recognition could be performed on the local machine, by what is called the *local-recognition model*, or the user's speech could be passed across a network to a remote server for remote recognition and a result sent back to the user's machine. The *remote-recognition model* provides a number of advantages. First, it helps to spread the popularity and use of spoken-language systems without the hardware costs otherwise associated with such systems. Because the recognition is done remotely, the user could use a relatively inexpensive machine with limited memory and still perform WWW navigation with speech. Second, this approach could also serve as a foundation for a speech-only interface over the telephone, which would allow the user to access the variety of useful information available over the Internet without needing a terminal. Other advantages of the remote-recognition model include the ability to control and collect the spoken utterances of the users (after being given permission to do so) from around the world for the building of standard language corpora, which will enable further development of the field.

Also, as the state of the art in speech-recognition capabilities improves, the software would only need to be updated at the central SLAM server site instead of at all sites that were using the interface.

There are also several potential disadvantages to the remote-recognition model. The transfer rate to and from the central recognizer may be quite low; however, given the likely short length of the transferred speech and the normal delays in accessing WWW documents, this effect does not appear to be serious (see section 5.10 for network rates for SLAM). Also, additional delays may occur from multiple clients trying to access the single central recognition server simultaneously. If the recognition is performed locally, the results should be faster because there is no network overhead, and the system protects the speaker's privacy by not passing their speech and its associated results back and forth across the network. Local recognition also allows for systems to perform end-of-utterance detection and other speech processing for recognition before the user has finished speaking, which is especially useful for the open microphone input option. For networked recognition systems, this type of processing may be difficult or impossible because of network breaks and delays.

4.6 Choice of recognizer

Designers of speech-enabled hypermedia systems face a variety of options with respect to the recognition subsystem.

First, the recognizer could be either vocabulary-dependent or independent. As part of a vocabulary-dependent system, a recognition system trained on a particular set of documents would generally have a better recognition performance on these pages. A vocabulary-independent system trained on sub-word components such as phonemes would be better able to handle recognizing words the system had not seen before, at the cost of less

reliable recognition than if the system had been trained on those particular words.

Second, a speech-enabled hypermedia system could be trained on individual speakers to create speaker-dependent recognition, or could be trained on a variety of speakers for speaker-independent recognition. Speaker-dependent systems have better recognition for the speakers they are trained on, but generally perform significantly worse for other speakers.

It may be possible to use a combination of recognizers to gain speech access to hypermedia. For example, vocabulary-dependent recognizers could be used for certain static pages for improved recognition accuracy, while vocabulary independent-recognizers could be used for the remaining pages. The obvious advantage would be improved recognition for the vocabulary-dependent pages, although there may be a higher overhead cost associated with switching among multiple recognizers. A system with multiple recognizers could also be set up across a network at multiple sites for increased system flexibility, so that users could choose the fastest recognizer or the recognizer with the best features for their needs, as well as for redundancy, in case one or more of the recognition sites become disabled.

4.7 Generating recognition models

Before a system can recognize what the user says, it must know what utterances to expect. This is handled by the hypermedia recognition system converting the text strings to be recognized (such as hyperlink labels from the current hypertext document, or from the user's hotlist) into pronunciation models to provide a phonemic representation of the text. For OGI's recognizer, these pronunciation models are then converted to state-based word models that the recognizer compares against the user's speech to find a best match.

The creation of pronunciation models and word models are non-trivial tasks without

wholly satisfactory solutions. The ideal system would create completely accurate models from any text string in real time. With the available technology, this is not possible. Therefore, compromises must be made in either speed or accuracy. For example, to match the word "Heather" to its proper pronunciation, a fast-but-inaccurate phoneme substitution might make the mistake of replacing the "ea" in "Heather" with a "long E" sound (like the vowel sound in "beet") instead of a "short E" sound (like the vowel sound in "bet").

These models can either be generated ahead of time or "on-the-fly." Generating these models ahead of time, requires storing this information somewhere. For the networked model of the recognition system, the information could either be stored at the user's end, on the WWW pages themselves (thereby creating what are known as *speech-enabled documents*), or at the site of the speech recognition server. Storing the models at the recognition site allows for fast recognition and prevents passing large chunks of recognition-related data from client to speech server but this solution does not scale well, as the recognition site becomes the sole location for storage and maintenance of the recognition models.

Chapter 5

The SLAM System

SLAM adds spoken language as an input to the Mosaic browser by enabling interaction with a remote server that provides (a) speech-enabled documents (see section 4.7) and (b) the recognition systems needed to use them. SLAM provides a relatively simple extension to Mosaic plus access to a SLAM server at the Oregon Graduate Institute (OGI) that performs speech recognition for a set of speech-capable hypermedia documents. In these documents, users are able to select hotlinks with spoken language. A major advantage of the remote-recognition model is that recognizers will not have to be developed and supported for all client platforms; rather, spoken-language interaction can be added to additional platforms through creation of new versions of the SLAM-extended Mosaic, which will require only minor modification of Mosaic.

SLAM is the first generally-available multimodal spoken-language interface to the World-Wide Web that can be easily run across platforms. No other such interface has been reported in the literature.

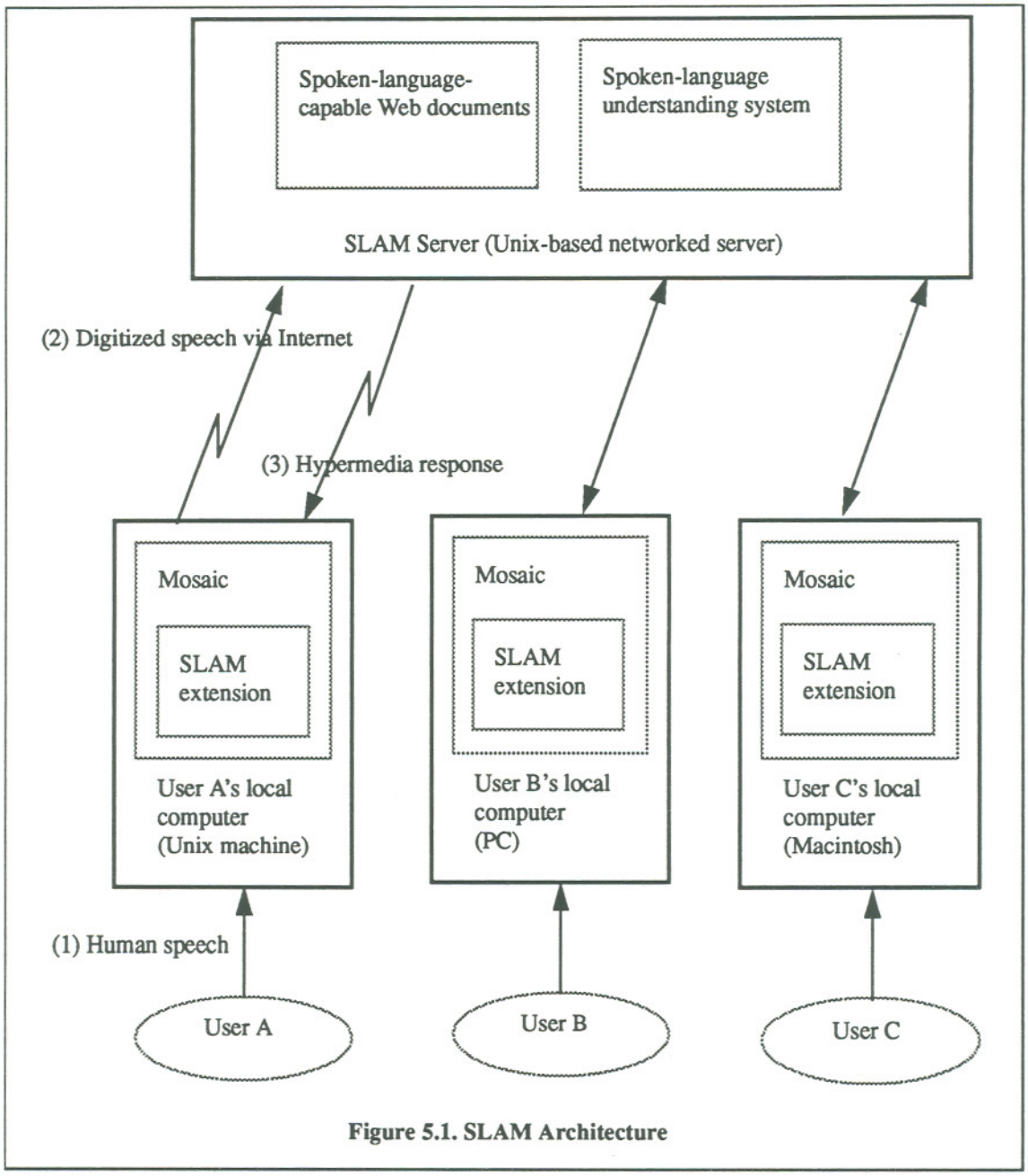

Figure 5.1. SLAM Architecture

5.1 SLAM architecture

SLAM is a spoken-language interface system for Mosaic based on local access to a remote- recognition-capable Web server. The overall architecture, as depicted in Figure 5.1 is based on a Web server that has spoken-language software and "speech-enabled"

HTML Web documents. The SLAM server receives, recognizes, and responds to requests from users running an extended version of Mosaic on their local computer. Users can use spoken language to select hotlinks in documents on the SLAM server, as well as hotlist items stored from previous sessions.

