NewsComm: A Hand-Held Device for Interactive Access to Structured Audio

Deb Kumar Roy

B.A.Sc. Computer Engineering, University of Waterloo, 1992

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, in partial fulfillment of the requirements for the degree of

Master of Science

at the

Massachusetts Institute of Technology

June 1995

©Massachusetts Institute of Technology 1995. All Rights Reserved.

Signature of the Author:

Program in Media Arts and Sciences
May 12, 1995

Certified by:

Christopher M. Schmandt
Principal Research Scientist
MIT Media Laboratory

Accepted by:

Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

NewsComm: A Hand-Held Device for Interactive Access to Structured Audio

Deb Kumar Roy

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, on May 12, 1995 in partial fulfillment of the requirements for the degree of

Master of Science

at the

Massachusetts Institute of Technology

Abstract

The NewsComm system delivers personally selected audio information to mobile users through a hand-held audio playback device. The system provides a one-to-one connection from individual users to information providers so that users can access information on demand with breadth and depth unattainable through traditional media. Filtering mechanisms help the user efficiently find information of interest.

The hand-held device receives audio through intermittent high band-width wired connections to a central audio server. The server collects and stores audio from various sources including radio broadcasts, books on tape, and internet multicasts. Each recording in the server is preprocessed by a set of audio processing algorithms which automatically extracts a structural description of the audio. The hand-held interface uses these structural descriptions to enable efficient skimming and searching. The device is ideal for use while commuting or exercising.

This thesis describes the design, usability testing, and implementation of the hand-held device, the audio server, and the speech processing algorithms.

Thesis Supervisor: Christopher M. Schmandt
Title: Principal Research Scientist

Thesis Readers

Reader _____

Walter Bender
Associate Director for Information Technology
Director, News In the Future
MIT Media Laboratory

Reader _____

Robert Donnelly
Vice President, Engineering
ABC Radio Networks

Reader _____

John Makhoul
Chief Scientist,
Bolt Beranek and Newman Inc.

Contents

Chapter 1: Introduction 7

1.1 Overview……..... 8
1.2 Motivations…… 9
1.2.1 Access to More Information than Broadcast Media Can Afford…… 9
1.2.2 Filtering Out Unwanted Information……..10
1.2.3 Why Audio?……...11
1.3 Structured Audio…… ...11
1.4 Interactive Access……..12
1.5 Overview of the Document…… ...12

Chapter 2: Related Work 14

2.1 SpeechSkimmer…… 14
2.2 A System for Capturing, Structuring and Accessing Telephone Conversations 15
2.3 NewsTime.....16
2.4 VoiceNotes.....17
2.5 AudioStreamer 17
2.6 Structure Out of Sound.....18
2.7 Filochat.....18
2.8 Speaker Segregation and Indexing.....18
2.9 Automatic Summary of Spoken Discourse.....19

Chapter 3: Structured Audio 20

3.1 Pause Detection 20
3.2 Locating Speaker Changes....21
Assumptions 22
3.3 The Speaker Indexing Algorithm 23
3.3.1 Signal Processing 23
3.3.2 Training the Neural Networks 24
3.3.3 Experimental Results....26
3.3.4 Discussion....27

Chapter 4: A Framework for Combining Multiple Annotations 29

4.1 Two Problems with SpeechSkimmer....29
4.2 The NewsComm Approach 30

Chapter 5: Design of the Hand-held Interface 36

5.1 Design Objectives……..36
5.2 Usability Testing……....37
5.3 Summary of the Design Results....37
5.4 Detailed Description of Each Design Iteration....38
5.4.1 Version 1 38
5.4.2 Version 2 39
5.4.3 Version 3 43
5.4.4 Version 4 45
5.4.5 Version 5 47

Chapter 6: Implementation of the Hand-Held 51

6.1 Software Implementations....51
6.2 Hardware Implementations 52
6.2.1 Version 1: Proof of Concept....52
6.2.2 Version 2: Porting the Software Interface to Hardware....57
6.2.3 Version 3: The Final Hand-Held Device 59

Chapter 7: The Audio Server 62

Chapter 8: Conclusions and Future Work 69

8.1 Conclusions 69
8.2 Future Work 70
8.2.1 Improve the Speaking Indexing Algorithm 70
8.2.2 Port the Hand-held Interface to a Standard Hardware Platform 71
8.2.3 Implement Time-Scale Modification 71
8.2.4 Add Audio Compression 71
8.2.5 Add More Annotations 71
8.2.6 Add Wireless Reception for Real-Time Data 72
8.2.7 Supporting Richer Listening History and Preferences 72
8.2.8 Add Recording Facilities to the Hand-Held to Enable Community Reporting 73

References **74**

Acknowledgments **77**

Chapter 1
Introduction

The world is becoming digital; television, radio, newspapers, music, and most other forms of information are making the transition from atoms and analog signals to digital bits. The advantages of going digital are tremendous. Bits can be shipped at the speed of light, stored in incredibly high densities, duplicated without degradation, and accessed in random order.

But the explosion of digital information has also created new problems. How do we find our way through such a complicated and dynamic landscape? NewsComm provides one solution to this problem by delivering personally selected audio to mobile users.

The NewsComm system includes a hand-held device which provides interactive access to structured audio recordings. The device, shown in Figure 1, has been designed for mobile use; it can be held and operated with one hand and does not require visual attention for most operations. Users can intermittently connect the hand-held to an audio server and receive personally selected audio recordings which are downloaded into the local random access memory of the hand-held. The user can then disconnect from the server and interactively access the recordings off-line. The system is ideal for delivering personalized information to users when their eyes are busy, such as when they commute or exercise.

Figure 1: The NewsComm hand-held audio playback device with headphones. The top face houses a display and controls for selecting and managing recordings which have been downloaded into the hand-held's memory. The right side houses the navigation interface which can be controlled with the thumb while holding the device. The device can be connected to a central server to download personally selected digital audio recordings.

1.1 Overview

Figure 2 gives an architectural overview of the NewsComm system. The audio server (top part of Figure 2) collects and processes audio from various sources including radio broadcasts, books and journals on tape, and internet audio multicasts[1]. Typical types of content might include newscasts, talk shows, books on tape, and lectures. The hand-held downloads recordings from the audio server through intermittent high-bandwidth connections[2].

---

[1]Multicasting provides one-to-many and many-to-many network delivery services for applications such as audio and video conferencing where several hosts need to communicate simultaneously. The NewsComm server receives audio from the Internet Multicasting Service, a group which regularly multicasts audio onto the internet.

[2]In the prototype system the connection is made by removing a memory card from the hand-held and inserting it into the server. In the future this connection might be made over television cable or ethernet.

The audio processor module in the server automatically finds two types of features in each audio recording stored in the server: pauses in speech, and speaker changes (points in the recording where the person talking switches). The locations of these features constitute a structural description of the recording. The audio and associated structural description are collectively referred to as structured audio. All audio in the server is structured by the audio processor and then stored in the audio library, a large network-mounted hard disk drive.

Users can download structured audio from the server by connecting their hand-held to the audio manager. The audio manager decides which recordings to download based on a preference file which the user has previously specified, and also based on the recent usage history uploaded from the hand-held.

Once the download is complete, the user can disconnect the hand-held from the server and interactively access the recordings using the navigation interface of the hand-held. The playback manager in the hand-held uses the structural description of the audio to enable efficient navigation of the recordings. It does this by ensuring that when the user wishes to jump forward or backward in a recording, the jump "lands" in a meaningful place rather than a random one. The structural description of each recording contains the location of all suitable jump destinations within the recording. The interface enables the user to efficiently skim and search audio, and to listen selectively to portions of interest.

1.2 Motivations

1.2.1 Access to More Information than Broadcast Media Can Afford

Broadcast media including radio and television typically have shallow repetitive coverage of only mainstream news to maximize the number of listeners. NewsComm creates a mechanism for one-to-one connections between individuals and information providers. The result is an increase in both the depth and breadth of information which the listener can access.

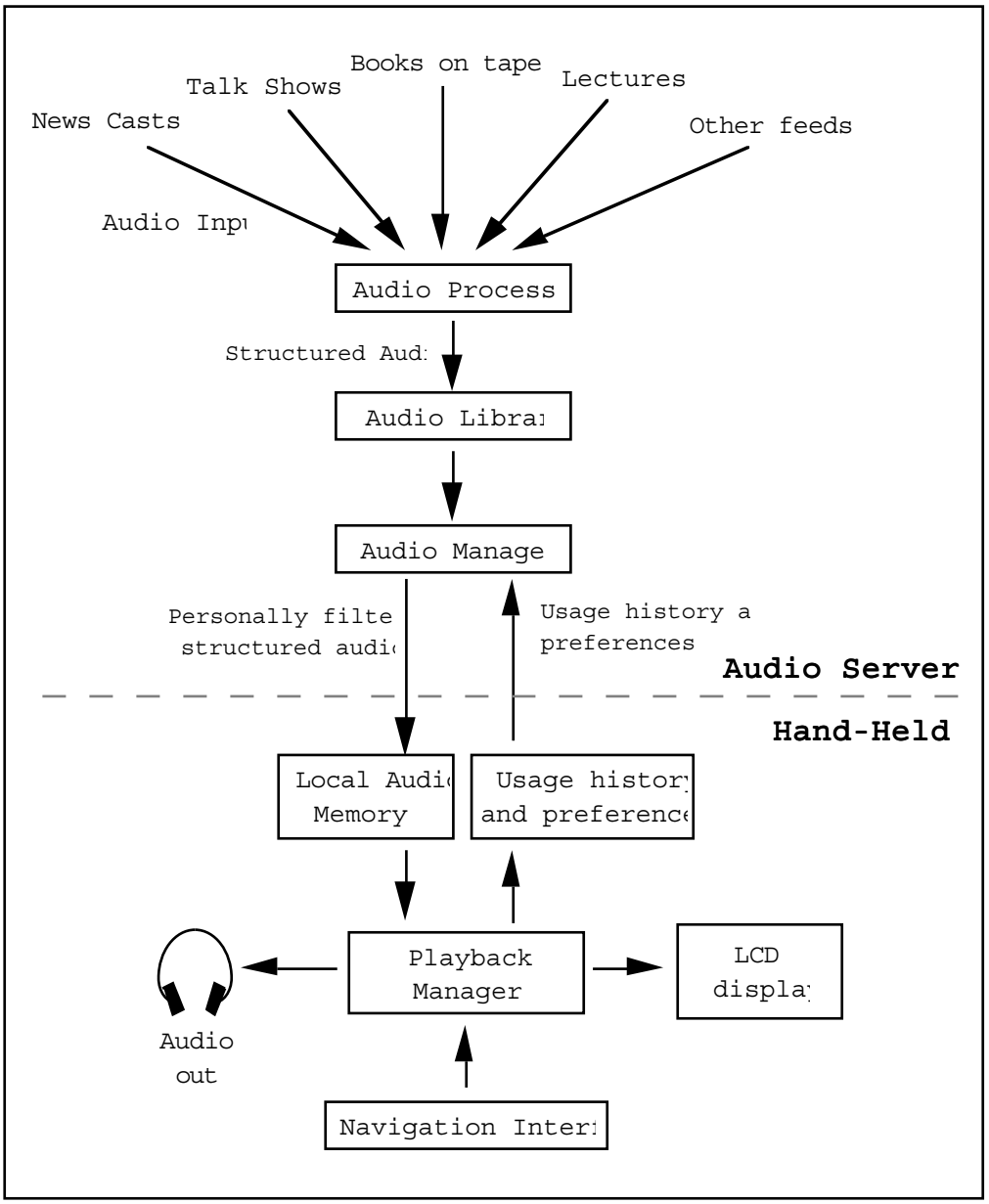

Figure 2: An overview of the audio server and hand-held playback device. The dotted line separates the components of the audio server and the hand-held. When the hand-held connects to the server, the usage history and preferences are uploaded to the audio manager, and based on this information a set of filtered structured audio recordings are downloaded into the hand-held's local audio memory.

1.2.2 Filtering Out Unwanted Information

An underlying assumption in the NewsComm system is that there is more audio available than the listener wants to hear. Imagine if every lecture hall, court, classroom, and other source of interesting audio is wired with microphones and connected to a large audio server. Enormous amounts of audio could be collected,

but some sorts of filtering mechanisms are needed to help find information of interest.

NewsComm filters information at two levels. At the first level, the audio manager uses usage history and user preferences to select which recordings are downloaded. For example, the user may prefer a specific source for morning news, and have an interest in journal articles regarding a specific topic. Only these filtered recordings would automatically be downloaded when the user connects to the server.

At the second level of filtering, the interactive control of the hand-held's playback manager enables the user to listen selectively to only portions of interest from the downloaded recordings. The assumption is that the audio server will download more information than the user wants to hear since there are no highly reliable ways to automatically determine what a user will and will not be interested in (so it is best to err on the side of too much information). The interactive interface lets the user make the final decisions on what he[3] will listen to.

1.2.3 Why Audio?

Audio is attractive for accessing information since it can be used when the user's eyes and hands are busy. NewsComm is aimed at mobile users who are performing some other simultaneous task such as walking, exercising, or driving. Visually impaired individuals are also potential NewsComm users.

1.3 Structured Audio

Audio is structured in NewsComm by automatically annotating pauses and speaker changes in each recording. Pauses are found by analyzing the long term distribution of energy in the audio signal. Speaker changes are found using a new algorithm called speaker indexing which uses dynamically generated back-propagation neural networks to cluster the mel-scaled spectra of vowels. The collection of one of more annotations of a recording are collectively referred to as the structural description of the recording. This thesis also introduces a framework

[3]The user is referred to as "he" for brevity and should be replaced with "he or she" in meaning throughout this document.

for combining multiple annotations of a media stream for the purpose of navigation.

1.4 Interactive Access

The hand-held interface exploits the structural information from the annotations to enable simple yet powerful navigation. For example, during a connection to the audio server, the hand-held might receive several recordings including a newscast. The listener can use the interface to select the newscast and start playing it. The newscast can be skimmed by listening to short portions and then pressing a "jump-forward" button to jump the play position to the next interesting event in the recording such as a long pause (which often precedes a change in story or topic) or a speaker change (which might be the start of an interview or field report). The user controls the depth of coverage of each story by deciding when and how often to jump. The structural description of the audio provides meaningful jump locations. Skimming the contents of a recording by listening to segments of audio following these locations (rather than random locations) has been shown to be a more effective way to get the gist of the contents of a recording [Arons].

The hand-held interface is the result of an iterative process of design and usability testing.

1.5 Overview of the Document

Chapter 2 reviews related work in both structured audio and audio interface design.

Chapter 3 describes the speech processing algorithms which are used in NewsComm to automatically extract structure from speech recordings. Experimental results of accuracy tests on the algorithm are also presented.

Chapter 4 describes a method for combining annotations from any number of different sources in a unified framework.

Chapters 5 and 6 presents the iterative development of the hand-held device. Chapter 5 describes the design and usability testing of five versions of the hand-held interface. Chapter 6 describes hardware and software implementation details of the device.

Chapter 7 describes the design and implementation of the audio server. The interaction between the hand-held and server are explained here.

Chapter 8 makes some concluding remarks about the thesis, and outlines future directions for the NewsComm system.

Chapter 2
Related Work

This chapter reviews several research systems which have addressed issues related to this thesis.

2.1 SpeechSkimmer

SpeechSkimmer is a hand-held interface for interactively skimming recorded speech [Arons]. The interface enables the user to listen to a speech recording at four levels of skimming. At the lowest level the entire recording is heard. At the second level pauses are shortened. At the third level, only short segments (called highlights in this thesis) of the recording following long pauses are played; the portions of the recording between these highlights are skipped. Level four is similar to level three; only highlights of the recording are played. In contrast to level three, the highlights are selected based on pitch analysis rather than pause locations.

The SpeechSkimmer interface also employs the synchronized overlap add method of time-scale modification to speed up and slow down speech without altering the pitch [Roucos].