Users on heterogeneous platforms—such as Macintoshes, PC's and X-Windows interfaces for Unix—will interact as usual with their local Mosaic browser to the World-Wide Web. As indicated by Arrow 1 in Figure 5.1, the user speaks to his or her local machine. The local Mosaic browser will contain extended code that digitizes the user's utterance and, as indicated by Arrow 2, sends the digitized signal to the SLAM server. The server processes the speech signal and matches the utterance to a WWW uniform resource locator (URL). As indicated by Arrow 3, the server then sends back to the local machine a hypermedia response, typically a new HTML document.

As this discussion indicates, SLAM's architecture is based on the remote-recognition model (discussed in chapter 4.5, with code for the server and client shown in Appendices A.3 and A.4 respectively) where the local browser does not (necessarily) perform recognition and the remote server provides both speech-capable documents and the speech recognition necessary for their full use.

A final consideration is that the SLAM product will not allow access directly to the worldwide network of Internet documents; authors of HTML documents will have to prepare speech-capable documents specially, or eventually, have a script to automatically create speech-ready versions of existing documents.

5.2 SLAM implementation

The principal components of SLAM's implementation include a minor extension to Mosaic's GUI and the networking and recognition modules associated with the server. The major functions operate as follows:

1. As SLAM starts up, the user's "hotlist" information file consisting of links stored from previous sessions is read in from the user's home directory. SLAM also saves pronunciation models for the "hotlist" items in the user's home directory so that these models do not need to be generated on the fly.
2. The user can use the extended browser to navigate the WWW in the same way that they use the Mosaic browser, by using the mouse to select hotlinks within the current document to bring up other documents.
3. Once the user reaches a speech-capable document (denoted by an icon at the top of the document), the user is also able to use the touch-to-talk speech facility of SLAM to select links. For documents that are too long to fit entirely on the current browser screen, SLAM views the document in its entirety, rather than focusing only on the part of the page that is visible to the user. This enables the user to use speech to specify items that do not appear on the current page.
4. As discussed in Appendix A.1, when the speech button is pressed, three things happen:
   a. Mosaic sends to the server the URL of the current document, so that the server can set up the recognizer with the right vocabulary.
   b. Mosaic prepares to accept a new document from the server in the usual manner, except that as the document comes back its headers are parsed for the speech pragmas.

c. A SLAM function digitizes speech from the system's usual audio input and sends it to the open SLAM server.

5. The SLAM server's recognizer tries to find a best match with the available word models corresponding to the resulting outputs that came from the user's "hotlist" and current page links, and returns the URL corresponding to the result back to the client machine. The client's extended Mosaic browser then retrieves the document specified by the URL.

The speech recognizer determines the appropriate target vocabulary and phrases through the SPEECH= tags, the arguments of which are pronunciations models of the label. In the current implementation of SLAM, SPEECH= tags appear together at the top of the "speech-ready" document in a form such as:

<SPEECH= ao r eh g ax n [.pau] g r ae j uw ih t [.pau] ih n s t ih t uw t
w eh dh axr >

in that each line in the SPEECH= section relates to a corresponding link label within the file (Note: [.pau] refers to a "pause" in the speech). For example, the first pronunciation model within this example, relating to the words "Oregon Graduate Institute," would refer to the first link in that document, which would be of a form like:

<A HREF="http://www.ogi.edu/"> Oregon Graduate Institute</A>

while the second pronunciation model would correspond to the link corresponding to the word "weather."

5.3 SLAM speech recognition

OGI has used neural-network-based recognition for limited vocabulary tasks for a number of years (Cole et al., 1990; Fanty et al., 1993). Given speech to be recognized, OGI's current technology (Cole et al., 1994) works as follows:

1. A seventh-order Perceptual Linear Predictive (PLP) analysis (Hermansky, 1990) is performed every 6 msec using a 10 msec window.
2. A neural network classifies every 6-msec frame as voiced or not voiced.
3. The PLP and voicing features in a 168-msec input window are used by a neural network to classify each frame phonetically. If task-specific training data are available, the network is trained on that. Otherwise, a network is trained on a phonetically diverse corpus of speech. The latter approach usually results in a significant loss of performance.
4. A modified Viterbi search finds the two best-matching utterances. The possible utterances are defined by a word-spotting grammar that typically allows any speech followed by one of several target words or phrases followed by any speech. Word pronunciations are provided by a dictionary and are usually hand-tuned to achieve maximum performance.
5. A confidence measure is computed as the difference between the average frame score of the top choice and the average frame score of the second choice.

A review of typical hypermedia documents (Mauldin, 1994) reveals that links are often proper nouns. This suggests that recognizers for hypermedia browsers will have to rely on vocabulary-independent techniques rather than extensive task-dependent training. The ability to generate accurate phoneme representations of these labels in near real-time would be a valuable step towards developing a future system that does not rely on "speech-ready" documents. SLAM uses a context-independent, task-independent phonetic classifier trained on the OGI Continuous English Speech Corpus, that contains the unconstrained speech of 690 speakers, each talking for up to one minute. The recognizer in the

current SLAM system was trained on 147 of these monologues, and produces scores for 39 American English phonemes.

SLAM uses a dictionary to find word pronunciations and uses automatic text-to-phoneme mapping to create pronunciations for words not in the dictionary. With the current system, "speech-ready" pages are created semi-automatically by processing each link label first through the Moby dictionary (which is strictly a table lookup) and through Bellcore's Orator text-to-speech system (which can create pronunciation models even for words the system has never seen). The script used in the creation of "speech-ready" pages is discussed in Appendix A.2.

A wide variety of microphones and recording environments will be used by remote users of SLAM. This sort of variation typically has a significant impact on the accuracy of recognizers. Hermansky and Morgan have developed RASTA spectral processing for improved robustness in different recording environments (Hermansky et al., 1994). Future versions of SLAM's recognizer will use RASTA -PLP to increase recognition robustness.

5.4 Scope of speech-access to hyperlinks

The current SLAM system enables the user to access links on the current page and items from the user's Mosaic hotlist. Allowing the user to access commands would be a straightforward addition, but this feature was not added to the current demonstration system because (1) the commands that would have been implemented (such as "back") are mostly immediately available to user through the keyboard input to the system (for "back," all that is needed is to type the letter "b" or "B"), through pull-down menu bar options, or through the buttons available at the bottom of the screen and (2) most of these commands are short words, which are recognized incorrectly more often than longer utter-

ances. However, the ability to use voice input to access commands would be useful for disabled users and is being considered as a feature for future versions of the system.

The current implementation of SLAM allows the user to access all links on the current document at all times, rather than being restricted to what the user sees. The obvious advantage of this method is that the user is no longer restricted to accessing links that are visible, thereby taking advantage of the spoken nature of the input. A disadvantage would be for long documents with many links, for which the user would wish to restrict the confusion to the recognizer by focusing only on links currently within the visible screen.

5.5 Parsing HTML documents

The current system uses the program **xtraclnk.pl** program that comes as part of the **html-chek** package (Churchyard, 1995) to extract the link labels and associated URLs from a HTML file.

A program external to Mosaic was chosen to perform the link parsing for several reasons. First, one goal of the implementation was to make as few changes to the Mosaic code itself as possible, for simplicity and modularity of implementation, as well as not having to worry about accommodating these changes to future releases of Mosaic. Second, a full set of links is needed from each document; although Mosaic maintains a copy of the entire document as a variable, Mosaic appears to extract links on a per-screen basis, while the current implementation required that the document be parsed on a per-document basis.

5.6 Microphone input method

Of the various methods of microphone input, a touch-to-start/stop-talking system was chosen for SLAM. Using this method, the user presses a key ("X" in the current design)

on the keyboard, says the appropriate utterance, and then presses the key again upon completing the utterance. The speech (combined with other SLAM components) is then sent to the recognizer.

Having an open microphone was considered and deemed highly impractical for the remote-recognition model in particular because constant reading of the speech would cause too much overhead across the network.

Earlier versions of the SLAM system used a "touch-to-talk" system that automatically stopped recording after a predefined number of seconds but finding a perfect input duration is impossible because of the varying lengths of linknames; moreover the extra "dead air" surrounding the sample could contain extraneous speech and other noises that could confuse the recognizer. In early tests, this effect was particularly noticeable for short link labels (particularly for ones with only one word, like "movies" and "weather").

An end-of-utterance detector that would time-out when the user had stopped speaking was considered. However, this method requires constant real-time processing of everything recorded by the microphone, and could not be implemented for the networked model because of network delays that would prevent this from being handled in real time.

5.7 Generating and storing speech-enabled documents

The generation of speech-enabled documents is a process that is handled semi-automatically with the help of a script. This script, called slam-enable, works as follows:

1. The slam-enable script is called with two arguments: 1) the URL of HTML document one wishes to speech-enable, and 2) a name to be assigned to the new document.
2. **url_get** (Lund, 1994), a program that takes a URL as an argument, returns the HTML contents of the file associated with the URL to STDOUT.

3. The contents of this HTML file are sent through **xtractlnk.pl,** which creates two temporary files that contain 1) a listing of the hyperlink labels for the HTML page and 2) a listing of the associated hyperlink URLs for that page.
4. All non-characters except for spaces are stripped from each line of the link labels file, as a preprocessing for text-to-phoneme generator.
5. The label contents are sent through the text-to-phoneme generator and the results are saved to a temporary file.
6. HTML code is added for hiding the pronunciation information within the document source and for calling the icon and hypertext that appear at the top of the document to tell the user that "This is a speech-enabled document," thereby creating the speech-enabled HTML document.
7. The pronunciation file created by the text-to-phoneme generator is converted to a word models file, which the recognizer uses in finding the best match against the user's speech.
8. Information is stored to a table that matches the speech-enabled document with its corresponding word-models file.
9. Temporary files created by the script are removed.

This information usually needs to have a few minor changes made by hand to compensate for weaknesses in the text-to-phoneme generator. The current text-to-phoneme generator does not handle numbers, and occasionally does not correctly translate proper nouns or acronyms. Also, one can easily add multiple pronunciations to words by creating multiple listings for the same word (for example, one can say "S L A M" or "slam" to access the link for the SLAM system home page).

5.8 Updating the user's screen

The current version of SLAM uses "remote control of Mosaic," (NCSA, 1994c) a built-in feature of Mosaic 2.4, to update the user's screen. This feature works as follows:

1. Create a /tmp/Mosaic.pid file, where pid is the process id number of the form:

```
goto
{URL to goto}
```

2. Send the signal:

```
kill -USR1 pid
```

and the specified URL is retrieved from its remote location and loaded into the Mosaic browser.

The user's client receives the new URL from the remote recognizer, based on the user's speech and the current page being viewed by the user, at which point the updated client calls the remote control of Mosaic to update the user's screen. SLAM's use of "remote control of Mosaic" is shown in Appendix A.1.

5.9 Current status

At this writing, the initial implementation of SLAM has been completed. The remote-recognition system runs in near real-time on a Sun SPARCstation 20 workstation. There is currently a "global hotlist" of approximately 20 words that the user can access from any document during the SLAM session (personalized hotlists will soon be available feature with SLAM). There are also an increasing number of speech-enabled documents (currently over 100 generated, although modifications to the automatically-generated pronunciation models need to be made) that have hyperlinks that can be accessed by speech from within SLAM; these links can be accessed whether the text for these links is on-screen or

off-screen, as long as a speech-enabled document is currently loaded into the browser. These speech-enabled documents were created with a script similar to the pseudocode for the slam-enable script presented in Appendix A.2.

The current system uses the remote-recognition model discussed in chapter 4.5. The "speech-enabled" pages are compiled ahead of time and are referenced in a table that is stored at OGI.

The user can navigate the WWW as always with the mouse, or can use SLAM's speech facility to access either the "global hotlist" at any time or all links on the current page (as well as all items in the "global hotlist") when on a "speech-enabled" page. As explained in Appendix A.1, the user can access the speech facility of SLAM through the modified hotkey binding for "X." When the user presses "X" on the keyboard, the system prompts them to begin speaking the name of the hotlist item (or, for speech enabled pages, the link name). When they have finished speaking, they press the "X" key again, at which time an external program is called. Based on the current URL for the system and the user's speech, a four-line file is created of the form shown in Figure 5.2, composed of the length of the current URL, the length of the speechfile, as well as the current URL itself and the speechfile.

This four-line file is sent to the SLAM client described in Appendix A.4, which in turn sends the file to the remote recognition server described in Appendix A.3. The remote recognizer returns a value associated with the new URL value with which to display on the user's screen, and the user's machine uses remote control of Mosaic to update the user's browser.

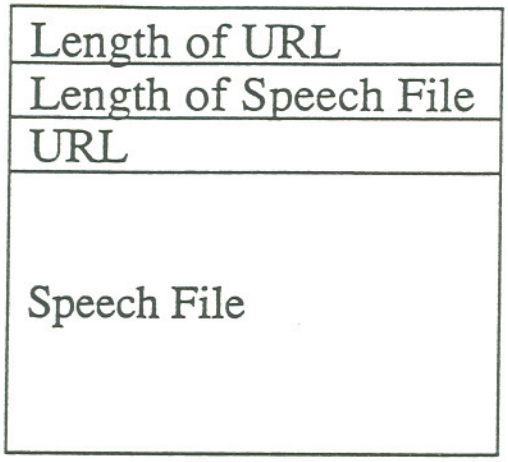

Figure 5.2: Input file for current SLAM demonstration system

5.10 Network rates for SLAM

Based on the current system (described in the previous section), the network performance of SLAM was measured based on 4 sample test files. These files were run from different machines from OGI, the University of Washington in Seattle, and North Carolina A&T in Greensboro, NC.

Although only a few trials were conducted, these tests seem to showed that for short files, SLAM's local version was only about a second faster than the remote version. For longer files, SLAM's local version was usually two or three seconds faster than the remote version. This might imply that SLAM would have an acceptable performance speed across these networks as implemented, and underscores the need for making the files to be transferred as small as possible from a network performance perspective.

Perceptual studies involving users should be done to determine how slow is "too slow" for such a system.

Data from these trials, as well as details about the data files, are found in Appendix B.

5.11 User tests

Issues regarding the development of SLAM became apparent during user testing of the sample system. One key comment from a user was that "I'll use it, but only if I can use it as fast or faster than I can use Mosaic." This seemed to reinforce the multimodal approach I have taken with building this system.

Users generally liked the demonstrations, although the poor recognition left them with the impression that this was merely a "toy" system, and left some with the idea that this is not a practical approach. An improved recognizer will be tried with the current system, and the speech-enabled pages will be modified for enhanced recognition, so I hope that these and other improvements to the system will convince the non-believers that this is a practical system.

Users showed tendencies of trying to stretch the scope of the utterances the system recognized, such as by unintentionally giving the wrong utterance when trying to access a hotlist item (such as by saying "weather" when the expected phrase was "Portland weather") or by adding additional words to the front or end of the utterance (such as by adding "show" to the front or "please" to the end of the given utterance). Additional word-spotting techniques within the recognizer and the ability to allow multiple pronunciations for a given utterance could help the system to handle such inputs.