NewsComm is similar to SpeechSkimmer since it also provides a navigation interface for efficiently skimming and searching audio recordings.

There are however several differences between SpeechSkimmer and the NewsComm hand-held interface:

- SpeechSkimmer uses pause detection and pitch analysis to annotate recordings; NewsComm uses pause detection and speaker separation

- NewsComm introduces a new framework for combining multiple annotations of a recording. This framework separates the design of the navigation interface from the underlying structural representation of the audio (See Chapter 4). In contrast, the SpeechSkimmer interface is directly tied to the underlying representation of the audio structure.

- SpeechSkimmer is an interface for skimming a single recording. The NewsComm interface is designed for handling multiple recordings. It includes controls for choosing which of multiple recordings to listen to, and also includes controls for managing server related operations.

- SpeechSkimmer is a tethered device (it is connected to a Macintosh computer system). The NewsComm hand-held is self-contained and requires no wired connection to external devices or power sources

2.2 A System for Capturing, Structuring and Accessing Telephone Conversations

Hindus et. al. designed a set of applications for capturing and structuring audio from office discussions and telephone calls, and mechanisms for later retrieval of the stored interactions [Hindus]. Audio recordings are annotated using a combination of manual and automatic methods. For manual annotation, the system provides a visual display of speech recordings; users can click on segments of pause separated speech and annotate them. The recordings are also annotated automatically using pause detection and speaker separation. Speaker separation is accomplished by comparing the acoustic signal at either end of a telephone conversation. The system supports an X-windows based graphical interface for accessing audio once it has been annotated. The interface displays segments of speech as rectangles. The length of the rectangles are proportional to the duration of the speech segment. The rectangles are laid out from left to right to represent the time sequence in which they occurred. The vertical axis of the display is used to indicate the speaker of each segment. For example a two person conversation consists of a series of rectangles with alternating vertical positions. The audio corresponding to each rectangle can be heard by clicking on the display.

NewsComm is similar to this system since it also uses pause detection and speaker separation to annotate recordings. However the NewsComm method for locating speaker changes does not rely on telephone acoustics (it uses speech processing methods to analyze voice characteristics of the speakers), so it can be used on a much broader range of audio recordings. During development of the NewsComm hand-held interface several graphical interfaces were implemented which provide on screen access to audio, similar to [Hindus]. However the main goal of this thesis work was the design of a hand-held device which would provide mobile access to audio without requiring much visual attention.

2.3 NewsTime

Horner designed a graphical interface for accessing structured audio called NewsTime [Horner]. NewsTime is similar to Hindus's system in that it also provides a screen-based visual interface to structured audio. NewsTime structures the audio track of television newscasts by locating pauses, and by extracting information from the closed caption data stream which accompanies many television broadcasts[4]. The closed caption data is used to locate story and speaker changes (which are explicitly specified in the data), and is also used to tag topics by detecting predefined keywords in the closed caption text. The graphical interface shows the closed caption text in one window, and a visual representation of the audio recording in another. Speaker changes, story changes, and topic locations are indicated with icons; clicking on any icon jumps the play position to the corresponding location in the recording. The interface provides a visualization of a complete recording and enables quick navigation to any portion of interest.

NewsTime relies heavily on closed caption information which restricts its use to only audio which have accompanying time-synchronized text transcripts. Although this allows much deeper understanding of the recording, it reduces the types of audio which can be accessed. NewsComm currently uses relatively shallow types of annotations, but is designed to naturally incorporate other annotations including those based on text transcripts.

---

[4]Closed captions provide time-synchronized text captions of television programs for the hard of hearing.

2.4 VoiceNotes

VoiceNotes is an interface for a hand-held audio note taking device which can be carried around and used to capture and organize short spoken notes [Stifelman]. The implementation consists of a modified microcassette recorder which is connected to a speech recognition system and a Macintosh computer. The device combines buttons and speech recognition input for navigation control and can be operated without visual attention. Short speech segments (notes) can be recorded and organized into lists. The interface enables the user to dynamically create and name lists, and use a combination of button presses and isolated word voice commands to navigate between lists and notes.

Many of the interface design issues addressed in Stifelman's thesis also apply to the NewsComm design problem. Speech recognition was not used in NewsComm primarily since the goal was to implement an untethered device, which makes it difficult to support speech recognition.

2.5 AudioStreamer

Mullins designed an audio browsing system which relies on the cocktail party effect, the ability humans have to separate and understand one stream of audio which is mixed with several other acoustic streams [Mullins]. The effect is named after the ability people have at a cocktail party to follow one conversation and tune out surrounding conversations. AudioStreamer uses three dimensional audio production to place three audio sources at distinct points in three dimensional space surrounding the head of the listener. All three streams are played simultaneously. The user can browse the streams by changing his focus from one source to the next (relying on the cocktail party effect to filter out the remaining two streams).

Salient events (pauses and speaker changes) are automatically tagged in each audio stream. AudioStreamer uses the speaker indexing algorithm described in this thesis to annotate recordings. When a salient event is reached in one of the streams, a chime is played to capture the listener's attention, and the relative volume of that stream is momentarily increased to ease the change of focus. The volume of the stream quickly decays unless the user is interested in listening to that stream. A computer keypad interface allows the user to adjust the relative volume of any stream manually.

In contrast to SpeechSkimmer and NewsComm which treat browsing as a linear problem (i.e. jumping back and forth in a single audio stream), AudioStreamer relies on parallelism.

2.6 Structure Out of Sound

Hawley designed a set of audio processing tools called sound sensors which extract structural information from audio recordings [Hawley]. Hawley implemented three sensors: a polyphonic pitch extractor, a music detector, and a pitch based speaker recognizer. The output of these sensors are combined and encoded in an ASCII text file which can be used by an application to access the contents of the recording. Hawley describes one such application called MediaWhacker, a graphical interface for navigating an annotated video stream. NewsComm also introduces a method for combing multiple annotation sources (see Chapter 4) and addresses interface issues of navigating structured media streams.

2.7 Filochat

Whittaker et. el. have developed a system called Filochat which integrates handwriting and recorded audio in a portable system [Whittaker]. It consists of a laptop computer attached to a sound card with microphone and speaker, and a write-on LCD tablet. The system can record audio and time-synchronized written notes. Audio can later be accessed by gesturing to a written note which automatically starts playing the audio which was recorded at the time the note was written. Usability tests have shown that the system is preferred to written notes alone, and in field test users perceived benefits of higher quality meeting minutes. Some of the ideas developed in Filochat might be applied to a future version of NewsComm if NewsComm's interface is extended to support user annotations of audio.

2.8 Speaker Segregation and Indexing

The NewsComm system uses a novel algorithm to separate and index speakers in a speech recording. This section describes two earlier efforts to solve similar problems.

Gish et. al. have developed a method for segregating speakers engaged in dialog [Gish]. The method assumes no prior knowledge of the speakers. A distance measure based on likelihood ratios is developed which is used to measure the distance between two segments of speech. Agglomerative clustering based on this distance measure is used to cluster a long recording by speaker. The method has been successfully applied to an air traffic control environment where the task is to separate the controller's speech from all pilots. Since the controller speaks more often than any of the pilots, the largest cluster is labeled as the controller.

Wilcox et. al. also uses a likelihood ratio based agglomerative clustering algorithm to index speakers [Wilcox]. Additionally, they use a hidden markov model to model speaker transition probabilities.

In contrast to both of these approaches, NewsComm uses back-propagation neural networks to cluster speech segments (rather than a likelihood ratio based distance measure).

2.9 Automatic Summary of Spoken Discourse

Chen and Withgott describe a method for summarizing speech recordings by locating and extracting emphasized portions of the recording [Chen]. Hidden markov models (HMMs) are used to model emphasis regions. The energy, delta energy, pitch and delta pitch parameters are extracted from the speech and used as parametric input to the HMM. Training data was collected by manually annotating the emphasized portions of several speech recordings.

NewsComm uses a similar method to automatically summarize a recording. Speaker change and pause locations are combined to locate points of interest in a recording. Short segments of speech following these locations are extracted and concatenated to form a summary of the recording.

Chapter 3
Structured Audio

This chapter describes the algorithms used in the audio processor to automatically extract the locations of pauses and speaker changes from a speech recording. Chapter 4 describes how these annotations are combined and used in NewsComm.

3.1 Pause Detection

The speech recording is segmented into speech and silence by analyzing the distribution of energy across the entire recording. The energy of overlapping frames of the input signal are computed using Equation 1:

$$E[n] = \sqrt{\sum_{i=(n-1)K/2}^{(n+1)K/2} \left(x[i]\right)^2} \qquad \text{(EQ 1)}$$

where *E[n]* is the energy of the n^{th} frame, *x[i]* is the value of the input signal at sample *i*, and *K* is the number of samples in a frame. In the NewsComm system, *K = 512*, and the sampling rate is 8 kHz, thus the frames are 64ms long and overlap adjacent frames by 32ms.

The histogram of frame energies is computed across the entire audio recording. Once the energy distribution has been computed, the 20% cutoff is found, and all samples of the recording which lie in the bottom 20% of the distribution are labeled as silence; the remaining 80% of samples are tagged as speech. The 20% cutoff was chosen based on the analysis of a set of BBC radio newscasts. The value (20%) assumes a 5:1 ratio of speech to silence in the audio recording which has

been found to an acceptable approximation for other professionally recorded speech (including books on tape and newscasts from other sources) through empirical observations made during the development of the algorithm.

Once the 20% threshold has been applied, single-frame segmentation errors are corrected: any single-frame segments (i.e. a single frame tagged as speech surrounded by silence segments or vice versa) are removed.

By recomputing the distribution for each recording to find the 20% threshold (rather than using a fixed energy threshold), the algorithm can deal with varying levels of background noise, assuming the noise level is approximately constant within the recording.

3.2 Locating Speaker Changes

An algorithm called speaker indexing (SI) has been developed which separates speakers within a recording and assigns labels, or indices, to each unique speaker. The current NewsComm system only uses locations of speaker changes and ignores speaker identity although identity information may be used in the future. The SI algorithm is described in this section.

To understand the goal of SI consider the following example: a recording which contains the voices of four people is shown in Figure 3. The top strip represents the sequence of speakers in the recording (time flows from left to right). In this example, Speaker A begins talking, followed by Speaker B, then Speaker C, then back to Speaker A and so on. Changes in speakers are indicated by vertical bars. Given the audio recording as input, the ideal output of the SI system is shown in the lower strip. Each speaker change boundary is located, and indices are assigned to each segment which are consistent with the original identities of the speakers. Since the SI system has no prior models of the speakers, it does not identify the speakers, but rather separates them from each other within the recording.

An important distinction between the SI problem and conventional speaker identification is that there is no assumed prior knowledge about the speakers in the input recording. In speaker identification, a set of models of all possible speakers is created using training samples of each speaker. Identification of an unknown sample is performed by comparing the speech sample to each speaker model and

finding the closest match. For the class of applications we are interested, we could not assume the a priori availability of training data for speakers. Thus conventional speaker identification techniques cannot directly be applied.

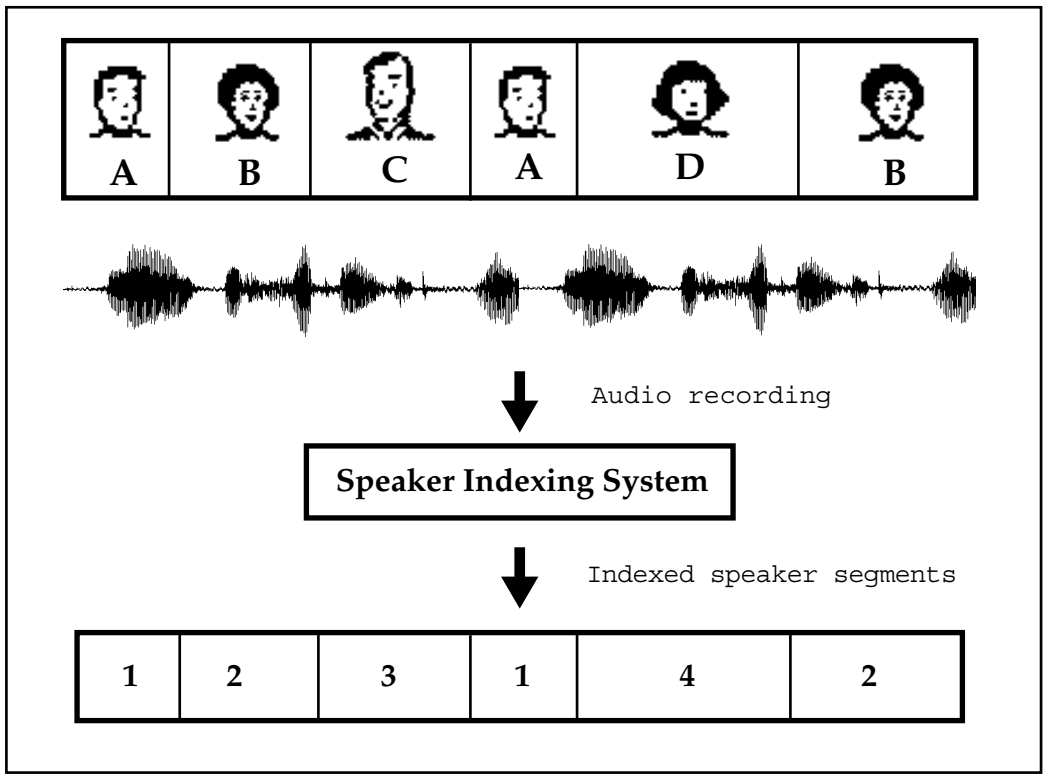

Figure 3: Ideal Output of the SI System: The top strip represents the sequence of four speakers in a recording (time flows from left to right). The audio recording (shown as a speech wave) is processed by the SI system which outputs a sequence of indexed segments. Ideally each segment output from the SI system corresponds to a speaker turn in the input recording, and the indices assigned to each segment correspond to a actual speaker identity (in this example Index 1 corresponds to Speaker A, Index 2 to Speaker B, 3 to C, and 4 to D). A simple application would be to play short audio segments directly following each speaker turn to get the gist of the recording, without having to listen to it all.

Assumptions

The SI system was initially developed to separate and index speakers in BBC radio newscasts. The BBC news broadcasts are 20 minutes long and contain between 12 and 20 unique speakers each. Each broadcast is hosted by two speakers, and the remaining speakers are foreign correspondents, special reports, and interviews. The background noise level varies widely, from very clean signals for the studio recordings of the hosts, to highly degraded signals of some field reports.

The assumptions afforded by the BBC indexing task are:

(1) The minimum speaker turn is 5 seconds

(2) The minimum pause between speaker turns is 0.2 seconds

(3) The entire audio recording is available before processing begins

Assumption (1) was found to be true through empirical analysis of several BBC news broadcasts; no speaker talks for less than 5 seconds except when an interview is conducted within the news program (in which cases the system is expected to miss segments). Also through empirical measurements, Assumption (2) was found to be valid for BBC news except during interviews; there is generally a clean break between speakers. Assumption (3) can be made in our situation since the audio is recorded and processed off-line.

3.3 The Speaker Indexing Algorithm

The speaker indexing algorithm dynamically generates and trains a neural net to model each postulated speaker found in the recording. Each trained neural net takes a single vowel spectrum as input, and outputs a binary decision indicating whether the vowel belongs to the speaker or not.

3.3.1 Signal Processing

Figure 4 shows the signal processing front end which extracts mel-scaled vowel spectra, and locates pauses in the speech recording. The speech input is sampled at 8000 samples per second using a 8-bit ulaw encoded analog-to-digital converter. On the far left, the adaptive speech and silence detector computes the speech/silence energy threshold of the recording by generating a histogram of the energy distribution over the entire recording, and tagging the low 20% of the distribution as silence as described in Section 3.1. The energy of the input signal is computed over a 64ms frame, overlapped 32ms. A pause detector locates contiguous frames of silence which last longer than 0.2 seconds (this is used to train the neural nets, as explained below). Each set of vowel spectra delimited by such pauses will be referred to as a “sentence” in the remainder of this section. Note that based on assumption 2 from the previous section, we can infer that each sentence must be spoken by only one speaker.