One problem that became apparent with the system during user testing was that of switching between speech-enabled and non-speech-enabled pages. After using speech to access several speech-enabled links in a row, the tendency was to try to use speech to access the links on a non-speech-enabled page. An even worse problem was that when the icon which denotes a page as being speech-enabled is off-screen, the user can easily forget whether a given page is speech-enabled or not. One possible solution to the latter problem

would be to change the color or border of speech-enabled pages so that it is obvious throughout the document whether a page is speech-enabled or not. Another solution would be to find another paradigm for SLAM besides the current one based on speech-enabled pages.

One change made to SLAM on the basis of user feedback involved changing the status line so that the user had a better idea of the system state. With the original Mosaic, when a link was chosen, the system would display the message "Done sending HTTP request; waiting for response" after the request was made. However, because the current system provides no external feedback to the user about what the results of the recognizer are when using the speech facility of SLAM, this message was changed to "Done sending HTTP request; waiting for response from $hostname" (where $hostname is the remote hostname) so that users would better know whether the site associated with the correct link had been recognized.

In general, users wanted more feedback from the system, so that they could quickly know which link the recognizer had chosen, why the recognizer made the choice it did, and (if the recognizer made the wrong choice) how the user could avoid that mistake in the future. Perhaps a window external to Mosaic could be run simultaneously to provide this feedback, or perhaps this information could be accessed within the Mosaic window itself.

The most popular feature of SLAM's current system was the convenient method of accessing items from the hotlist, since for many cases this seemed to have a significant speed and convenience advantage over accessing visible links with speech or the mouse. Perhaps a version of SLAM should be created that focus only on allowing the user to universally access a dynamically changing speech hotlist.

Some users said that they would like to use the system but do not have easy access to

systems with microphone input. Fortunately, the number of systems with this feature is expected to continue growing, and lack of a microphone should soon stop being a limitation for potential users.

Virtually all users who tested the system expressed a need for better system recognition and said they would like to have unconstrained access to links and other features of Mosaic. Several of these people, however, said that they would gladly use a system similar to the current demonstration system until these requested improvements are made.

Chapter 6

Future Work

Despite the progress made with SLAM to date, much remains to be accomplished. As suggested at in sections 4 and 5, there are several areas in SLAM that could benefit from added functionality.

One area for improvement for SLAM is in improving the interface to better handle the speech aspects of the system, in particular expanding the number of components of the system that are accessible with speech and resolving recognition ambiguities between these components. Improvements could also be made with the speech recognizer, as well as improvements that could make the system run better on a variety of platforms. User studies should be done to better determine usage patterns for the WWW and how speech could better aid in accessing this information. Related projects that explore voice-only access to the WWW and that make greater use of the speech samples are natural extensions of the work with the SLAM system. Finally, this research can serve as the basis for future work into the difficult problems involved with spoken-language access to pictures, icons, and unconstrained navigation of the WWW.

6.1 Improved speech access

One major area for improvement of the SLAM system would be finding ways in which the user can best access the speech features of the system, through enhanced use of speech for accessing commands, adding separate speech hotlist structures and international language features, and allowing different ways to handle scope and ambiguity issues associated with speech interfaces to hypermedia.

6.1.1 SLAM command access

The current version of SLAM does not allow for speech recognition of Mosaic commands, although this would be a natural extension of the system and would be straightforward to implement. It may be necessary, however, to have some type of prefix word, such as "Mosaic" or "command" prefacing the actual command (such as "back" or "home"). Having a prefix word serves two purposes. First, it serves to lengthen the speech, which aids in disambiguation during recognition. Second, it helps to avoid confusion between the commands and the link labels that are pronounced the same way. Before spoken command access to SLAM is added, however, an improved recognizer should be put in place to help to avoid spiral degradation errors (Oviatt, 1992) that will occur when commands and their successive attempts to be corrected with speech are misrecognized.

Another important direction for the development of the SLAM system is to enable use of flexible, natural language commands so that the system might take greater advantage of the strengths of the spoken language modality. Eventually, it would be nice to have the user be able to create custom commands such as "back three pages" or "show parent directory" with speech. Such a system should have a recognizer to deal with grammars, although the current system does not.

6.1.2 Current screen only or document-wide access

A better system than the current one, which only allows for document-wide access to links, would allow the user to choose their manner of accessing links (through some type of set-up file or command-line option): document-wide or simply being restricted to the current page. As discussed in chapter 5.4, document-wide access allows the user to take advantage of the ability of speech to refer to non-visible items, while screen-wide access could improve the recognition rate by constraining the number of hotlink choices available to the recognizer.

6.1.3 SLAM speech hotlists

SLAM could benefit from a separate speech-related hotlist instead of merely using the existing Mosaic hotlist. The user may wish to keep separate the links they want to use with speech and the links they discover while browsing. The name, which is stored in the title and therefore saved in the hotlist, is often undescriptive, unpronounceable, or both. One well-known problem with earlier versions of Mosaic hotlists is that there is no built-in feature for hierarchically saving items into the hotlist. If such a hierarchical structure existed, perhaps a standard directory called "SLAM" or "Speech" could be created for which to store speech-related hotlinks. Such hotlist structures are now available in Mosaic 2.5 (NCSA, 1994d).

6.1.4 Handling ambiguity in alternative outputs

The current system makes no attempt to handle ambiguity or confusions among spoken inputs.

There are several ways this ambiguity could be handled. The system could flag identical words (the "here" in multiple "click here" links, for example), homophones (such as,

in this case, "hear") or similar sounding words (like "ear") as the system is creating the pronunciation models. The system could tell the page's author of the confusion, and perhaps direct the author towards how to correct the problem. If the author did not correct the problem, then when the user accessed one of the ambiguous words, perhaps a dialogue (either through a dialogue box or perhaps thorough voice input/output) could be initiated by the system to ask the user to be more specific about which item was intended. Such a system could handle confusion between links, commands, hotlist items, and combinations of these.

6.1.5 International language aspects of SLAM

As part of the World-Wide Web, SLAM's expected usage crosses all international boundaries. Therefore, methods need to be put into place to better allow SLAM to handle non-English text and speech.

One problem not directly addressed by our system is that of non-English hypertext labels. As this is the *World-Wide* Web, links can appear in a variety of languages and not all of the sounds from these languages can be mapped to English phonemes. A short-term solution to this problem would be to map these sounds to their closest English equivalents or to use the flexibility of speech interfaces to map these link names to words that *do* have corresponding English equivalents, if possible. A longer-term solution would be use of non-English language corpora to train the recognizer on non-English as well as English phonemes. SLAM may even be used to aid in collecting this data as international users send their speech to a central recognizer to perform recognition.

For one thing, the speech recognizer will have additional difficulty with non-native English speakers, especially if they are not speaking English. The current recognizer is only trained on 39 American English phonemes, so authors of speech-enabled documents

for this system know to create only American English documents for this system. Perhaps international speech could be used for training future SLAM recognizers (or perhaps recognizers devoted to particular languages could be used in different locations).

Another international language issue involved with SLAM is that the text-to-speech generator outputs the speech label phonemes in OGIbet representation. Perhaps this could be replaced by an alphabet like Worldbet with a larger set of international pronunciation symbols.

6.2 Other interface improvements

SLAM's interface during the use of the remote-recognition model can be improved in two major ways from the currently-implemented model.

One improvement would be to enable the user to cancel the connection to the remote recognizer by clicking on the spinning globe in the upper right of the Mosaic window, which is the same way the user cancels the connection to other remote sites when using Mosaic.

Another improvement for the current version would be to have the system status line above the bottom row of buttons on the Mosaic interface give better feedback to the user, such as "unable to connect to speech server" and "X bytes out of Y bytes received," which is similar to the way Mosaic handles other data transfers.

6.3 Improvements to the recognition

SLAM's recognition component could be improved in a number of ways. One area that could use a great deal of improvement is the creation and storage of word models and pronunciation models. There is a significant trade-off between speed of generation and

accuracy in the these models, and ideally we would want models that could be generated quickly and accurately.

More study should be done to determine differences in quality vs. speed-of-generation between the current SLAM method of using fast-but-inaccurate methods (which can provide poor representations of link labels at the rate of roughly 100 per second) instead of the combination of Moby and Orator which SLAM currently uses, which is reasonably accurate but does link label conversions at a rate much slower than real-time. Perhaps expanding a look-up table for the fast-but-inaccurate model would provide the needed compromise for pronunciation generation.

Additional options should also be looked at in the areas of having multiple recognizers, either at the local site to handle requests to different pages for greater accuracy, or at multiple remote sites to allow for faster SLAM accesses, to handle special types of SLAM requests, and to serve as back-ups in case of network failures at other site.

6.4 Implementation improvements

SLAM's implementation can be improved in several ways. One major area for expansion of this project will be making SLAM available on a variety of platforms. Certainly the platforms on which the original Mosaic itself became popular (Unix, Macintosh, and PC-Windows) should also be the major platforms on which SLAM should focus. Just as Mosaic has spread to nearly all major platforms as the software product became popular, I hope that use of SLAM will also become similarly widespread.

Platform independence is another continuing goal for SLAM. Most of the current system is quite modular and could be made to operate on variety of platforms as well as with a variety of browsers with only a few changes. Parts of the current system that could be made more independent and less system-specific include:

(a) the use of htmlchek's link parser; this parser was chosen because it performed the needed task of extracting link labels and URLs from documents well. However, it also has a great deal of functionality unneeded for the SLAM project and, since it's written in Perl, may be difficult to port to other platforms, and therefore should probably be replaced in SLAM with a fast, task-specific parser written in a more easily-portable language;

(b) the use of url_get, a third-party program written in Perl for getting the contents of WWW page from outside of Mosaic; the equivalent to this program will need to be found on other platforms, or the use of url_get will need to be replaced by gathering this information from within Mosaic itself; and

(c) the current method of gathering documents with "remote control of Mosaic" from Mosaic 2.4 is being discouraged by NCSA, who encourage the use of the newly-implemented (for release 2.5 and beyond) Common Client Interface (CCI) (NCSA, 1995) as a sockets-based way of exchanging information between Mosaic and external programs. Because it would be possible to run SLAM entirely outside of the main Mosaic program (see Appendix A) if the external SLAM system could have access to (1) the current process id of the window to change and (2) the current URL being viewed in that window, and since CCI seems to allow this information to be passed to external programs, then it would seem that the use of CCI could enhance future versions of SLAM.

A further enhancement of the implementation would extend the types of information passed to the remote recognizer. Instead of merely passing the current URL and the user's speech from the client to the server, as the current version of SLAM does, future versions of the SLAM client could accept a variety of parameters including the contents of the

user's hotlist, information about what links are currently onscreen, requests for specific SLAM servers, the window history, the microphone type, and other user preferences.

Some effort should be made to find ways to compress the types of files that are being sent back and forth across the network. The speech files that are being sent could be compressed, as could the text files, which could significantly cut down on network overhead.

Finally, the SLAM server needs a better method of handling multiple calls. This issue took a back seat to other implementation issues during the development of the initial SLAM system but will become important if the SLAM system becomes popular. Better methods of handling multiple simultaneous accesses, both in terms of server performance and their effects on the current single-recognizer system, should be explored.

6.5 User studies

A valuable study could be done to see how (and how often) users would use the speech facilities of a speech-enabled Mosaic browser. Such a study could be devised by either fully enabling a Mosaic system (which we have made progress towards, but not completed) or could be done as a "wizard" study whereby the user and wizard both see the same copy of the user's screen; when the user wishes to use speech, the wizard is the one who actually changes the screen for them. There are versions of Mosaic that allow multiple users to see the same screen; such a modified version would be useful for such an experiment. Such an experiment could be used to measure the frequency of use of speech vs. the mouse for commands, items on/off-screen, hotlist items, and window history. Similar experiments could also determine required levels of recognition accuracy and speed of recognition necessary to be an acceptable option for such a system.

Another important area for investigation is finding the best method for making use of the ability to have multiple pronunciations for links with SLAM. Investigations could be

made to determine when users would choose alternate pronunciations: for unknown words, for words with multiple possible pronunciations, or for convenience for long or common phrases? Studies of the word usage patterns for Web users would provide insight into how to best implement a system that allows for multiple pronunciations of link names.

6.6 Making greater use of the user's speech

The current implementation of SLAM discards the user's speech after processing by the recognition server. One thing that could be done, with the user's permission, would be to use this speech for other purposes, such as for training a general-purpose interface to speech- passing applications on the Internet such as voice mail, filling out on-line surveys by voice, and for speaker identification data as an additional means of security on the WWW.

The user's speech could also be used for speech research for such applications as speech corpora data collection (and could be especially valuable for collecting international speech data) and as training data for some future version of SLAM which could value from added accuracy from vocabulary- or speaker-dependent pages.

6.7 Speech-only access to the WWW

One direction to further pursue with this research is in creating a speech-only browser for the World-Wide Web, for users who do not have a screen to use (such as when one is on the telephone), or for disabled users who *cannot* view a screen, and yet still wish to access the variety of information available on the Web.

Such a system would need synthesized-voice output from the system, as well as allow voice input. A useful part of such a system would involve having a means of summarizing

WWW documents for users (Raman, 1995), so that long spoken utterances from the system could be replaced with statements like "The following document has one main title, two subtitles, and five paragraphs."

6.8 Speech access to hot icons, imagemaps, and text-entry forms

Currently SLAM does not permit selection of highlighted pictures and icons, although with Mosaic you can click on these items to bring up other WWW pages. One solution to this limitation of SLAM would be to number the highlighted icons so that each item receives a known, unique label. Another solution would involve using the filename relating to the icon or picture, as well as labels within the HTML code to give a descriptive name to the image, it may be possible to specify these images in future versions of SLAM. A further challenge in this area will be the selection of parts of items known as imagemaps, which in Mosaic call different WWW pages depending on *where* within the image one clicks. This would seem to be a very difficult task to accomplish in general with speech alone and may be one task that is better restricted to multimodal or other systems that use mouse-based input unless some way could be found to label the different regions.

Speech access to Mosaic's text-entry forms is another difficult problem for the SLAM system. The keyboard is probably a superior interface for this task, although a spoken letter recognition might be one solution to this problem.

6.9 Unconstrained multimodal access to the WWW

An eventual goal of this system is to provide multimodal access to any document on the WWW. In one sense, this is accomplished already, by being able to access hotlists with speech on any WWW document. With faster word modeling and more flexible and accu-

rate recognizers, future systems will allow users to access the full range of Mosaic capabilities and many features not yet imagined through WWW browsers like Mosaic.

Bibliography

Arbash, V. (1995). Personal communication, April, 1995.

Arons, B. Hyperspeech: Navigating in speech-only hypermedia. (1991). *ACM Conference on Hypertext: Hypertext '91 Proceedings*, San Antonio, Texas, December 15-18, 1991, p. 133-146, Baltimore: ACM Press.

CERN (European Laboratory for Particle Physics) (1994). "World-Wide Web Home", URL: http://info.cern.ch/, (undated).

Churchyard, H. (1995). "htmlchek HTML syntax and cross-reference checker version 4.1, February 20 1995 -- menu," URL: http://uts.cc.utexas.edu/~churchh/htmlchek.html, February 20, 1995.

Cohen, P. (1992). The role of natural language in a multimodal interface, *Proceedings of the ACM Symposium on User Interface Software and Technology*, Monterey, California, November 15-18, 1992, p. 143-149, Baltimore : ACM Press.

Cole, R., M. Fanty, Y. Muthusamy and M. Gopalakrishnan (1990). Speaker-independent recognition of spoken English letters, *Proceedings of the International Joint Conference on Neural Networks*, San Diego, CA, June 1990, v.2, p. 45-51, Hillsdale, NJ: Lawrence Erlbaum Associates.

Cole, R., Hirschman, L., et al. (1995). The challenge of spoken language systems: Research directions for the Nineties. *IEEE Transactions on Speech and Audio Processing*, 3(1), p. 1-20.

Cole, R., Novick, D., Burnett, D., Hansen, B., Sutton, S. & Fanty, M. (1994). Towards automatic collection of the U.S. Census, *Proceedings of the 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 19-22, 1994, Adelaide Convention Centre, Adelaide, South Australia, v.1, p. 93-96, Piscataway, NJ : Institute of Electrical and Electronics Engineers.

Conte, E. (1994). "A Basic HTML Style Guide: Readability", URL: http://heasarc.gsfc.nasa.gov/0/docs/heasarc/Style_Guide/readability.html, June 22, 1994.

Domel, P. (1994). Webmap - A graphical hypertext navigation tool. *Advance Proceedings of Mosaic and the Web: the Second International WWW Conference '94*, p. 785-798.