On the right hand side of Figure 4, a fast Fourier transform (FFT) of the input signal is computed using a 64ms hamming window with 32ms overlap. The resultant spectrum is passed through a mel-scaled filter bank which produces a 19 coefficient spectral vector. In the time domain, a peak picker estimates the location of vowels by picking peaks of the energy of the speech signal (vowels have relatively high airflow and thus a corresponding peak in the energy contour). The "logical and" of the outputs of the peak picker and the speech/silence detector is computed in order to eliminate false vowel detection by the peak picker during background noise.

Only the mel-scaled spectra corresponding to each vowel is output to the neural network portion of the system. This is depicted by the sample mel-scaled spectrogram in the figure which represents several seconds of speech. Four frames have been identified by the peak picker as vowels and are output to the neural network portion of the system. Non-vowel information is discarded in order to reduce the size of the neural networks.

Although most vowels in the recording will occupy more than a single 64ms frame, the current implementation only selects the single frame corresponding to the center of the energy peak.

3.3.2 Training the Neural Networks

The SI system employs back propagation neural networks to model each postulated speaker in the input recording. Back propagation neural networks are trained through a supervised process [Rumelhart]. For a network with binary output, a set of positive and negative training examples are required. The examples are presented in sequence to the network. The weights of the network are adjusted by back-propagating the difference between the network's output and the expected output for each training example in order to minimize the error over the entire training set.

If the positive training examples are a subset of the vowels spoken by some Speaker X, and the negative examples are a subset of the vowels spoken by all the other speakers, we can expect the trained network to differentiate vowels generated by Speaker X from all other speakers (including vowels that were not in the training set).

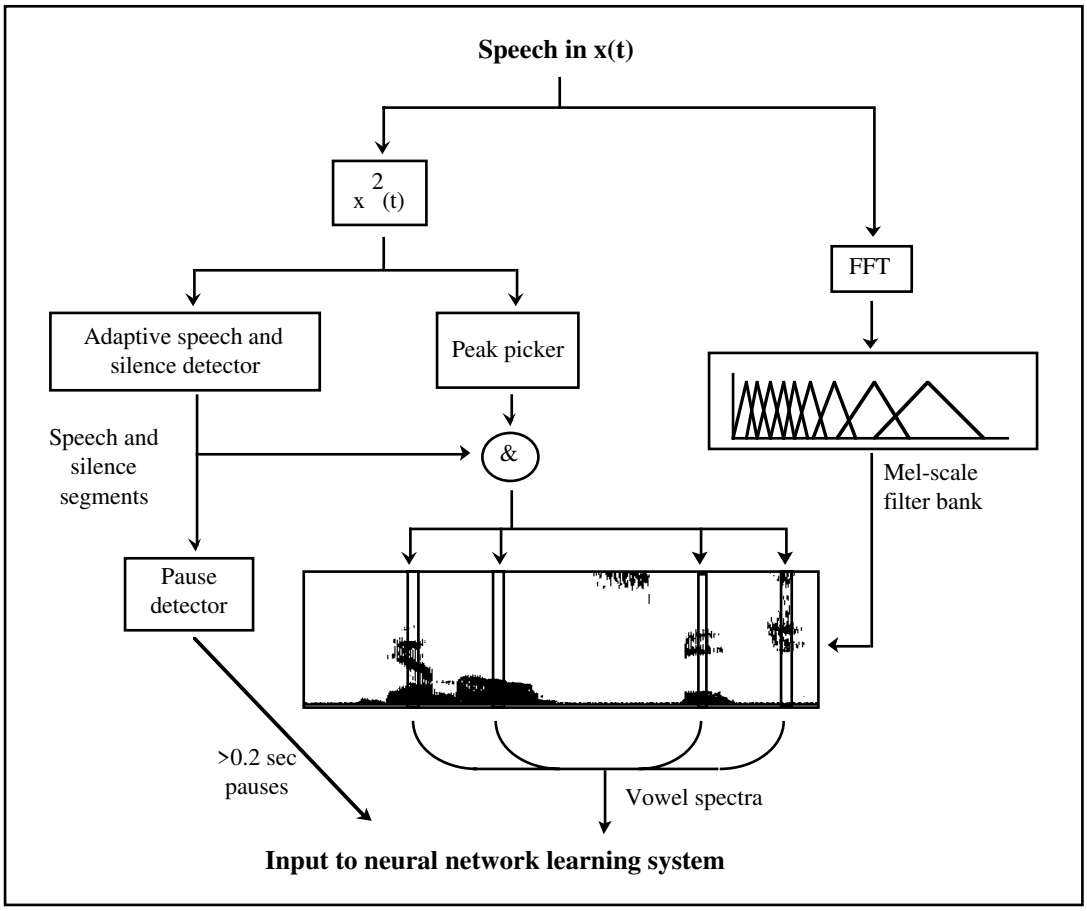

Figure 4: Signal Processor extracts mel-scaled spectra of vowels, and locates pauses longer than 0.2 seconds.

However, since there is no a priori knowledge of the speakers, training data must be selected automatically. This selection process begins by assuming that the first 5 seconds of the recording was spoken by a single speaker, Speaker 1. The spectra of the vowels from this 5 second segment comprise the positive training data for the first neural net. A random sampling of 25% of the remainder of the recording is used as negative training data. Note that the negative training set selected in this manner will probably contain some vowels which belong to Speaker 1, leading to a sub-optimal speaker model.

Once the neural network has been trained using this training set, the network is used to classify every vowel in the recording as either belonging to Speaker 1 or not (true or false). The resultant sequence of classification tags is then filtered to eliminate tags which do not conform to Assumption 2. This is accomplished by applying a "majority rules" heuristic; for each sentence in the recording, if the majority of tags belong to Speaker 1, then all of the vowels in that sentence are

tagged as true. On the other hand, if the majority are classified as false, then all tags for that sentence are set to false. This filtering process has two effects: (1) possible false-positive tags generated by the neural network are removed, and (2) vowels which were not recognized as Speaker 1 are "picked up" in cases where the majority (but not all) of the vowels in a sentence were positively tagged. This filtering process partially compensates for errors in the training set. A second filter is then applied which enforces Assumption 1: any sequence of tags which is shorter than the minimum speaker turn is inverted.

Once the two levels of filters have been applied, the neural network is re-trained. All of the vowels which have been classified as Speaker 1 (after filtering) are collected and comprise the new positive training set, and again 25% of the remaining vowels (randomly selected) comprise the negative training set. This entire training, tagging, and filtering cycle is repeated until no further positive training vowels are found.

Once the first speaker has been located, the audio corresponding to that speaker is removed from the input recording, and a new neural network (for Speaker 2) is created and trained on the remaining audio using the same procedure. This cycle is repeated until all audio in the input recording has been indexed.

3.3.3 Experimental Results

The accuracy of the speaker indexing algorithm has been tested on two sets of data. The first is a set of ten 20-minute BBC newscasts recorded over a two week period. Each recording contains about 15 unique speakers. The second test set contains six 15-minute clips of TechNation interviews [TechNation]. Five of the TechNation clips contain two unique speakers, and the remaining clip contains three.

Speaker transitions and indices for all 16 recordings were hand annotated. A set of test software has been written which runs the speaker indexing software in batch mode on all recordings in a test set and computes average accuracy scores across the entire set by comparing the output of the indexing program to the manual annotations.

Accuracy has been measured in three ways for each test set:

- Speaker indexing: the number of frames of the recording that were indexed correctly as a percentage of the total number of frames
- Speaker change hits: the percentage of speaker changes which were detected by the algorithm with an error of less than 1.0 seconds
- False alarm percentage: the percentage of speaker changes detected by the algorithm which were not classified as hits

The results are shown in Table 1:

Test set	**Indexing Accuracy**	**Speaker change hits**	**False alarms**
BBC newscasts	64	50	55
TechNation	89	57	57

Table 1: Experimental results of the indexing algorithm (all values are percentages)

3.3.4 Discussion

The indexing algorithm has a relatively high error rate for all three measures. We believe that the main reason is the training initialization process which uses random selection of negative data for training the neural nets. Analysis of the algorithm shows that in many cases a poor choice of initial training vectors causes segments of a recording which belong to a single speaker to be fragmented and assigned multiple indices. This leads to a drop in indexing accuracy, and a rise in the false alarm rate. Similarly, poor training data can also cause different speakers to be collapsed into one neural net model. This leads to a drop in speaker change hits and indexing accuracy.

As a next step we plan to introduce a clustering step into the process which does an initial coarse level separation of speakers similar to [Gish] and [Wilcox]. This stage will be used to select initial training data for the neural networks.

Another source of error may be the use of mel-scaled spectral coefficients rather than a smoothed representation of the spectrum such as linear predictive coding or cepstral coding since most speakers have overlapping fundamental frequency characteristics [Makhoul]. We plan to switch to a cepstral representation to test this hypothesis.

It is important to note that although the error rates are high, the system does locate half or more of the speaker changes in recordings. The NewsComm interface has been designed with the assumption that the structural description of the audio have errors. Even with the given error rates, in practice the NewsComm hand-held has been proven to be an effective navigation device when speaker indexing output is combined with pause locations.

Chapter 4
A Framework for Combining Multiple Annotations

The goal of a structured representation is to have "handles" into a large media stream. If placed in meaningful or salient locations, these handles can be used to increase the efficiency of browsing and searching the stream. The NewsComm system chooses the location of these handles by combining information about pause and speaker change locations. Long pauses usually predict the start of a new sentence, a change of topic, a dramatic pause of emphasis, or a change in speaker [O'Shaughnessy]. Speaker changes can be useful when listening to an interview, conversation, debate, or any other recording containing multiple speakers.

This chapter first reviews the representation scheme used in an earlier skimming system, SpeechSkimmer [Arons], and then describes the new NewsComm framework for combining multiple annotations.

4.1 Two Problems with SpeechSkimmer

SpeechSkimmer is an interface for interactively skimming recorded speech (the system is reviewed in Chapter 2) [Arons]. In this section we consider two problems with the SpeechSkimmer interface which are addressed in the NewsComm system.

SpeechSkimmer associates levels of skimming with specific structural properties of the recording. For example, at the third level of skimming segments of the recording following long pauses are played. At the fourth level, segments determined by variations in pitch (indicating emphasized speech) are played. The system can be expanded by adding more types of annotations and assigning a new level of skimming to each type.

Two problems with this model have been identified. First, the number of levels of skimming (and thus the number of controls in the interface) is directly tied to the number of types of annotations available. The interface must be expanded as new annotation types are added. This becomes impractical for more than three or four types of annotations since the interface becomes too complex. Second, there is no guarantee of even temporal coverage of the recording; if the distribution of a specific annotation type is temporally uneven, then the resulting skim of the recording will also be uneven. For example, if a recording has many long pauses in the first half and only shorter pauses in the second half, at level three skimming SpeechSkimmer will not play many segments from the second half of the recording.

4.2 The NewsComm Approach

In NewsComm a framework has been developed which combines any number of annotations and provides a uniform representation of the structure to the navigation interface. In addition to audio, the framework can be used with other types of media streams including text and video. Unlike SpeechSkimmer, this framework separates the design of the interface from the structural description of the audio.

To describe the NewsComm framework we first define some terms:

play position: a pointer into the audio recording which indicates the most recently played sample

jump: to move the play position from its current location to a non-adjacent position

jump range: the maximum distance a jump can move the play position from its current position

jump granularity: a quantitative measure of the spacing between jumps in a recording. The closer together the jumps, the finer the granularity; the more spread out the jumps, the coarser the granularity.

All iterations of the NewsComm interface design use the fundamental notion of *jumping* to facilitate navigation functions including skimming and searching. The purpose of the framework described in this chapter is to locate jump locations within an audio recording at any level of granularity. The jump locations can be used by applications to enable efficient access to the contents of the recording. Recordings can be skimmed by playing short segments following each jump. Recordings can be summarized by extracting and concatenating speech segments following each jump location.

Note that the interface does not need to know how the jump locations were chosen, thus the design of the interface is isolated from the underlying annotations.

We now define the salience of the i^{th} frame of the recording, S:

$$S[i] = \sum_{j=1}^{n} w_j A_j[i] \qquad \text{(EQ 2)}$$

where there are n types of annotations, w_j is the weight of the j^{th} annotation type, and $A_j[i]$ is the value of the j^{th} annotation for frame i of the recording. Frames are 64ms long and spaced 32ms apart as defined in Section 3.1. This is a general equation which can be used to compute salience based on any number of annotations.

In the present system there are two types of annotations: pauses, and speaker changes, and the weights for both annotations are set to 1.0. Equation 2 may be rewritten for this case as:

$$S[i] = A_{pause}[i] + A_{sc}[i] \qquad \text{(EQ 3)}$$

where $A_{sc}[i]$ and $A_{pause}[i]$ are the values of the annotations for frame i of the recording. The value of $A_{sc}[i]$ is binary: 1 if a speaker change has been detected for the *ith* frame, 0 otherwise. $A_{pause}[i]$ is scaled to a value between 0 and 1 by subtracting the length of the shortest pause from all pause lengths, and then dividing each resulting pause by the length of longest resulting pause.

The weights of both annotations are set to 1.0 in Equation 3 for the following reason: if $A_{SC}[i] = 1$ (i.e. there is a speaker change at frame i), then the only frames which will be assigned a higher salience are other speaker changes which occur at longer pauses. By setting both weights to 1.0, and by virtue of the fact that pause lengths are scaled to a value between 0 and 1.0, speaker changes are always considered more salient than pauses with no associated speaker change. The overall effect of Equation 3 is to separate frames into two sets: frames at speaker changes, and frames without speaker changes. The salience of frames within each set are ordered by length of pause (the longer the pause the higher the salience). The salience of all frames in the first set (speaker changes) are guaranteed to be higher than the salience of any frame in the second set.

The salience measure is used by NewsComm to locate jump locations within a recording at any desired level of granularity. Given a position within a recording, the next jump location is chosen by finding the frame with the highest salience within the jump range. Thus the jump range controls average jump size within a recording. If the jump range is set to 0, every frame becomes a jump location. At the other extreme if the jump range is set to the size of the entire recording, only one jump location will be selected: the frame with the highest salience across the entire recording.

The jump location selection process may be thought of as sliding a window across the recording. We start with the window positioned so that the left edge of the window lines up with start of the recording (the recording is laid out from left to right). The length of the window corresponds to the jump range. To select a jump location, we find the frame within the window with maximum salience. We then slide the window over so that the left edge lines up with the jump location we just selected. We repeat this process of picking the next jump location and sliding the window until the window reaches the end of the recording. To jump backwards the window is placed to the left of the play position instead of the right and slid to the left after each jump location is chosen.

The use of the jump range concept ensures even temporal coverage of the recording. Even if all of the most salient frames of a recording are located in the first half of the recording, the framework guarantees coverage of the second half as well. This is particularly useful when the annotations are generated automatically and contain errors.

Figure 5 presents a sample set of annotations to clarify the jump selection process. The plot at the top of the figure represents a sequence of four speaker segments spoken by three speakers A, B, and C. Time is represented from left to right.

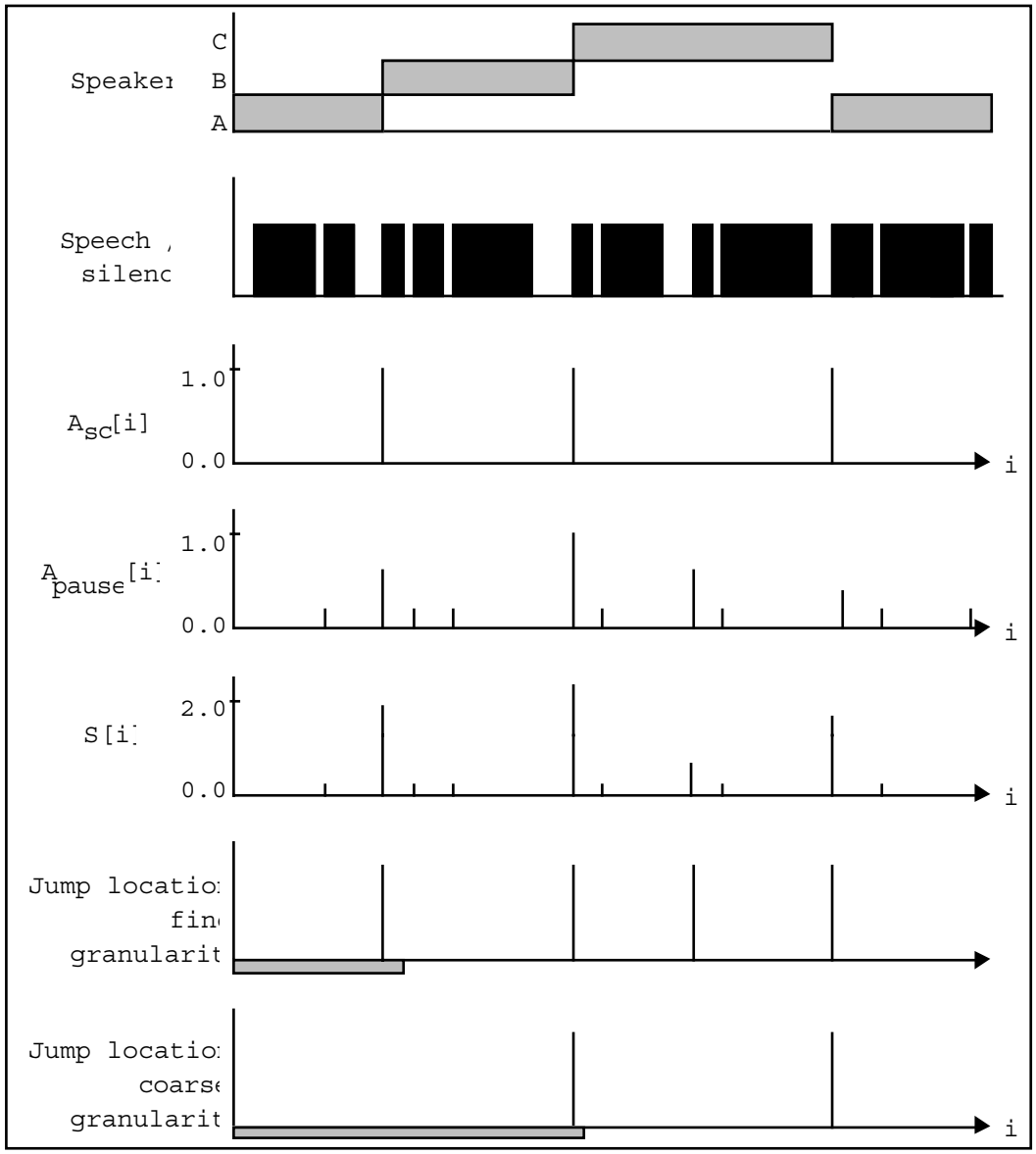

Figure 5: An example of how annotations are combined in NewsComm for a recording containing three speaker changes and several pauses.

The next plot shows speech and silence segments for the same recording. These would be found using the pause detection algorithm described in Chapter 3. The black bars indicate speech, and the white bars silence. The duration of each pause is proportional to the width of each corresponding white bar.

The next plot shows the values of $A_{sc}[i]$ for this recording. The value is binary: it is 0 for all frames except the three locations where speaker changes occur (note that they line up with the speaker transitions in the first plot). This information would be automatically extracted by the speaker indexing algorithm.

The fourth plot shows the values of $A_{pause}[i]$. The value is proportional to the length of the corresponding pause shown in speech/silence plot. Note that only the last frame of each pause is assigned a non-zero value.

The fifth plot is $S[i]$ as defined by Equation 3. Notice that the range of values doubles to (0,2.0). The highest values in $S[i]$ are at the three speaker changes, and these values are further ordered by the pauses which coincide with each change. In this example the second speaker change has the highest salience, followed by the first and then the third.

The final two plots show jump locations which would be chosen using two different jump range settings. The first of these plots shows jump locations using a relatively short jump range, and thus a fine granularity. The thin gray bar at the bottom left of the plot shows the size of the jump range. Using this jump range, four jump locations are chosen: the first, second, and fourth are the speaker change boundaries, and the third is a long pause.

The final plot shows the two jump locations chosen using a coarse granularity jump range. In this case for the initial jump selection two speaker changes are included in the look-ahead window, and the second is chosen since $A_{pause}[i]$ and thus $S[i]$ has a higher value at that frame.

The Effect of Jump Granularity on Story Boundary Detection in BBC Newscasts

An experiment has been conducted to study the effect of jump granularity on the number of story boundaries identified as jump locations by the framework. Story boundaries are desirable points to locate in a newscast since the user can browse the recording by jumping between stories[5]. The locations of all story boundaries in

[5]In the future we would expect newscasts to be manually annotated in the news office which produces it, but since this is not presently the case newscast provides a good case for testing the framework.

four 20-minute BBC newscasts were manually annotated. A jump location is considered to coincide with (or hit) a story boundary if they occur less than 1.0 seconds apart.

Ideally the jump locations would coincide with only story boundaries. The assumption, based on empirical observations of the newscasts, is that speaker changes and long pauses usually coincide with story boundaries. Figure 6 shows the results on the four 20-minute BBC newscasts. The line marked with squares shows the percentage of story boundaries located as a function of the jump range. As expected the two are inversely related. The line marked with diamonds shows the false alarm rate of the jump locations; the false alarm rate is the percentage of all jump locations which do not occur at story boundaries.

The false alarm rate dips at a jump range of 60 seconds. This is a reasonable jump range setting to use for accessing this type of recordings since the false alarm rate is at a minimum (70%), and the story hit percentage is relatively high (67%).

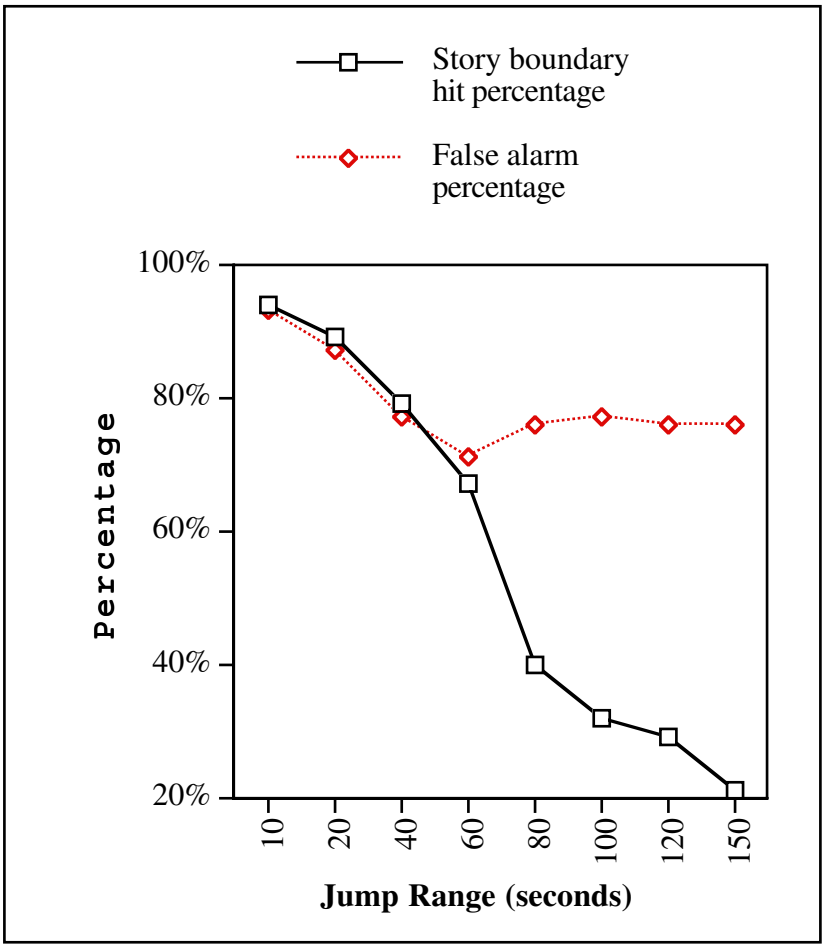

Figure 6: A plot of the number of story boundaries in the BBC test newscasts versus the jump range of the annotation framework.

Chapter 5
Design of the Hand-held Interface

This chapter describes the design of a series of navigation interfaces for the hand-held playback device. Section 5.1 outlines the objectives of the design. Section 5.2 outlines the usability testing procedure which was used to test the designs. Section 5.3 summarizes the design results, and finally Section 5.4 presents a detailed description of the five iterations of the interface design.

The design and implementation of the interfaces are treated as separate issues in this thesis. The first and fifth versions of the interface design described in this chapter were implemented in hardware, and the second, third, and fourth interfaces were implemented in software in an X-windows environment. This chapter discusses the design of the interfaces from a functional perspective. Chapter 6 describes the implementation details.

5.1 Design Objectives

The following objectives were set for the hand-held interface design:

- Enable natural and efficient navigation of audio recordings
- Be small enough to be held and controlled with one hand
- Require little or no visual attention to operate, especially for common operations such as jumping within a recording

- Facilitate selecting which of multiple recordings to listen to, and communicate simple requests to the server. These controls were not added until the fourth version of the design.

All of these objectives were met in the fifth iteration of the design.

5.2 Usability Testing

Two of the five designs (the second and third designs) were (relatively) thoroughly tested by users. For each of these designs, four subjects were asked to perform the following four steps:

- Visually inspect the interface and describe what the subject believes each element of the interface will do if activated.
- Explore the interface without instruction from the investigator. The purpose was to determine if the interface design is intuitive for users who have no information about how to use it.
- Perform a search task. The subject was asked to find a specific news story within a recording of a 20 minute newscast using the interface. The purpose was to determine the usability of the interface for search tasks.
- Describe features of the interface which were found easy to use or useful and features which were found difficult to use or of no use, and additional features the subject would like to have added to the interface.

Notes were taken at each step of this process and used to redesign the interface.

The first, fourth, and fifth iterations of the interface were also tested by users, but using less structured methods.

The goal of the user studies was to expose flaws in the design and determine the set of features which users find useful for the task of skimming and searching audio recordings.

5.3 Summary of the Design Results

Five versions of the hand-held interface were designed over a five-month period. The initial interface was very simple and designed as a proof of concept (mainly to

test hand-held hardware components of the system). The next two designs were progressively more complex and powerful and exposed increasing amounts of navigational control to the user. After performing usability tests on these two designs it was clear that the interface was too complex and had to be simplified. The fourth and fifth iterations were made progressively simpler. The second and fifth design are comparable in complexity, however the functionality and usability improved considerably. Casual user studies have shown that users are easily able to understand and use the final interface whereas the initial interfaces are virtually unusable.

The author found a tendency to expose increasing control to the underlying indexing abilities of the system but usability studies consistently showed that a simple interface retaining only basic indexing control was preferred.

5.4 Detailed Description of Each Design Iteration

5.4.1 Version 1

Description

The first interface was very simple as shown in Figure 7. The interface consists of only a four-position Nintendo-style controller mounted in the top panel of the hand-held's case. The controller is positioned for easy access using the right thumb. A set of four sample audio recording was loaded into the hand-held for testing the interface.

Pressing the controller up and down switches between recordings. Each time a new recording is selected, a male human voice announces the name of the selected recording. Pressing the controller right starts playing the selected recording. Once the recording is playing, pressing right or left jumps to the next or previous jump location. Jump locations were found using the framework described in Chapter 4 with the granularity set for jumps approximately 30 seconds apart. The recordings were each about 5 minutes long.

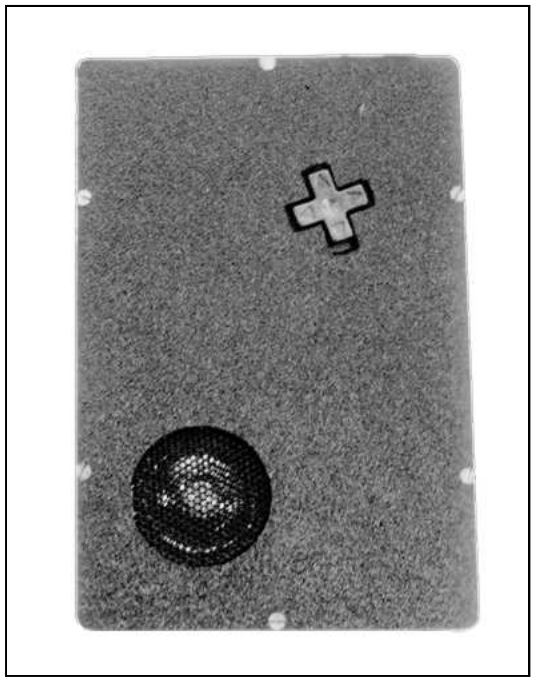

Figure 7: The first version of the hand-held has only a Nintendo-style controller (upper right) for navigation. The case is made of hard plastic and measures 6.75”(h) x 4.5”(w) x 1.5”(d).

Observations

Although the functionality of the interface was too simple for in depth user testing, one surprising result was that the controller’s position was poorly chosen. Although placed at an angle so that it would sit directly beneath the right thumb when the case is grasped from the right side, it was found to be stressful on the wrist to hold the case in this manner for extended periods of time. In later designs of the hand-held’s case, simple mock up models of the device were made from wood and foam to verify basic ergonomic factors before building the actual device.

5.4.2 Version 2

Description

The second, third and fourth iterations of the interfaces were implemented in software. These interfaces consist of a X-windows program which runs on a Sun workstation. The user can use a mouse to adjust slider, and to click on control buttons. Audio is played through the workstation’s speaker. This interface was implemented in software for rapid design prototyping as explained in Chapter 6.

We now define two terms:

highlight: a short segment of audio following a jump location (the jump locations are chosen using the framework defined in chapter 4).

skimming: a mode of playback in which only the highlights of the recording are played in sequence; the portions of the recording between highlights are skipped.

Figure 7 shows a screen capture of the interface. Five buttons are arranged in a single row in the center of the window. The name and function of each button shown in Figure 7 is defined in Table 2.

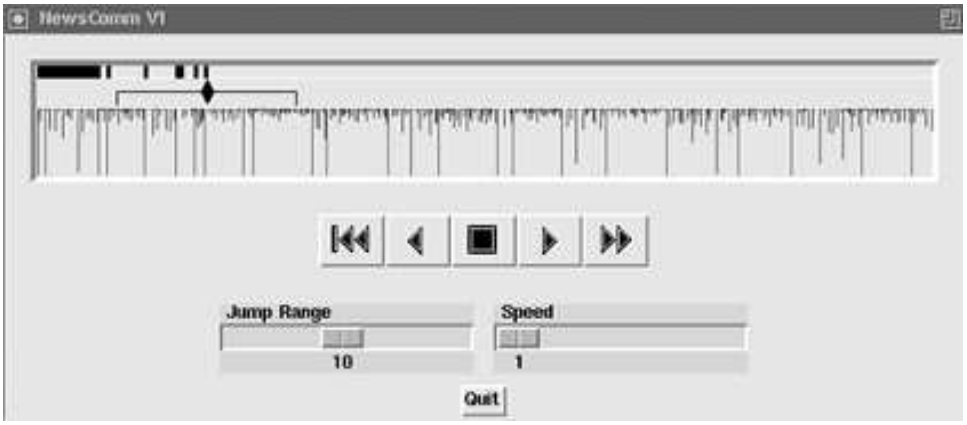

Figure 7: Version 2 of the interface includes a display of the structural description (top of window), five navigation buttons, and two sliders to control jump range and playback speed.