Fanty, M., Schmid, P., & Cole, R. (1993). City name recognition over the telephone, *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 27-30, 1993, Minneapolis Convention Center, Minneapolis, Minnesota, Minneapolis, April 1993, v.1, p. 549-552, Piscataway, NJ : Institute of Electrical and Electronics Engineers.

Goddeau, D. (1995). "Recent Publications," URL: http://sls-www.lcs.mit.edu/~dg/publications. html, February 1, 1995.

Goddeau, D., Brill, E., Glass, J., Pao, C., Phillips, M., Polifroni, J., Seneff, S., and Zue, V. (1994). GALAXY: A human-language interface to on-line travel information, *Proceedings of the International Conference on Spoken Language Processing*, Yokohama, Japan, Sept. 1994, p. 707-710, The Acoustical Society of Japan.

Hemphill, C. (1995). Personal communication, February, 1995.

Hermansky, H. (1990). "Perceptual Linear Predictive (PLP) Analysis of Speech," *J. Acoust. Soc. Am.*, v. 87, no 4., p. 1738-1751.

Hermanky, H., Morgan, N., & Hirsch, H. (1994). Recognition of Speech in Additive and Convolutional Noise Based on RASTA Spectral Processing, *Proceedings of the 1994 IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 19-22, 1994, Adelaide Convention Centre, Adelaide, South Australia, v.1, p. 421-424, Piscataway, NJ *:* Institute of Electrical and Electronics Engineers.

Lefebvre, P., Duncan, G., & Poirier, F. (1993). Speaking with computers: A multimodal approach, *Proceedings of EuroSpeech'93: Third European Conference on Speech Communication and Technology*, Berlin, p. 1665-1668.

Lunati, J.-M., & Rudnicky, A. (1990). The design of a spoken language interface, *Proceedings of the DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, June 1990, p. 225-229.

Lund, J. (1994). "Perl Program for Retrieving Web Documents." URL: http:// www.utexas.edu/~zippy/url_get.html, September 28, 1994.

Mauldin, M. (1994). "Frequency and Length of half a million WWW Anchors." URL: http://fuzine.mt.cs.cmu.edu/mlm/links.html, June 24, 1994.

Moran, T., & Anderson, R. (1990). The workaday world as a paradigm for CSCW design, *Proceedings of the Conference on Computer-Supported Cooperative Work, October 7-10, 1990 Los Angeles, CA*, p. 381-393, New York, NY : The Association for Computing Machinery.

Muller, M. & Daniel, J. (1990). Toward a definition of voice documents, *Proceedings of Conference on Office Information Systems*, Cambridge, MA, 25-27 April 1990. SIGOIS Bulletin, 1990, v. 11, (no.2-3): p. 174-183.

NCSA (National Center for Supercomputing Applications) (1994a). "A Beginner's Guide to HTML," URL: http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html (undated).

NCSA (National Center for Supercomputing Applications) (1994b). "NCSA Mosaic Home Page," URL: http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html, (undated).

NCSA (National Center for Supercomputing Applications) (1994c). "Using Mosaic by Remote Control," URL: http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/remote-control.html, December 14, 1994.

NCSA (National Center for Supercomputing Applications) (1994d). "Help on Nested Hotlists," URL: http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/help-on-nested-hotlists.html, August 23, 1994.

NCSA (National Center for Supercomputing Applications) (1995). "NCSA Mosaic Common Client Interface (CCI 1.0)," URL: http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/CCI/cci-spec.html, January 8, 1995.

National Coordination Office for HPCC (1994). "Information infrastructure technology and applications." Technical report, Office of Science and Technology Policy, Executive Office of the President, Washington, DC, February 1994.

Nigay, L., & Coutaz, J. (1993). A design space for multimodal systems: concurrent processing and data fusion, *Proceedings of InterCHI'93*, Amsterdam, April, 1993, p. 172-178, New York : Association for Computing Machinery.

Olary, Bob (1995). "Hartsfield School's Home page," URL: http://www.hartsfield.leon.k12.fl.us/, April 12, 1995.

Oviatt, S. (1992). Pen/voice: Complementary multimodal communication, In *Proceedings of Speech Tech'92*, New York, February 1992, p. 238-241.

Oviatt, S., & Olsen, E. (1994). Integration themes in multimodal human-computer interaction, *Proceedings of Intl. Conference on Speech and Language Processing '94*, Yokohama, p. 551-554, The Acoustical Society of Japan.

Paciello, M. (1995). Personal communication, January, 1995.

Raman, T. (1995). Personal communication, March, 1995.

Resnick, P. (1990). The rainbow pages: Building community with voice technology, *Proceedings of DIAC-90, Directions and Implications of Advanced Computing*, Boston, MA, July 1990, p. 2-13, Palo Alto CA: Computer Professionals for Social Responsibility.

Rudnicky, A. (1993). Factors affecting choice of speech over keyboard and mouse in a simple data-retrieval task, *Proceedings of EuroSpeech'93: Third European Conference on Speech Communication and Technology*, Berlin, p. 2161-2164.

Saito, Takeshi (1995). "Faculty of Dentistry Home Page," URL: http://133.5.231.40/, (undated).

Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages, *IEEE Computer*, 16(8), p. 57-69.

Stephenson, K. (1994). Subject: Re: NCSA Mosaic Mac 2.0 Alpha1. Article <2tcnr8$lhh@vixen.cso.uiuc.edu> in comp.infosystems.www, June 12, 1994.

Stock, O. (1991). Natural language and exploration of an information space: The AlFresco interactive system, *Proceedings of the Twelfth International Conference on Artificial Intelligence*, Darling Harbour, Sydney, Australia, 24-30 August 1991, p. 972-978, San Mateo, Calif. : M. Kaufmann Publishers.

Wallach, D. (1994). "WWW Size," World-Wide Web page summarizing NSFNet statistics collected at nic.merit.edu, URL: http://www.cs.princeton.edu/grad/dwallach/www-talk/size.html, March 22, 1994.

Whalen, T., & Patrick, A. (1989). Conversational hypertext: Information access through natural language dialogues with computers, *Proceedings of the Conference on Human Factors in Computing Systems: CHI '89*, Austin, Texas, April 30-May 4, 1989, p. 289-292, New York, NY : ACM Press.

Appendix A: Code and pseudocode from the project

The following section contains code and pseudocode from the SLAM project, specifically:

- Pseudocode for changes to Mosaic's gui.c and my slam_navigate() program
- Pseudocode for the slam_enable() script
- Code for slamserver.c, which controls SLAM's remote-recognition system
- Code for slamclient.c, which passes data to the remote-recognition system

A.1. Pseudocode based on modifications to Mosaic's gui.c

This code comes from:

/projects/interactive/A/media/slam/system/mosaic-mods/src/gui.c

and contains the "main loop" for the program which gets called every time the user presses 'x' on the keyboard.

Within Mosaic's "hotkeys" definition:

```
set TOGGLE = 0;
if "X" pressed
   if (TOGGLE == 0)
      {
      set TOGGLE=1;
      message("Recording speech;press 'X' when finished talk-
   ing");
      display_watch_icon;
      start_recording_speech > $SPEECHFILE;
```

```
    }
    else /* (TOGGLE == 1) */
    {
    set TOGGLE=0;
    stop_recording_speech;
    message("Processing files");
    slam_navigate($PROCESS_ID $CURRENT_URL $SPEECHFILE);
    }
```

where **slam_navigate()** is a program external to Mosaic defined as follows:

```
slam_navigate($PROCESS_ID $CURRENT_URL $SPEECHFILE)
  {
  set URL_LEN = length($CURRENT_URL);
  set SPEECH_LEN = length($SPEECHFILE);
```

```
/* Create the 4-line input file for the slamclient of the
    specified form */
echo "$URL_LEN \n $SPEECH_LEN \n $CURRENT_URL \n
      $SPEECHFILE" > $SLAMFILE
/* Use "slamclient," the compiled version of slamclient.c,
    to find the resulting URL */
set NEW_URL = `slamclient < $SLAMFILE`
/* Use Remote Control of Mosaic to update the user's
screen */
echo "goto \n $NEW_URL" >! /tmp/Mosaic.$PROCESS_ID;
kill -USR1 $PROCESS_ID;
}
```

A.2. Code for SLAM-Enable (speech-enabled document conversion script)