Button	Name and function
	REWIND: Move the play position to the beginning of the recording
	JUMP-BACK: jump back to the most salient sample within the jump range
	STOP: Stop playing
	PLAY: Start playing from the current play position; if pressed while already playing, jump forward
	SKIM: Start skimming from the current play position; if pressed while already skimming, jump to next highlight

Table 2: Navigation controls of Version 2.

When a jump is executed (by pressing either JUMP-BACK, PLAY, or SKIM) the jump location is found using the framework described in Chapter 4. The jump locations are computed on the fly relative to the current play position.

There are two sliders beneath the row of buttons. The jump range slider adjusts the size of the jump range from a minimum of 12 seconds to a maximum of 240 seconds. The speed slider adjusts the playback speed from 1.0 to 3.0 times normal. Playback speed is modified without affecting the pitch of the audio by using the synchronized overlap add method [Roucos]. The QUIT button at the bottom of the interface exits from the program.

Referring again to figure 7, the long rectangular window at the top of the interface is a visual representation of the audio recording. The small black diamond in the top left area indicates the current play position. When audio is being played, the diamond is animated and moves from left to right. The square bracket (which opens downwards and is centered on the diamond) indicates the value of the jump range; the longer the jump range, the wider the bracket.

The black bands in the narrow strip above the diamond are a trace of the parts of the recording which have already been played. As the diamond traverses an area of the display, the corresponding area of the strip is colored black.

Structural information is displayed as vertical markings in the lower portion of the audio display (under the diamond). On the workstation's color screen two colors are used to display these markings but the gray scale screen capture process used to generate Figure 7 did not retain this distinction. The full height vertical bars are displayed in red and mark speaker change locations as determined by the speaker indexing system (Chapter 3). The shorter varying length marks indicate locations of pauses. The length of each mark is proportional to the length of the associated pause.

Although one of the design objectives is to require little visual attention, this interface includes a visual display of the audio[6]. The motivation for including the display was to explore which information users found to be worth displaying. The

---

[6]In fact the entire interface is visual since it is a windows application, the buttons and sliders can be physically realized and could be operated by touch without visual attention.

final interface uses a limited text-only display which provides a subset of the information found most useful in these early designs.

Observations

The system was loaded with a 20 minute recording of a BBC news broadcast for usability studies. Overall, users found the interface difficult to use. The skimming function was poorly understood and not used during the test task. Icons were confusing, and the visual display was unclear.

After initial visual inspection, most subjects found the following:

They understood that the pause and speaker change markings were a representation of the audio but misinterpreted the pause markings as energy of the signal, and also misinterpreted the speaker change marks as manual segmentation marks.

The REWIND, STOP and PLAY buttons were understood, but the JUMP-BACK and SKIM buttons were confusing. Subjects thought the JUMP-BACK button would play the audio in reverse since the icon is the symmetrical inverse of the PLAY button. Subjects had no idea what the SKIM button might do. Some guessed it meant to fast forward to the end of the recording although they observed the absence of the vertical bar present in the REWIND button's icon.

The speed slider was consistently associated (correctly) with the play speed, but no subject had any idea what the jump-range slider might do.

After completing the visual inspection of the interface, the subjects were instructed to experiment with each of the controls and try to confirm or learn the use of each. Subjects quickly learned that the JUMP-BACK button causes the play position to jump back. All of the subjects also confirmed the meaning of the REWIND, PLAY, and STOP buttons. However, all but one of the subjects failed to understand the skimming function or use the jump-range slider.

Three of the users discovered that clicking the PLAY button while already playing triggered a jump forward, although they found it impossible to predict where the play position would jump (they had the same problem with the JUMP-BACK button). This caused considerable frustration since the subjects could see the pause and speaker change marks (i.e. the visual display), but because they did not

understand the jump selection algorithm, they could not consistently predict the location of each jump.

In the third stage of the usability test, subjects were instructed to locate three specific stories within the recording. All subjects used a similar strategy: they pressed rewind to reset the play position and then used a combination of play-speed (slider), PLAY, JUMP-BACK, and jump-forward (by clicking PLAY while playing). None of the subjects used the skim mode or adjusted the jump range slider.

Finally subjects were asked what they would like change in the interface. They all asked for direct manipulation of the play position by clicking and dragging the diamond icon to a different position. Clearly, direct control of the play position is important. This is similar to the results Arons found after conducting usability studies of SpeechSkimmer [Arons]. Arons found that users wanted some indication of where within recording the current play position was, and some method of control to jump directly to a specific location in the recording.

5.4.3 Version 3

Description

Figure 8 shows the redesign of the interface based on the usability studies on Version 2. Several changes were made:

The audio display window was removed since the target hand-held hardware will not have a large display screen (it was included in the original interface to find out what information is important for users to see). Also, it was predicted that without the display of the structural information users would be less likely to attempt to predict jump locations, thereby reducing frustration.

A small text window was added which displays the current mode (stop, play, or skim), the current play position as “minutes:seconds” and the total duration of the recording (in the same xx:yy format). In figure 8 the play position is at 00:00 (start of recording) and the recording is 20 minutes long (20:00). The mode information was added to the display to help reduce confusion caused by having a hidden skim mode (entered when the user presses the SKIM button).

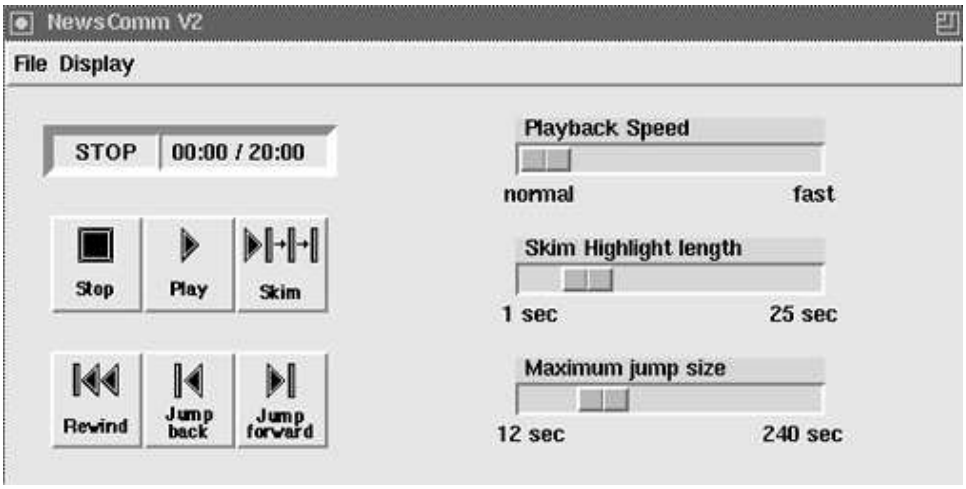

Figure 8: In Version 3 the display was reduced to a small text display (above the navigation buttons). The functionality, icons, and layout of the navigation buttons were redesigned; a new slider for controlling highlight length was added.

A JUMP-FORWARD button was added to complement the JUMP-BACK button. The jump-forward function of the PLAY/SKIM buttons when pressed while already playing/skimming was removed.

A new slider was added to control the duration of highlights.

Several icons and labels were modified:

- A vertical bar was added to the JUMP-BACK button to differentiate it from the play button; the reverse symmetrical icon was used for JUMP-FORWARD
- The SKIM icon was changed to a series of bars and "skip arrows" to differentiate it from the REWIND icon
- Written labels were added to each button
- Names of sliders were expanded for clarity, and use similar terms (skip, jump) as buttons with related functions
- Range labels were added to the sliders

The layout of buttons and sliders was rearranged into three clusters. The first cluster consists of the mode buttons: STOP, PLAY, and SKIM. The second cluster consists of navigation buttons: REWIND, JUMP-BACK, and JUMP-FORWARD. Finally, the sliders were placed together on the right. The sliders are

vertically ordered so that the sliders and the associated buttons are close to one another (for example, the play-speed slider effects PLAY and SKIM, so they are placed in close proximity; skim-highlight-length slider effects SKIM so they are close together; jump-size slider effects SKIM, JUMP-FORWARD, and JUMP-BACK so they are close together).

Observations

A second set of four subjects were tested with the new design. There was a clear improvement in the usability of the buttons. Subjects quickly understood the distinction between modal and navigational control, although the function of the skim mode took some time to understand. All four subjects had varying amounts of trouble understanding the function of both the highlight-length and jump-range sliders.

Some subjects did not notice the contents of the single line text display. Once pointed out to them, they were able to guess the meaning of the display.

Subjects had several suggestions for how to improve the interface. Similar to Version 2 subjects' requests for direct manipulation of the play position diamond, subjects who tested Version 3 wanted direct control of the play position through a continuous scan control similar to that found on conventional compact disc players.

Once the function of the highlight and jump-range sliders was explained, most subjects agreed that although useful, these sliders exposed more control than necessary (at the expense of complicating the interface). Two subjects suggested replacing the sliders with buttons which could cycle through some limited number of optimal settings.

5.4.4 Version 4

Description

In the fourth version of the interface, the three mode buttons and three navigation buttons were left unaltered since they were successful in user tests of Version 3. As shown in Figure 9, the changes made are:

The sliders were replaced with two buttons labeled PLAY-SPEED and SKIM-MODE. Pressing PLAY-SPEED cycles the play speed through three settings (1.0,

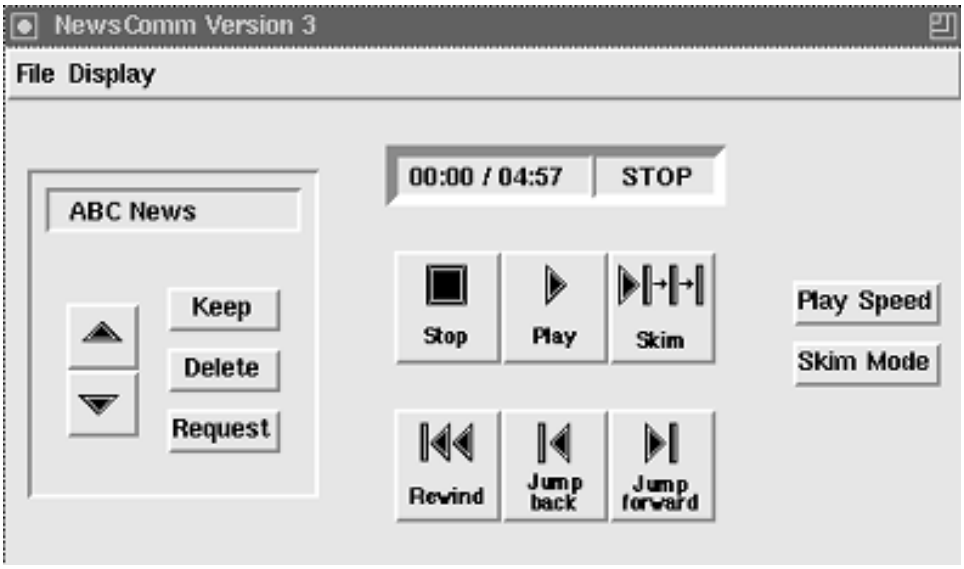

Figure 9: In Version 4 the sliders which control playback speed and skimming parameters were replaced with two buttons which cycle through preset (discrete) settings of the three parameters. The display and navigation controls were not modified. A new set of controls and display were added (on the left) to select and manage multiple audio recordings.

1.4, and 1.8 times faster than normal speed). SKIM-MODE cycles through the combinations of settings of the highlight-length and jump-range. The three skim mode settings roughly correspond to skimming with fine, medium, and course granularity.

A new set of buttons and line of display was added to support multiple recordings. The display shows the name of the currently playing recording. The up and down arrow buttons (on the left) enable the user to move through a list of available recordings. The interface announces the name of the new recording using prerecorded audio. For example when the user presses the down button and selects the BBC newscast, the interface plays a short audio file which says, “B-B-C news”. If the file is a summary file, the interface prefixes the menu name with “Summary of” (summary files are explained below).

The KEEP and DELETE buttons let the user mark which recordings should be retained and discarded at the next connection with the server. The REQUEST button is used to request full versions of recordings when only a summary of the recording has been downloaded. These three buttons could not be tested in usability studies of Version 4 since the audio server had not been implemented yet.

Automatic Summaries of Recordings

Summaries were introduced in this version. A summary of a recording is a concatenation of only the highlights of the original recording, with 200ms 200Hz tones separating each highlight. The summary of recordings are generated by the audio processor module of the audio server. The result is an automatically extracted summary which is usually sufficient for the listener to decide whether he wants to download the entire recording during the next connection to the server.

Observations

Tests of Version 4 suggested that the skim mode was still unclear, and users continued to feel a need for more direct navigational control.

The dynamic generation of jump locations (based on the current play position) caused confusion since jumping back twice from slightly different originating locations will not necessarily land the play position in the same place. The result is that the jumping function seems to have inconsistent behavior.

5.4.5 Version 5

Description

Version 5 of the interface was implemented in hardware in two forms shown in Figure 10a and Figure 10b. Three main changes were made to the interface:

- The skim mode was eliminated
- A set of four direct navigation buttons were added which enable coarse and fine granularity jumps
- Jump locations were precomputed and fixed rather than dynamically computed on the fly

The function of the recording selection and management controls were not changed but all of the icons were redesigned as shown in Figure 10b and Table 3. Table 3 describes the controls of both implementations of Version 5.

The granularity of jumps tied to the COARSE-JUMP buttons depends on the type of audio being played and is set manually in the audio server. For example, a granularity of approximately 30 second jumps using a 60 second jump range was found appropriate for navigating newscasts as described in Section 4.2. The FINE-

JUMP buttons are tied to jumps roughly 5 seconds apart and usually correspond to grammatical sentence breaks.

Although the skim mode was removed, a similar function can be achieved by holding down any of the jump buttons; doing so plays 0.5 second segments of the recording and then jumps automatically to the next jump location. This is similar to the scan operation of a CD player and is understood easily by users.

Observations

Tests of this interface have indicated that the four button interactive navigation is preferred to an automatic skim mode since the system never initiates a jump automatically (which causes a feeling of lack of control over navigation). The two levels of granularity seem sufficient for navigation; there seems to be no need for the more elaborate control over the jump algorithm afforded by the earlier more complex interfaces.

The fixed jump locations was a major improvement to the usability of the device. The locations served as landmarks within the recording which could be revisited as points of reference. One person picked up the leather cased hand-held and immediately started navigating through a newscast, and after a minute of use exclaimed, “I can actually use it! And if I can use it, anyone can!” [Driscoll].

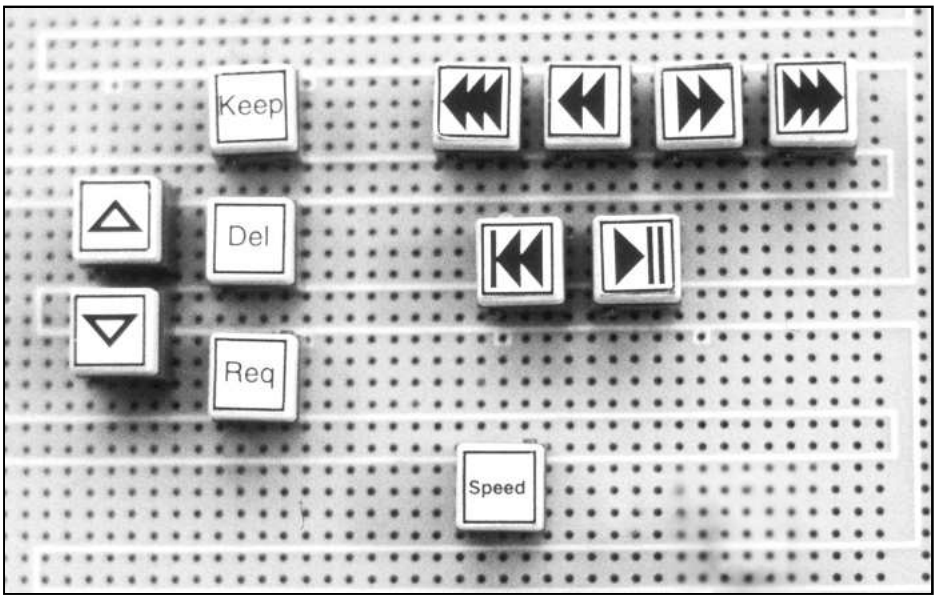

Figure 10a: The keypad interface of Version 5. See the text for descriptions of the function of each control (a photo of the complete hardware system can be found in Chapter 6, Figure 13a).

Figure 10b: Details of the top, front and right sides of the final hand-case which incorporates Version 5 of the interface design. The front panel houses the recording selection and management controls and a 2x16 character LCD screen, and the right side houses the navigation controls. The opening at the bottom of the right side provides access to the memory card eject slider. The case is made of soft leather and measures 7.5”(h) x 3.75”(w) x 2.0”(d).

Flat mount	Leather case	Name and function of button
		COARSE-JUMP-BACK: Jump back at coarse granularity; hold down to do a coarse level backward skim
		FINE-JUMP-BACK: Jump back at fine granularity; hold down to do a fine level forward skim
		FINE-JUMP-FORWARD: Jump forward at fine granularity; hold down to do a fine level forward skim
		COARSE-JUMP-FORWARD: Jump forward at coarse granularity; hold down to do a coarse level forward skim
		REWIND: Move the play position to start of the recording
		PLAY/PAUSE: Toggle between playing and stop
		SPEED: Designed to switch between three preset playback speeds, but not currently implemented
		MENU-UP: Select the next recording in the menu
		MENU-DOWN: Select the previous recording in the menu
		KEEP: Keep the current recording at next connection with the audio server
		DELETE: Delete the current recording at next connection with the audio server
		REQUEST: Download the full version of the current recording (this button works only when the current recording is a summary).
		DOCK: Press this button before connecting the hand-held to the audio server, and again after disconnecting

Table 3: Icons, names and functions of the buttons of Version 5. The interface was implemented in two versions of hardware (the first was mounted flat, the second was in a leather case); the icons for each implementation are listed in the columns on the left.

Chapter 6
Implementation of the Hand-Held

This chapter describes each stage of implementation of the hand-held interface. The first version was a hardware implementation, built to evaluate the HP95LX palmtop computer as a platform for development. After this initial version was functional, a software-only graphical interface was implemented to facilitate rapid design. After three iterations of designing and testing the software interface, a hardware version of the final interface was built. This hardware version was then further tested and modified. Finally, a case was designed and sewn from leather to house the hardware in an ergonomically pleasing form.

6.1 Software Implementations

Early interface design was done using a software-only implementation on a Sun Sparc station to facilitate rapid development. Tcl was used to generate the graphical components of the interface [Ousterhout], and a C program was used to provide the underlying functionality. Audio functions such as playing selected segments of sounds files were achieved by using the Audio Server developed by the Speech Group at the Media Lab [Schmandt].

The Tcl/C based simulation proved to be an extremely efficient way to design the hand-held's interface. The Tcl component of the simulation guided the choice of buttons and LCD display used in the hardware design, and about 80% of the C

code from the software simulation was later reused in the hardware implementation[7].

6.2 Hardware Implementations

Three versions of the hand-held have been implemented in hardware. The first is a proof of concept design done mainly to verify that the HP95 would serve the purposes of this thesis. The second is a port of the final software interface (Version 4 of the interface as described in Chapter 5). Some modifications were made to the navigation controls during this port. This hardware implementation is not meant for hand-held use; the components are all attached to a flat board and provide a platform for developing the hand-held's software. The third hardware implementation is the final leather encased hand-held device pictured at the beginning of this thesis.

A basic design decision for all of the hardware designs was to put local memory in the hand-held for audio storage since it is relatively simple to implement. Alternatively the device could be a cellular telephone type device which uses a two way point to point wireless connection to relay audio from the server and send navigation commands back. This wireless model is more difficult to implement but is certainly a feasible alternative for realizing the NewsComm system. In fact, in a commercial version it may be cheaper to use the wireless model if there are enough users.

6.2.1 Version 1: Proof of Concept

The first version of the hand-held hardware was built as an exercise to determine:

- The appropriate embedded controller
- The easiest method to transfer audio from the server
- The necessary software and hardware to play audio
- How to do keypad input with the chosen controllerx

[7]The C code was recompiled using a 8088 compatible compiler so that program could be run on the PC-XT architecture embedded controller of the hand-held.

The HP95LX palmtop computer made by Hewlett-Packard (shown Figure 11) was chosen as the controller for the device for two main reasons. First, the HP95 is the smallest commercially available stand-alone PC-XT compatible computer; this is important since conventional PC programming tools can be used for software development. Second, the HP95 has an 8-bit digital-to-analog converter (DAC) built in so it has the capability to produce telephone quality audio. In later versions of the palmtop (the HP100LX and HP200LX) this DAC circuitry was removed since it was underutilized by commercial software.

Audio is stored in a 20MB flash RAM card which is installed in the HP95's PCMCIA slot. The 20MB card can hold up to 40 minutes of 8kHz 8bit uncompressed audio[8].

Figure 11: The HP95LX is an 11 ounce palmtop computer. It has a PC-XT architecture, contains a 80C88 compatible CPU, 512K of system RAM, an 8 bit digital-to-analog converter, a type II/III PCMCIA port, and a RS-232 serial port.

[8]At the time this thesis was written the highest density flash RAM card available was 80MB which would increase the capacity of the device to 160 minute of uncompressed audio.

A pentium PC was used to compile C code and assemble assembly code written for the HP95. The compiled programs and audio recordings were transferred from the PC to the HP95 using the RS-232 serial port of each computer.

Figures 12a and 12b show the external and internal views of this implementation. The case is made from ABS plastic. The six sides were cut, machined, and then glued together using an epoxy adhesive. The top plate is attached using six removable machine screws. A Nintendo controller was removed from a commercial game controller and mounted on the top panel of the case. Figure 12c shows the components in the device. The controller is connected to the HP95 by tapping the keyboard connections of the HP95 and connecting the joystick in parallel to four of the keys of the keyboard.

A low pass filter and amplifier is used to filter and amplify the output of the HP95 DAC before driving a small 8 ohm speaker which is also mounted on the top panel of the case. The low pass filter is a passive RC circuit, and the amplifier was extracted from a commercially available powered speaker.

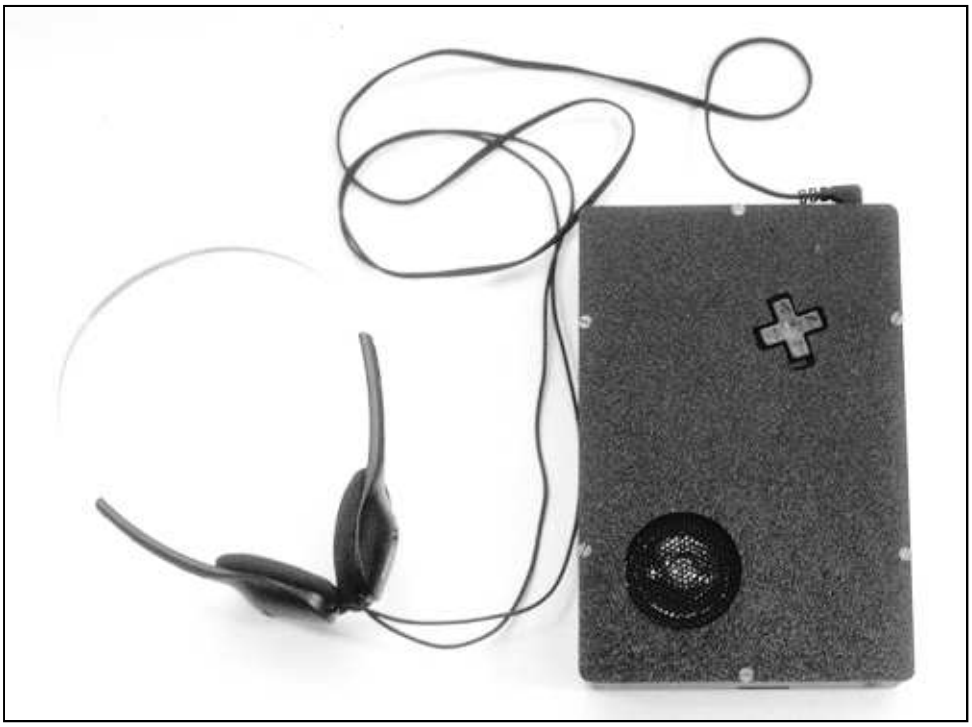

Figure 12a: The first hardware implementation of the hand-held was housed in a custom made hard plastic case. The Nintendo controller was mounted in the upper right corner to enable easy access to the right-hand thumb. A small speaker is mounted on the lower left.

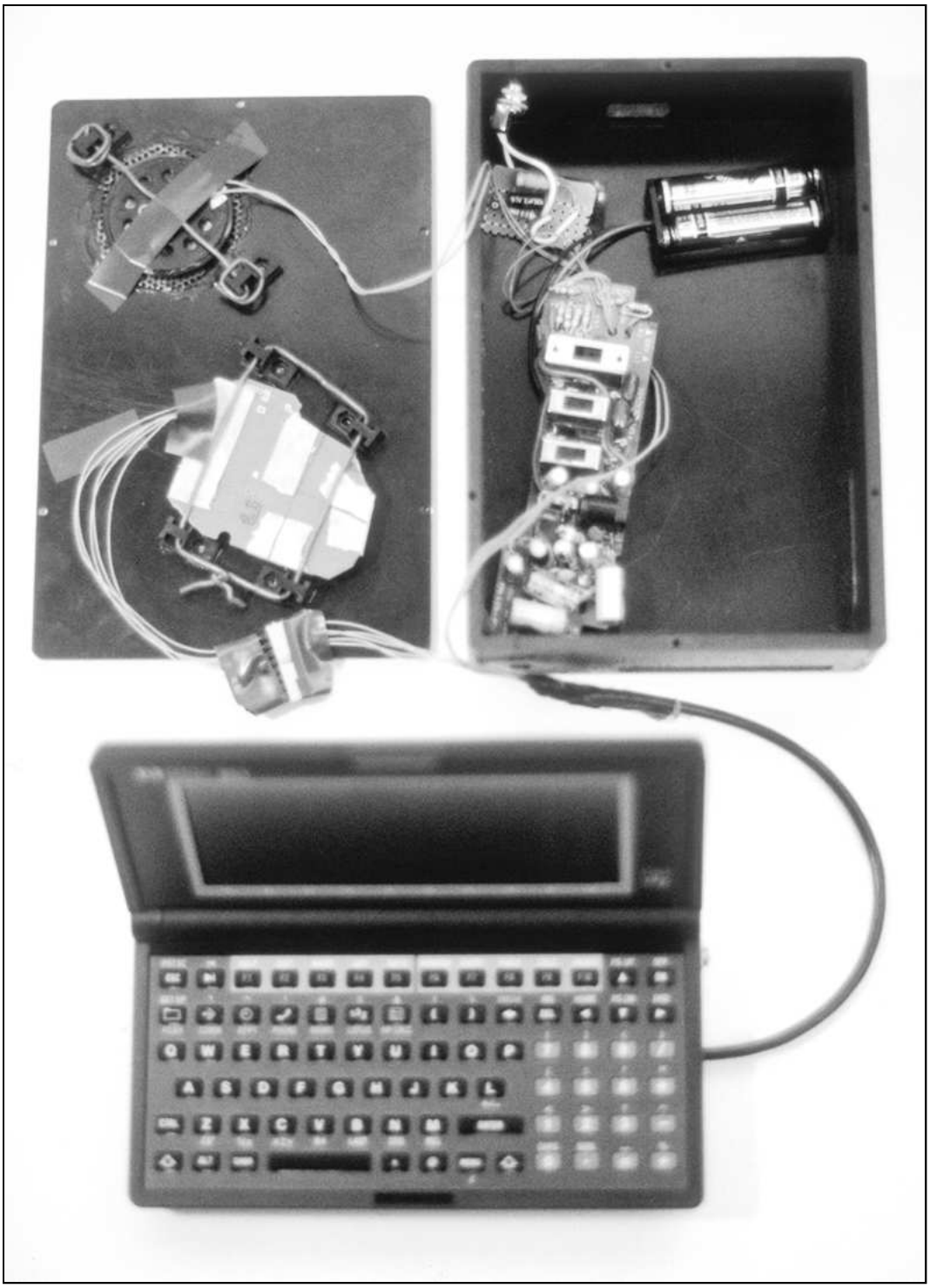

Figure 12b: Inside the plastic case of Version 1. The top panel is on the upper left and houses the speaker (top) and Nintendo controller. The case (upper right) contains an audio jack for connecting headphones, low pass filter, amplifier and 3V battery pack. A cable connects the output of the HP95's DAC to the audio amp, and taps from the HP95's keyboard to the controller.

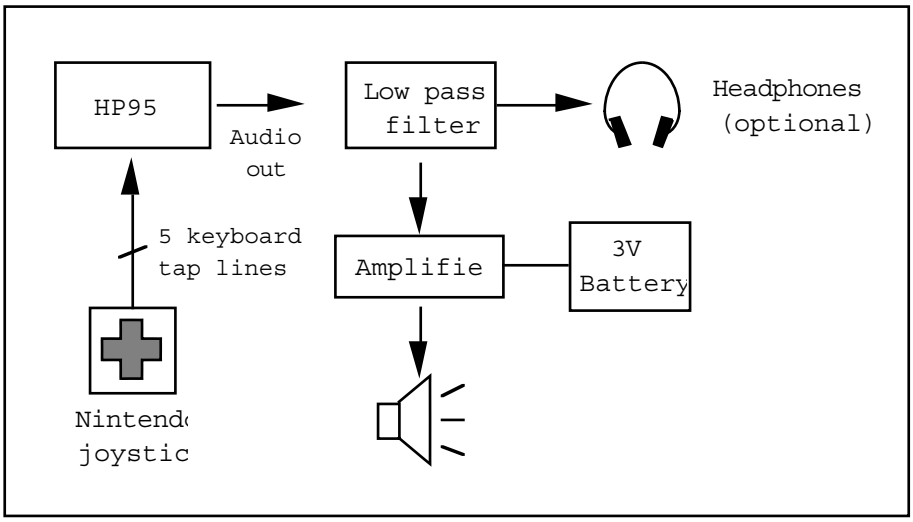

Figure 12c: Components in Version 1 of the hardware hand-held. The Nintendo joystick is connected in parallel to the keyboard of the HP95. The audio output from the HP95 is passed through a low pass filter and amplifier and connected to a small speaker. Optionally the listener can connect a pair of headphones for private listening.

The audio driver for the HP95 is written in 8088 assembly language. When passed a pointer to an audio file, it plays the entire file or return when a key is hit on the keyboard. The driver sets up a timer interrupt equal to the sampling period (1/8000 seconds) and sends the each byte from the audio file to the DAC.

Evaluation of Version 1

Several lessons were learned from building this prototype:

- The RS-232 port is too slow for transferring audio (with a maximum transfer rate of 19,200 baud).
- The keyboard connection was extremely unreliable; the HP95 crashes about every 30 minutes when the Nintendo controller is attached. This is probably due to noise picked up through the unshielded hook-up wires.
- The placement of the Nintendo controller is awkward: the device cannot be used for long periods of time due to wrist strain.
- The only way to access the HP95 is by removing the screws which hold the top plate in place. This is slow and awkward.

6.2.2 Version 2: Porting the Software Interface to Hardware

Hardware Description

Figure 13a shows the second hardware implementation of the hand-held, and Figure 13b describes the components and connections between them.