Pseudocode for the SLAM-enable script, used for creating speech-enabled documents):

```
url_get URL --> URL_CONTENTS_PLUS_HEADER

strip_head URL_CONTENTS_PLUS_HEADER --> URL_CONTENTS

xtraclnk URL_CONTENTS --> URLFILE LINKFILE

strip_non_char LINKFILE --> LINKS_NO_CHAR

make_phoneme LINKS_NO_CHAR   --> PRONUNCIATION_MODEL

create_sep URL_CONTENTS --> SEP_URL SEP

collate URLFILE URL_CONTENTS --> COLLATED_FILE

parse_dict COLLATED_FILE --> WORD_MODEL WORD_MODEL_NAME

update_model_table SEP_URL WORD_MODEL_NAME -->
    NEW_MODEL_TABLE
```

Functions of SLAM-Enable:

url_get: Used to retrieve contents of http document including header

Input: URL to get contents + header

Output: Contents of http document + header

strip_head: Removes header from http document file

Input: Contents of http document + header

Output: Contest of http document

xtraclnk: To extract link labels and URLs from http documents

Input: Contents of http document

Output: 2 files, one containing the URLs for the http document, one at a time; and one containing the link labels for the http document, one at a time

strip_non_char: Non-characters removed from label file; or for hot icons with no label, the label line is left empty

Input: Link label file with non-characters

Output: Link label file without non-characters

make_phoneme: Converts each line of text in the input file to phonemes

Input: Input text file, with one link label per line

Output: Link label file represented by phonemes (pronunciation model)

create_SEP: Creates a Speech-Enabled Page by combining the pronunciation model and contents of http document

Input: Contents of http document + pronunciation model

Output: Speech-Enabled Page with the following information at the top of it:

```
<SPEECH =
Pronunciation for link 1
{remaining pronunciations}
>
Speech-Enabled Icon and link 1
{remaining link labels}
```

collate: Creates a file of the following triplets: {URL, pronunciation, blank line}

Input: pronunciations, URLFILE

Output: collated file of triplets

parse_dict: Creates word model file for recognition

Input: collated file of triplets

Output: word model file

update_model_table: Updates the word model table

Input: URL to Speech-Enabled Page and the name of the word model

Output: word model table updated with the new entry

A.3. Code for the SLAM server

This code comes from:

/projects/interactive/A/media/slam/system/mosaic-mods/src/dh-support/networking/slam-server.c

and is being used to handle the remote-recognition for SLAM.

```
/*****************************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SERV_TCP_PORT 5555
#define SERV_HOST_ADDR "129.95.44.43"

#define recog_dir "/projects/interactive/A/media/slam/system/mosaic-mods/dh-support/
    recognizer"
#define combine_hotl_command "/projects/interactive/A/media/slam/system/mosaic-
    mods/dh-support/networking/combine_hotl.sct"
#define word_model_dir "/projects/www/SLAM/models"
```

```
#define MAXLINE 256


char *pname;

/*************************************************************/

new_getline(char *line, FILE *fp)
{
char c;
int count;

      count = 0;
      while ((c = fgetc(fp)) != EOF)
         { if (c == '\n')
            { line[count] = '\0';
              return strlen(line);
            }
          else
            {
            line[count] = c;
            count++;
            }
         };
      return 0;
}

/*************************************************************/
```

```
/* from page 281:

The following function is used by the connection-oriented servers.  It is an
echo function that reads a line from the stream socket and writes it back to
the socket.

*/


/* Specifically, this is the routine for the server which reads the
socket for the input file, and then parses the file and writes the
results to files.  The file is of the form:

  URL length (in bytes)
  speech file length (in bytes)
  URL
  speechfile
 */

sockread(sockfd, sockbuf)
int   sockfd;
char  *sockbuf;
{
int n;
int x;                  /* Keeps track of socket buffer */
int ptr;
int amountread;         /* Keeps track of amount of speechfile read */
```

```
char    speechlength[20],  /* Length of URL, speech files */
        urllength[20];

int     speechlen,       /* Same values as prev, but int's not strings */
        urllen;

char    urlfile[20],     /* Filename for URL, speech files */
        speechfile[20];

char  *sockchar;
int urlfd, speechfd;     /* File Descriptors for URL, Speech files */

char command[256];



ptr = 0;                 /* Reset pointer */

strcpy(urlfile, "temp.url");
strcpy(speechfile,    "/projects/interactive/A/media/slam/system/mosaic-mods/dh-support/
    recognizer/foo.mu");

/* Creating fd's for the URL and speech files:
  From page 172 of K & R */

/if ((speechfd = creat(speechfile, 0)) == -1)
  printf("Can't open %s\n", speechfile);



x = 0;
```

```
while ((read(sockfd, sockbuf, 1)) && (sockbuf[0] != '\n'))
        {
         urllength[x] = sockbuf[0];
         x++;
         sockbuf++;
        }

x = 0;
while ((read(sockfd, sockbuf, 1)) && (sockbuf[0] != '\n'))
        {
         speechlength[x] = sockbuf[0];
         x++;
         sockbuf++;
        }

urllen = atoi(urllength);
speechlen = atoi(speechlength);

/* Read URL value from the socket and print the contents to a file */
n = read(sockfd, sockbuf, urllen);

/* Look through the word model table for the correct word model to use */
read_table(sockbuf);

/* Read speech file from the socket and print the contents to a file */

/* Read file in at 512 byte chunks */
ptr = urllen;
```

```
amountread = 0;

while ((speechlen - amountread) >= 512)
  {
   n = read(sockfd,sockbuf + ptr, 512);
   ptr += n;
   amountread += n;
  }

/* Final read, to fill in missing (< 512 byte) piece */
n = read(sockfd,sockbuf + ptr, (speechlen - amountread));


write(speechfd, sockbuf + urllen, speechlen);

if (close(speechfd) != 0)
      fprintf(stderr, "Error in closing files\n");

}

/*********************************************************/

/*====================
readtable: Created by David House, March 9, 1995

Reads in 2 lines at a time from the the "word model table."

The first line gets compared against the "URL" value passed from the client.
If there's a match, the word model file relating to the second line gets
```

```
passed to recognizer directory to be used as *the* word model for
recognizing the speech.
=======================*/

read_table(sockbuf)
char *sockbuf;
{
FILE *tablefp;
int done;
char line1[MAXLINE], line2[MAXLINE], command[MAXLINE];

/* Open the specified models table */
if ((tablefp = fopen("/projects/www/SLAM/models/models_table", "r")) == NULL)
{
  printf("Can't open Models Table\n");
}

done = 0;

/* Look at the word model table file and find the appropriate match */
while((new_getline(line1, tablefp) != 0) && (done == 0))

  { new_getline(line2, tablefp);

    if (strcmp(sockbuf,line1) == 0)

    /* Copy word model file to recognition directory to be seen as *the*
         word model file */
    { /* sprintf(command,"cp %s/%s >! %s/word_models", word_model_dir, line2,
```

```
            recog_dir); */
        sprintf(command,"%s %s",combine_hotl_command, line2);
        system(command);
        done = 1;
       }
   }
if (done == 0)
/* If no matching URL was found, use the default hotlist word model */
          {  sprintf(command,"cp  %s/dh-hotl_word_model  >!  %s/word_models",
    word_model_dir, recog_dir);
        system(command);
  }


fclose(tablefp);
}

/*********************************************/

main(argc, argv)
int argc;
char *argv[];
{
intsockfd, newsockfd, clilen, childpid;
struct sockaddr_incli_addr, serv_addr;
      char sockbuf[51200];
      char return_val[512];
      FILE *resulturlfp;

pname = argv[0];
```

```
if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
{ printf("server: can't open stream socket\n");
        exit(1);
    }

  /* Bind our local address so that the client can send to us */

  bzero((char *) &serv_addr, sizeof(serv_addr));
  serv_addr.sin_family= AF_INET;
  serv_addr.sin_addr.s_addr  = htonl(INADDR_ANY);
  serv_addr.sin_port         = htons(SERV_TCP_PORT);

  if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
     {
        printf("server: can't bind local address\n");
        exit(1);
     }


  listen(sockfd, 5);

  for (;;) {
     clilen = sizeof(cli_addr);
     newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);

     if (newsockfd < 0)
       {printf("server: accept error\n");
        exit(1);
```

```
        }

        if ((childpid = fork()) < 0)
          printf("server: fork error\n");
        else if (childpid == 0)  {
            close(sockfd);

        /* Read a value from the client, and set up the word modeling */
          sockread(newsockfd, &sockbuf);

          system("/projects/interactive/A/media/slam/system/mosaic-mods/dh-support/recog/
      rsh.recog > /tmp/tempurl");
          resulturlfp = fopen("/tmp/tempurl","r");
          fgets(return_val,512,resulturlfp);

          unlink("/tmp/tempurl");
          write(newsockfd, return_val, strlen(return_val) + 1);

            exit(0);
        }

        kill(childpid, SIGKILL);
        waitpid(childpid, NULL, 0);
        close(newsockfd);
    }

}
```