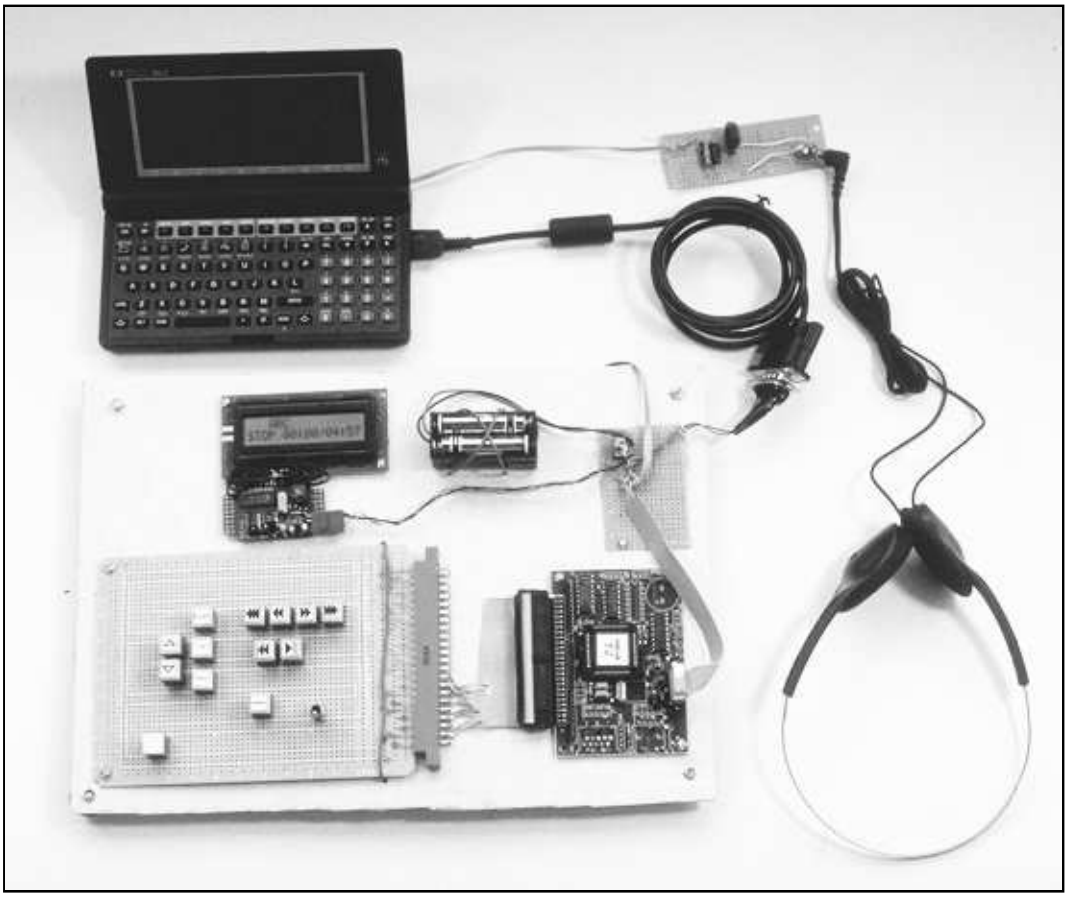

Figure 13a: In the second hardware prototype the components are attached to flat piece of cardboard and serve as a test bed for software development. The HP95 is shown in the top left; the headphones are connected to the HP95 via a low pass filter; the custom keypad, LCD module, encoder, decoder and battery pack are attached to a piece of cardboard. The only connection from the display and keypad to the HP95 is through an RS-232 serial cable. Note that all components are battery operated and the system is "untethered" (i.e. no connections to external power supplies or host computers).

In version 2 no enclosing case was built; the components are connected using flexible cables so that they can lie flat as shown in photograph 13a. The HP95 is connected to an LCD display and a keypad through the RS-232 serial port.

The keypad consists of a set of tactile keyboard type buttons mounted on a prototyping board. The keypad is wired in a matrix configuration and connected to

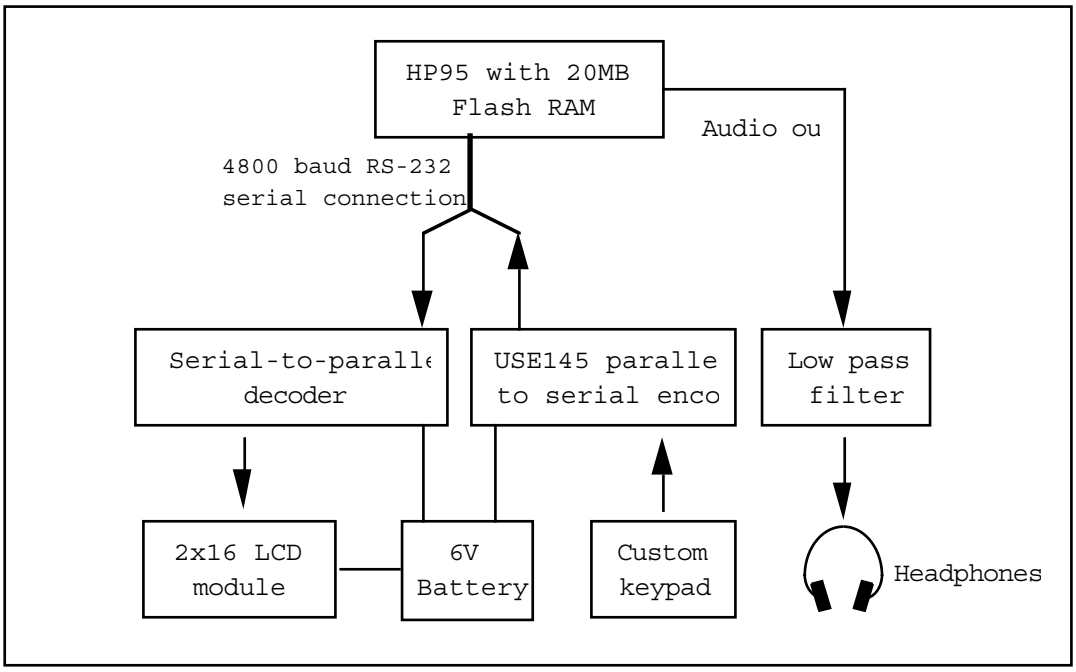

Figure 13b: Components of Version 2 include the HP95, the LCD display and decoder, the keypad and encoder, audio filter and headphones, and battery.

the USE145 parallel to serial encoder. The USE145 generates unique ASCII codes in RS-232 format for each key on the keypad. The serial link avoids the noise problem encountered with the keyboard tap connections made in the first prototype.

The tactile buttons which were selected for the keypad require a reasonable amount of force to activate. This guards against accidental key presses. This is important in a non-visual interface where the user's fingers may rest lightly on the keys for orientation before one of the buttons is actually pressed. The buttons have a click feel which provides feedback when they have been successfully pressed.

A 2x16 character LCD module is connected to a serial to parallel decoder. The module is used to display the name and duration of the current recording, the current play position, and the mode of operation (SKIM, PLAY, and STOP). The display and keypad are both 4800 baud RS-232 devices; they are connected to the HP95's serial port.

The amplifier and speaker used in the first prototype were removed because the quality of audio that a small speaker can produce is not sufficient to justify the additional space they require. The audio output of the HP is passed through a low pass filter and connected to a female 1/8" mono RCA jack. Figure 13a shows a

pair of walkman headphones connected to the jack. Powered speakers can also be attached.

A PCMCIA adapter was installed in the host PC so that data could be transferred faster. Using the flash RAM card, data can be transferred by physically inserting the flash card into the PCMCIA drive of the PC and writing to it. The card is then removed and re-inserted into the HP95. This solved the slow transfer rate problem encountered in Version 1.

The audio server runs on the same PC and uses the memory card as the transfer mechanism for uploading preferences and history information, and for downloading audio recordings.

Software Modifications

The software for audio playback was largely unmodified from version 1. Serial I/O routines were written in assembly language to read the USE145 output, and to update the LCD screen.

The navigation logic software (which determines the behavior of the device when each control button is pressed) was ported from the Sun workstation software interface. This portion of the code was written in modular C on the Sun and was recompiled without major modifications for the HP95. All links to the Tcl interface were replaced with links to the serial I/O routines on the HP95. Similarly, all calls to the Sun audio drivers were replaced with calls to the audio routines written for the HP95.

6.2.3 Version 3: The Final Hand-Held Device

Figure 14 shows the final hand-held device. The components used in Version 2 were removed from the flat back panel, and new small connectors were made to interconnect the components. All of the components including the HP95 are housed in a leather case.

The case was constructed using a combination of cardboard and leather. A sewing pattern for a single-piece leather cover was designed using a computer drawing program. The pattern was printed on paper and used to trace out and cut a single piece of leather. The seams of the case were then sewn together with nylon thread to give the leather its basic rectangular form. The case has openings for

Figure 14: Photo of the final (version 3) hand-held. The case is made of soft black leather lined with cardboard supports from the inside. The HP95 is inside the case. The buttons are low profile tactile keyboard buttons. The screen is a 2 line by 16 character LCD module. The top panel (not in view) houses the main power switch, power indicator LED, headphone jack, and two buttons (the DOCK and SPEED buttons).

access to the LCD display, buttons, and PCMCIA card. The seam on the bottom of the case is attached using a Velcro fastener enabling quick access to the HP95 and 6V battery.

The case has an internal frame made from six pieces of cardboard. All of the hardware components (buttons, circuit boards, LCD module, audio jack, and power switch) are attached to the cardboard pieces through cut-out holes. The six cardboard pieces are glued to the inside of the leather case.

The case was not made from hard plastic like version 1 because working with plastic was found to be a slow process, and there was no easy way to enable fast access to the HP95 or battery inside the case. The soft leather also has nicer look and feel.

Since the skim mode was removed in the final interface, the area of the LCD display which was used to show the modal information now displays the status of the current recording (either KEEP, DEL, or REQ indicating a status of keep, delete, or request - see Chapter 7 for an explanation of these states).

Chapter 7
The Audio Server

The audio server collects, structures, and stores audio recordings. When a hand-held is connected, the server receives the user's listening history and preferences which are used to select the next set of recordings to download to the hand-held.

Figure 15 shows the components of the NewsComm audio server. The audio processor (which runs on a Sparc 10 workstation) currently receives audio from four sources:

- A satellite dish receives newscasts from ABC Radio. Newscasts are received hourly and are 5 minutes long. Only one newscast per day is currently placed in the NewsComm server.
- A conventional FM radio receiver receives a daily 20 minute BBC newscast (rebroadcast from England by a local FM radio station).
- A series of medical journal abstracts were digitized from cassette tape and stored in the server . These journals-on-tape are commercially available and are typically purchased by physicians who listen to them while driving or performing other tasks [Hasapes].
- TechNation, a weekly talk show available over the internet [TechNation]. This talk show is an hour long and is multicast over the internet m-bone. Due to memory limitations of the hand-held prototype (40 minutes capacity), only the first 15 minutes of each show are stored in the server.

Audio recordings from each of these sources are processed by the algorithms described in Chapter 3. The newscasts are processed daily using a UNIX cron job which automatically runs pause detection and speaker indexing on the recording. The structural descriptions (pause and speaker change locations) are encoded in ASCII files and stored with the corresponding recordings in the audio library. Summaries of the medical journals and TechNation talk shows have been generated and stored as separate recordings. An index file is generated which lists all of the available recordings in the library. The audio manager uses this file to access the contents of the audio library. The library is a one gigabyte networked hard disk drive.

The audio server supports two classes of recordings: updatable, and series. Updatable recordings include newscasts, weather, and any other continuously updated information in which only the latest version of the information is usually of interest. In the current implementation the ABC and BBC newscasts are classified as updatable.

Series are ordered sets of recordings such as the chapters of a book on tape, or a sequence of interviews from a talk show. The TechNation and Medical abstracts are examples of series recordings. One-of-a-kind recordings are a special case of the series class with a set size of one.

NewsComm users can subscribe to any subset of the information sources available to the audio server. The list of sources which the user wishes to download constitutes the user's preferences and is stored in the memory card of the hand-held.

Usage history, preference information and an index of the recordings in the hand-held's local memory are all stored in a table of contents (TOC) file. The TOC is originally generated by the server and downloaded to the hand-held along with a set of audio recordings.

The TOC file is read by the hand-held after disconnecting from the server so that it knows what recordings are present in its local memory. When the hand-held is next connected (docked), it will generate a modified version of the TOC containing updated usage information (it does this by modifying the status flag of recordings,

explained below). The server then reads this TOC file and thus receives the updated usage information form the hand-held.

The audio manager runs on a Pentium PC which has access to the audio library over a local area ethernet network connection. The PC has a PCMCIA adapter attached so that it can write files onto the PCMCIA flash card from the hand-held. To make a connection between the hand-held and the audio server, the user must perform the following steps:

- Press the DOCK button on the hand-held; this causes the hand-held to generate an updated TOC file and save it onto its PCMCIA flash RAM card
- Eject the PCMCIA card from the hand-held and insert it into the PC's adapter
- Initiate a download session on the PC (by running a program); the PC reads the TOC file from the flash card, and uses the preferences and usage history to decide which recordings to download; it then copies the selected files to the PCMCIA card, and generates a new TOC file
- Remove the PCMCIA card from the PC and re-insert it into the hand-held
- Press the DOCK button on the hand-held again; at this point the hand-held re-reads the TOC file and is ready for use

An Example of a Server Connection

This section presents the results of an actual interaction between the server and hand-held. In the initial state the TOC file in the hand-held contains the following information:

Preferences: ABC News, BBC News, TechNation, Emergency Medicine
Memory capacity: 19500K
Number of recordings: 4

Name	Size	Type	Status	Class	Index
ABC News (May 5)	2520000	complete	delete	update	1
BBC News (May 5)	9600032	complete	delete	update	2
Summary of Emergency Med #1	595351	summary	keep	series: emerg_med	1
Summary of TechNation #1	753528	summary	keep	series: tech	1

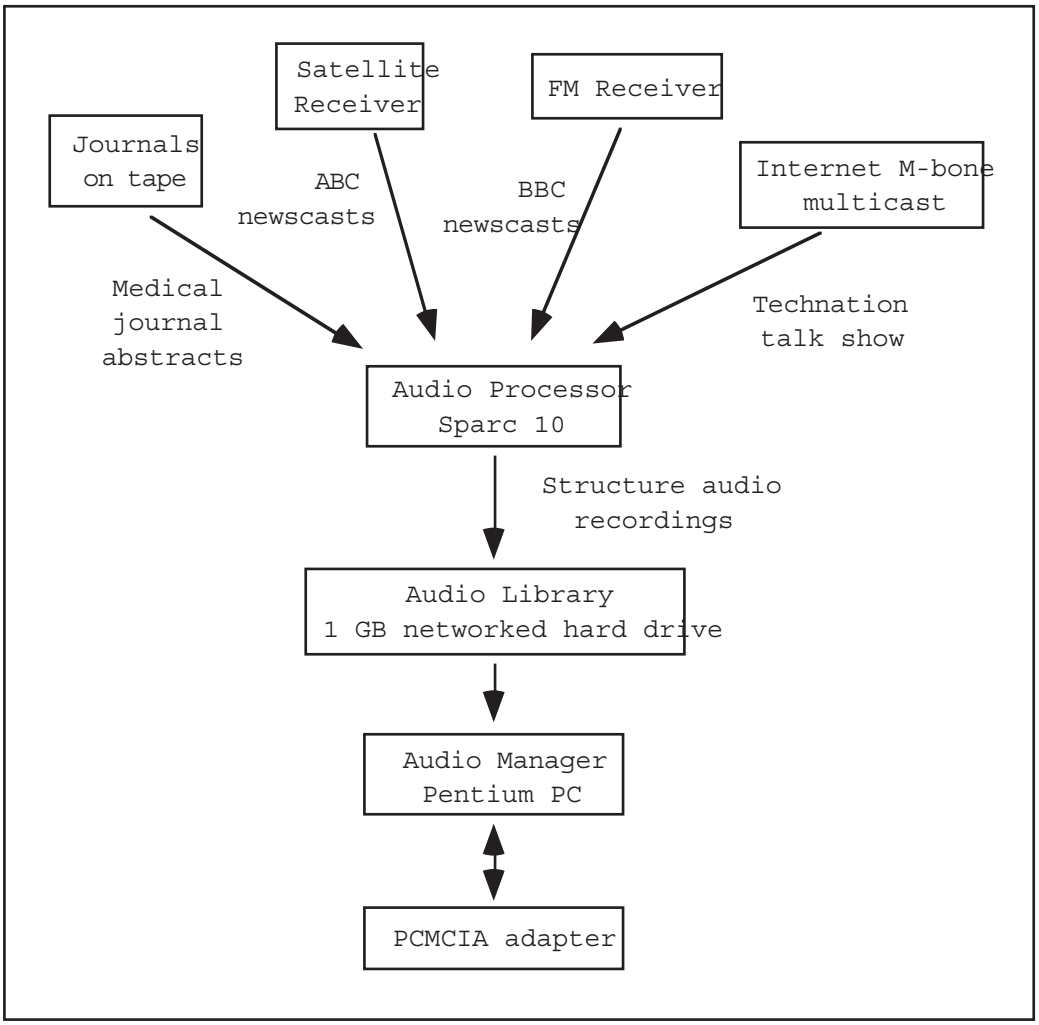

Figure 15: Components of the current NewsComm audio server. The audio processor which locates pauses and speaker changes runs on a Sparc 10 workstation. The resulting structured audio is stored on a 1 gigabyte networked drive (the audio library). The audio manager runs on a Pentium PC which downloads audio to the hand-held by copying audio files from the library to the hand-held's PCMCIA flash RAM card via the PCMCIA adapter which is connected to the PC.