A.4. Code for the SLAM server client

This code comes from:

/projects/interactive/A/media/slam/system/mosaic-mods/src/dh-support/networking/

slamclient.c

and is used as the current client for the SLAM remote-recognition server.

```
/*****************************************************************/

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define SERV_TCP_PORT 5555
#define SERV_HOST_ADDR "129.95.44.43"
```

```
/* Orginally written by Ken Maupin; Read in a file of data from STDIN
for the client to pass to the server */

int fileread(int filebuf)
{
int n;
int ptr;

ptr = 0;

while (n = read(0,filebuf + ptr, 512) )
  {
/*     printf("Fileread: %s\n", filebuf + ptr); */
    ptr += n;
  }
return ptr;
}


/* The following is based on the TCP client code given in Stevens'
  "Unix Network Programming */

main()
{
  int sockfd;
  int n;
  struct sockaddr_inserv_addr;
  char filebuf[51200];
  char *returned_val;
```

```
int filesize;
char *sockbuf;

/* Fill "serv_addr" with address of server we want to send to */

bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family        = AF_INET;
serv_addr.sin_addr.s_addr   = inet_addr(SERV_HOST_ADDR);
serv_addr.sin_port          = htons(SERV_TCP_PORT);

/* open socket */

if ((sockfd = socket(AF_INET,  SOCK_STREAM, 0)) < 0)
    { printf("Client: can't open stream socket");
      exit(0);
    }

/* connect to server */

if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
          printf("client: can't connect to server\n");

/* Instead of using the str_cli function given in Stevens, we're writing
  our own.  */

/* Read in a file from STDIN */
filesize = fileread((int) &filebuf);

/* Write the contents to the socket */
```

```
  write(sockfd, &filebuf, filesize);

  /* Read the result into the returned value */
  while (( n = read(sockfd, &returned_val, 1)) > 0)
     write(1,&returned_val,n);

  close(sockfd);
  exit(0);
}
```

Appendix B

Network Tests of SLAM

SLAM was run from the following machines at sites at the University of Washington, the Oregon Graduate Institute, and North Carolina A & T.

An attempt was made to run the program at times of both high and low network activity.

The program was timed for completing each of four sample test files. These files can be described as follows:

Testfile #1:

Filesize: 9288

Number of links on page: 21

Current URL: http://www/~dhouse/

Speech: "Weather" [9248]

Result:gopher://wx.atmos.uiuc.edu/00/States/Oregon/
Metro%20Area%20Extended%20Fcsts%20%28Portland%29

Testfile #2:

Filesize: 11337

Number of links on page: 21

Current URL: http://www/~dhouse/

Speech: "Portland, Oregon" [11296]

Result: http://www.ee.pdx.edu/depts/fpa/html/portlandtour.html

Testfile #3:

Filesize: 28745

Number of links on page: 21

Current URL: http://www/~dhouse/

Speech: "Go to the S.L.A.M. home page" [28704]

Result: http://www.cse.ogi.edu/SLAM/

Testfile #4:

Filesize: 7269

Number of links on page: 7

Current URL: http://www.cse.ogi.edu/SLAM/available-pages/orgtest-SEP.html

Speech: "Movies" [7200]

http://www.msstate.edu/Movies/

System: u.washington.edu

		Test Run #				
Machine	Arch	1	2	3	4	Date
stein	DECmips	2.1	2.3	7.3	1.8	Sat, 25 Mar 95 17:27
alfred	Alpha	2.1	3.6	6.3	1.7	Sat, 25 Mar 95 17:46
mead	AIX	2.4	2.6	7.1	1.9	Sat, 25 Mar 95 17:54
stein	DECmips	2.3	2.4	5.1	1.8	Sat, 25 Mar 95 18:02
hardy	DECmips	2.3	2.4	4.7	1.9	Sat, 25 Mar 95 18:11
stein	DECmips	2.5	2.7	4.4	2.0	Tue, 28 Mar 95 12:30
alfred	Alpha	2.5	2.6	6.3	2.7	Tue, 28 Mar 95 12:43
mead	AIX	2.4	2.5	9.8	4.0	Tue, 28 Mar 95 12:53
hardy	DECmips	4.2	4.5	5.3	2.6	Tue, 28 Mar 95 13:08

Test Run #	1	2	3	4
Range	[2.1-4.2]	[2.3-4.5]	[4.4-9.8]	[1.7-4.0]
Median	2.4	2.6	6.3	1.9
Average	2.5	2.8	6.8	2.3

System: cse.ogi.edu

		Test Run #				
Machine	Arch	1	2	3	4	Date
calvin	DECmips	2.0	2.2	3.4	1.5	Wed, 29 Mar 95 12:17
sail	SunOS	6.4	2.3	3.4	1.6	Sun, 9 Apr 95 23:14
lateen	SunOS	4.3	2.2	3.3	1.6	Fri, 21 Apr 95 17:15
lateen	SunOS	4.0	2.1	3.4	1.8	Fri, 21 Apr 95 18:50
sail	SunOS	2.5	2.4	3.6	1.7	Fri, 21 Apr 95 18:54
calvin	DECmips	4.1	2.2	3.5	1.5	Fri, 21 Apr 95 19:14
lisa	Alpha	4.0	2.2	3.4	1.6	Fri, 21 Apr 95 19:59
calvin	DECmips	4.3	2.2	3.6	1.5	Sat, 22 Apr 95 16:02
lateen	SunOS	2.2	2.0	3.3	1.5	Sat, 22 Apr 95 16:11
sail	SunOS	2.4	2.3	3.8	1.7	Sat, 22 Apr 95 16:17
lisa	Alpha	2.4	2.3	3.8	1.7	Sat, 22 Apr 95 16:17

Test Run #	1	2	3	4
Range	[2.0-6.4]	[2.0-2.4]	[3.3-3.8]	[1.5-1.8]
Median	4.0	2.2	3.4	1.6
Average	3.5	2.2	3.5	1.6

System: ncat.edu

Machine	Arch	Test Run # 1	2	3	4	Date [EDT]
mercury	DECmips	3.0	5.0	5.0	2.9	Thu, 6 Apr 1995 23:29
mercury	DECmips	3.0	4.5	5.9	2.4	Fri, 7 Apr 1995 00:51
mercury	DECmips	3.6	3.1	4.9	2.4	Fri, 21 Apr 1995 22:18
mercury	DECmips	5.0	3.5	5.6	2.9	Sat, 22 Apr 1995 19:42
mercury	DECmips	5.6	5.3	7.3	2.6	Thu, 27 Apr 1995 18:05
mercury	DECmips	5.8	3.2	5.6	2.4	Mon, 1 May 1995 12:41
mercury	DECmips	6.1	3.3	5.8	2.7	Tue, 2 May 1995 14:06
mercury	DECmips	7.2	7.5	6.0	3.8	Thu 11 May 1995 14:56
mercury	DECmips	4.5	3.8	4.9	6.6	Fri 12 May 1995 17:19

Test Run #	1	2	3	4
Range	[3.0-7.2]	[3.1-7.5]	[4.9-7.3]	[2.4-6.6]
Median	5.0	3.8	5.8	2.6
Average	4.9	4.3	6.3	3.2

Biographical Note

The author was born 15 December 1969 in Chicago, Illinois. In 1993 he graduated Magna Cum Laude with a Bachelor of Science degree in Computer Science from North Carolina State University in Raleigh, North Carolina.

Following graduation, he began working towards a Master of Science degree in Computer Science and Engineering at the Oregon Graduate Institute of Science and Technology in Portland, Oregon. He has recently accepted a Member of Scientific Staff position with the Advanced Information Technology department of the MITRE Corporation in McLean, VA.