The first line lists the sources of information the user wishes to receive audio from. The second line specifies the amount of free space in the hand-held's local memory (in this case the hand-held had a 20MB flash RAM card with 19.5MB available for audio storage). The next line specifies the number of recordings currently in local memory.

The table specifies the following information about each of these four recordings:

- **Name**
- **Size** in bytes
- **Type** - either a complete recording or an automatically generated summary
- **Status** - if *keep*, it will be retained at next server connection, if *delete* it will be deleted at next connection, and if *request*, the complete version of the recording will be downloaded at next connection (only summaries can have the status *request*)
- **Source** specifies if the recording is updatable or a series. If it is a series, the series name is specified as well
- **Index** specifies the source of news for news files (1=ABC, 2=BBC in this example); if the recording is part of a series, it specifies the index number of the recording within the series

In this example the hand-held initially contains four recordings: the May 5th ABC newscast, the May 5th BBC newscast, a summary of TechNation #1, and a summary of Emergency Med #1. Notice the status of each recording which is the default set by the server. News recordings by default are deleted and overwritten with the newest news, thus the status is initialized to *delete* for both newscasts. In contrast, both summaries are defaulted to *keep*.

The user listens to the recordings, and does the following:

- Presses the KEEP button for BBC News
- Presses the REQUEST button for Emergency Med #1
- Presses the DOCK button

The hand-held modifies the TOC file so that it now contains the following:

Preferences: ABC News, BBC News, TechNation, Emergency Medicine
Memory capacity: 19500K
Number of recordings: 4

Name	Size	Type	Status	Source	Index
ABC News (May 5)	2520000	complete	delete	update	1
BBC News (May 5)	9600032	complete	keep	update	2
Summary of Emergency Med #1	595351	summary	request	series: emerg_med	1
Summary of TechNation #1	753528	summary	keep	series: tech	1

The only information which has changed in the TOC from initial state is the status of the BBC and Emergency Med files. The status of these files have been changed to *keep* and *request* respectively. The hand-held is then connected to the server (by removing the flash RAM card and inserting it into the PCMCIA adapter on the server) and the server program reads in the TOC file and downloads a new set of audio. The following is the status information generated by the server during this connection:

```
NewsComm Server

Subscribed to news source "ABC News"
Subscribed to news source "BBC News"
Subscribed to series "Tech Nation"
Subscribed to series "Emergency Med"

Hand-held memory size: 19500K

Updating news recordings:
Retaining "ABC News"; 16980K left
Downloading current "BBC News"; 7379K remaining

Servicing series recordings:
Downloading full version of "Emergency Med #1"; 5363K
   remaining
Retaining "Tech Nation #1"; 4610K remaining
Downloading summary #2 from series "Emergency_Med"; 4183K
   remaining
Downloading summary #2 from series "Tech_Nation"; 3684K
   remaining
```

The server first verifies which sources of audio the user's preference list specifies. It then echoes the local memory size of the hand-held. Next, the news recordings are updated. In this example, the BBC newscast is left intact since it had the status *keep*. The ABC newscast is overwritten with a newer recording. The final block of status statements describe downloads for the TechNation and Emergency Med series. The "Emergency Med #1" was requested by the user, so the complete version of the summary is downloaded. The TechNation #1 was neither requested nor deleted, so it is left intact. Finally, two new summaries, one from each series, are downloaded automatically. The resulting TOC file which is generated by the server is shown below:

Preferences: ABC News, BBC News, TechNation, Emergency Medicine
Memory capacity: 19500K
Number of recordings: 6

Name	Size	Type	Status	Source	Index
ABC News (May 5)	2520000	complete	delete	update	1
BBC News (May 6)	9600032	complete	keep	update	2
Emergency Med #1	2016000	complete	keep	series: emerg_med	1
Summary of Emergency Med #1	426679	summary	keep	series: Emerg_med	2
Summary of TechNation #1	753528	summary	keep	series: tech	1
Summary of TechNation #2	499608	summary	keep	series: tech	2

The server has made several changes in the TOC file:

- There are now 6 recordings listed in the table (two summaries were added).
- The ABC newscast is stale (May 5th), but the new BBC newscast has been downloaded. Note that the status of the ABC recording is still *keep*; once the user marks a recording as *keep*, it will be retained until the user hits the DELETE button.
- The summary of Emergency Med #1 has been replaced by the complete version of the file (as the size and status columns indicate).

The server downloads one new summary from each subscribed series at each connection time as a way of offering new content.

Chapter 8
Conclusions and Future Work

8.1 Conclusions

This thesis has presented the design, testing, and implementation of a new audio information delivery system. A hand-held audio playback device has been developed which enables users to interactively access structured audio recordings using a simple button interface. To support the hand-held, an audio server has been built which structures and stores audio recordings collected from various sources. Some of the original contributions of this work are:

- An algorithm for separating and indexing speakers in a recording using neural net clustering
- A framework for combining multiple annotations of a media stream
- A hand-held interface for navigating structured audio

Usability tests were essential in guiding the design of the hand-held interface. An important lesson learned from these tests was that although the author had a tendency to design powerful but complex interfaces, in most cases users preferred a less powerful but simple interface. For example, early versions of the interface include a skimming mode which automatically plays only selected highlights of a recording. Additional controls enable the user to adjusts parameters of the skimming mode. Although users generally agreed that the skimming mode is useful, they consistently preferred a simpler interface with no second modality; the users could achieve the same affect of skimming by manually pressing a jump

forward button after hearing each highlight. Although this method of skimming requires more input from the user, it reduces the confusion of having a hidden mode, and it gives direct navigation control to the user.

The speaker indexing algorithm is able to detect about half of the speaker changes in speech recordings. Several potential causes for the errors have been identified including poor initial selection of training data, and a sub-optimal representation of the audio signal. Even with the present error rates the speaker indexing algorithm was successful in enabling efficient navigation when combined with pause locations.

The framework for combining annotations was successfully used to combine the output of the speaker indexing algorithm and pause detection. The framework was used to provide jump locations for five different interface designs demonstrating the separation of interface design from the underlying representation of the media.

Implementing the hand-held in hardware was a useful reminder of the inseparable relationship of function and form. Although the interface was first implemented in software, its usefulness in this form is limited since it relies on a visual display for output; in this environment it is more efficient to visually skim and read information rather than listen. The hardware version demonstrates a truly portable device which provides a function which a visual display system does not: interactive access to information for mobile users using a non-visual interface.

8.2 Future Work

This section outlines several directions in which the NewsComm system can be expanded.

8.2.1 Improve the Speaking Indexing Algorithm

As discussed in Section 3.3.4, two potential causes for the current errors in the speaker indexing algorithm have been identified. We are planning the following steps to improve performance:

- Add an initial clustering step to choose training data for each neural network
- Use a cepstral representation of the input signal

8.2.2 Port the Hand-held Interface to a Standard Hardware Platform

Work is underway to port the hand-held to a standard PC laptop platform so that any laptop equipped with a large hard drive and sound card can run the NewsComm system. This will enable better long-term usability testing.

The interface can also be redesigned for car use; the jump buttons could be placed on the steering wheel, and the remainder of the controls could be integrated into the dashboard similar to conventional car stereo systems.

8.2.3 Implement Time-Scale Modification

Time scale modification can be added to the hand-held device as an additional method of speeding up access to recordings.

8.2.4 Add Audio Compression

The single most expensive component of the NewsComm prototype is the flash RAM card used as local memory in the hand-held. This is motivation for adding audio compression to better use memory. Adaptive delta pulse code modulation can be used to compress audio by a factor of 2 with little degradation in sound quality. For speech recordings much higher compression factors can be attained using newer compression techniques which are specially designed for speech.

8.2.5 Add More Annotations

In this thesis only two types of annotations were demonstrated (speaker changes and pauses). The framework described in Chapter 4 can facilitate any number of annotations. Possible additional annotations include:

- Pitch based locations [Chen], [Arons]
- Topic changes (located by modeling the output of a speech recognizer run on the audio)
- Key word spotting
- Human editing

Human editing is an important annotation in particular since it is highly reliable. Simple annotations such as marking the start of new stories in a newscast or the

start of each chapter in a book on tape would be useful. Similarly the start of new songs in an album or compilation would be useful if the user was listening to music.

Other possible annotations for non-speech audio include locating:

- Music segments
- Applause
- Human speech in a mixed recording containing speech and non-speech such as a movie or television sound track.

8.2.6 Add Wireless Reception for Real-Time Data

For everyday use NewsComm would be more useful if it had a real time data channel for receiving current information such as timely broadcast news, pager messages, email, and voice mail. Timely broadcast news (newscasts, traffic and weather reports) can be received using a standard radio receiver and an analog-to-digital converter. Many FM stations in the U.S. now support RDS (Radio Data System) which is an in-band low bit rate encoding scheme. RDS equipped receivers can decode these bit streams on the fly. RDS could be used by the broadcasters to encode indexing and content descriptions which NewsComm could use to filter and properly tag incoming audio. Pages, email, voice mail, and cell phone functionality could then be supported by using FM carriers and merging the non-redundant hardware into one device; by adding an FM RDS receiver, a pager sized LCD screen, and audio recording hardware, NewsComm could provide all of these functions.

8.2.7 Supporting Richer Listening History and Preferences

The current implementation of NewsComm supports only simple notions of user history and preferences. In the future, the user history can include tracking which portions of a recording are skipped and which portions are heard; user preferences could include topics and keywords which are of interest. As the preferences are made more complex, the problem of a user interface for enabling the user to specify and review the information becomes more important as well.

8.2.8 Add Recording Facilities to the Hand-Held to Enable Community Reporting

A far reaching extension of NewsComm is to support recording in the hand-held device. This would enable community recording since users could record and then upload audio to their local server. Editors could decide to distribute selected recordings to other servers enabling news sharing within and across local communities.

References

[Arons] Arons, B. Speech Skimmer: Interactively Skimming Recorded Speech. Ph.D. thesis, MIT Media Lab, 1994.

[Chen] Chen, F., Withgott, M. "The Use of Emphasis to Automatically Summarize a Spoken Discourse". Proc. Int. Conf. Acoustics, Speech and Signal Processing. Vol. 1, pp. 229-232, 1992.

[Driscoll] Jack Driscoll, private communication, 1995.

[Gish] Gish, H., Siu, M., Rohlicek, R. "Segregation of Speakers for Speech Recognition and Speaker Identification". Proc. Int. Conf. Acoustics, Speech and Signal Processing. Vol. 2, pp. 873-876, 1991.

[Hasapes] Hasapes, G. (Executive editor). Emergency Medical Abstracts. Center for Medical Education, Harleysville, PA, 1995.

[Hawley] Hawley, M. Structure out of Sound. Ph.D. thesis, MIT Media Lab, 1993.

[Hindus] Hindus, D., Schmandt, C., Horner, C. "Capturing, Structuring, and Representing Ubiquitous Audio." ACM Trans. on Information Systems, Vol. 11, No. 4, pp. 376-400, 1993.

[Horner] NewsTime: A Graphical User Interface to Audio News. Masters thesis, MIT Media Lab, 1991.

[Makhoul] Makhoul, J. Personal communication, 1995.

[Mullins] Mullins, A. Audio Streamer (Media Lab Masters thesis, in progress)

[O'Shaughnessy] O'Shaughnessy, D. "Recognition of Hesitations in Spontaneous Speech." Proc. Int. Conf. Acoustics, Speech and Signal Processing, pp. 1521-1524, 1992.

[Ousterhout] Ousterhout, J. Tcl and the Tk Toolkit. Addison-Wesley, Reading, MA, 1994.

[Roucos] Roucos, S. and Wilgus, A. "High Quality Time-Scale Modification for Speech." Proc. of Int. Conf. Acoustics, Speech and Signal Processing, pp. 493-496, 1985.

[Rumelhart] Rumelhart D., Hinton, G., and Williams, R., "Learning representations by back-propagating errors," Nature, Vol. 323, pp. 533-536, 1986.

[Schmandt] Schmandt, C. Voice Communication with Computers. New York: Van Nostrand Reinhold, 1994.

[Stifelman] Stifelman, L., B. Arons, C. Schmandt. "VoiceNotes: A Speech Interface for a hand-held voice notetaker". In Proc. of INTERCHI Conference, ACM SIGCHI , 1993.

[TechNation] *Information available by sending email to technation@usfca.edu.*

[Whittaker] Whittaker, S., Hyland, P., Wiley, M. "Filochat: Handwritten Notes Provide Access to Recorded Conversations". ACM CHI Conf. Proc., pp. 271-279, 1994.

[Wilcox] Wilcox, L., Kimber, D., Chen, F. “Audio Indexing using Speaker Identification”. Xerox PARC ISTL Technical Report No. ISTL-QCA-1994-05-04, 1994.

Acknowledgments

Thanks to:

Chris Schmandt, my advisor, for introducing me to audio interfaces and giving me the support to explore the ideas which I've presented in this thesis. He taught me the value of working with real people in real environments when designing interfaces.

Jeff Herman for countless brainstorming sessions, for teaching me everything I know about usability testing, and especially for being a great friend and officemate.

The Aardvarks (Josh Bers, Sara Elo, Jeff Herman, Earl Rennison, and Nicolas Saint-Arnaud) for making our Wednesday evening meetings a great success. They listened to my ideas, tried out my prototypes, and helped bring a balance to otherwise hectic weeks with great food, drink, and conversation.

My thesis readers: Walter Bender for useful comments about the motivation and design of the system. Bob Donnelly for providing many insights into the world of broadcast media. John Makhoul for discussions about speech processing methods and applications.

Jack Driscoll for reminding me of the big picture when I was lost in the trenches of coding.

Sumit Basu for discussions about the speaker indexing algorithm, and for writing all of the accuracy testing software.

Vadim "Tetris" Gerasimov for debugging my (undocumented) assembly code.

Sami Shalabi for helping me battle the nightmares of PC interrupts and memory conflicts.

Felice Napolatino for putting up with my endless requests for hardware.

Maribeth Back, Mike Casey, Barbara Ceconi, Jim Clemens, Dan Gruhl, Tomoko Koda, Charla Lambert, Warren Sack, Jordan Slott, Lisa Stifelman, John Watlington, and Yin Yin Wong for useful comments and help.

Sandra Bunch for early brain storming sessions, for naming the system, and for constant support and enthusiasm for my work.

My Mother and Father for their constant support, enthusiasm, and confidence in me throughout the years.