MAS.632 Speech Interfaces and Mobile Devices
Fall 2008

4

Applications and Editing of Stored Voice

The previous chapter described some of the basic techniques for digitizing and encoding speech for either transmission or storage but said little of its role as a computer data type. This chapter focuses on computer applications of stored speech with particular emphasis on the underlying system support they require, especially sound storage facilities and editing. First, a taxonomy of voice applications is described based on the usage characteristics of situations in which speech is digitized and played back. Next, one of these classes—speech in documents—is explored in some detail. Finally, this chapter presents underlying representations and a number of examples of speech editors, which are required to support sophisticated use of audio in documents. In the course of this discussion, many examples of research projects utilizing digitized speech are mentioned to illustrate the range of approaches that have been taken to applications of stored voice.

This chapter focuses on application areas, their associated system requirements, and the functionality they require for supporting voice digitization and playback. Discussion of many user interaction requirements and techniques is postponed until Chapter 6, which deals with user interface issues for interactive speech response systems that employ both digitized and synthesized speech. For the purposes of this chapter, it is assumed that each application runs in isolation; Chapter 12 discusses issues of voice as a data type, including visual representations and sharing of voice data among applications on a computer workstation.

TAXONOMY OF VOICE OUTPUT APPLICATIONS

Stored voice is essential to a variety of applications such as these.

- Toys that play a small number of messages from read only memory.
- Dictation, in which voice is recorded for later transcription into text by another person.
- Voice mail, in which a message is recorded informally by a caller and is accessed once by the recipient over a telephone-based interface.
- Voice annotation of text or multimedia documents, which involve more careful authoring and often a visual user interface.

This section characterizes such applications according to the functionality that stored voice requires as differentiated by how users employ it in these classes of applications. Applications may emphasize either recording or playback, and the stored voice may be heard once or repeatedly. How the stored voice is to be used in a particular application determines the underlying system support required for both recording and playback as well as influences the choice of a particular speech coding technology and how it is stored as computer files.

Playback-Only Applications

Stored voice can be employed in a variety of applications for which a small repertoire of utterances needs to be spoken. Depending on the application, varying degrees of speech quality may be acceptable. Applications at the low-end of voice output technology employ single integrated circuits for both the storage and decoding of the digitized speech. Examples of this simple technology are inexpensive consumer products such as talking toys (dolls, stuffed animals, etc.) and video games. Some answering machines use stored speech to record the time onto the answering machine tape at the end of each incoming message. Answering machines that store the caller's message digitally are also appearing as consumer products. Voice warnings are used in both commercial and military aircraft cockpits. Automobile manufacturers have used digitized speech for warnings, and some cash registers recite each item and its price; however, neither of these two applications has proven to be very popular.

In situations where only minimal quality speech output is needed, it may be economical to use an asymmetric combination of speech coder and decoder. It is practical to use a computationally intensive coding algorithm, such as LPC, if the decoder is simple and inexpensive. Only the decoder needs to be included in a product which supports playback only.

Higher quality speech is needed for recorded public announcements such as the recitation of station names on automated trains or periodic, repetitious announcements over airport public address systems (e.g., warnings about taking baggage carts down the escalator). Because it is important for the listener to understand the message, these applications demand higher quality speech. In

Japan such announcements are ubiquitous in public locations and are usually spoken in a stylized female voice.

A similar class of applications are the so-called **audio-tex,** or **interactive voice response (IVR)** telephone-based information services. To use such services, one dials a telephone number and selects topics with touch tones. Recorded speech is used both for the information to be played back after selection as well as for prompts and menus. Examples include services to hear a weather forecast for a particular city, to find out one's current bank account balance, and to access an airline's flight schedule. The stored voice for these applications needs to be of adequate quality to be easily understood over the telephone by infrequent users. Because the service is accessed over the telephone, the sound data is most conveniently stored digitally on magnetic disk at a single site. Interaction techniques for interactive voice response systems will be discussed in detail in Chapter 6.

For all the applications just mentioned, the stored speech is used as "read only" data as the purpose of the application is to play a fixed set of prerecorded sounds. There is no user interaction with the voice itself; the system simply plays it at the appropriate time in response to the user's input. There is no need for a speech editing capability or a visual representation of the recording on a display, and the speech may be stored in whatever format is most convenient for the output device.

The developers of such applications occasionally need to update the voice recordings, and may use an editor or speech development system to do so. This development system does not need to be distributed with the application since updating can be done at a single, central location on an infrequent basis and distributed in ROM (Read Only Memory), floppy disks, or digital tape. Even for the developers, however, "editing" often consists of simply rerecording the passage repeatedly until the desired diction is achieved.

Some applications make use of a limited repertoire of "canned" speech segments to compose sentences; this involves concatenating short segments of digitized sound. One example is the mechanized playback of telephone numbers from directory assistance. It is essential that the individual sounds be accessible quickly enough during playback to avoid a pause between segments. If the speech is stored in semiconductor memory, this is not an issue, but for slower media, access time may be problematic.

In most of these applications, voice is chosen because it is the best output medium available. Because voice broadcasts through the air, it can be used in situations where displays would not be appropriate or visible. It is chosen as a warning channel because it is effective when the user is doing something else with his eyes, such as watching where he is driving or flying. In the case of the telephone-based services, access is also a dominant issue; users of such services take advantage of the nearly universal availability of the telephone. These advantages will be discussed in more detail in Chapter 6.

Interactive Record and Playback Applications

Voice mail is currently the most pervasive example of the class of application that involves frequent recording and playback of speech segments. Each time a call is

received, a new message is recorded. Usually the recipient listens to the message only once but may replay parts of the message while transcribing information such as the caller's telephone number. Although voice mail systems are most often treated simply as elaborate answering machines (with which messages are recorded once and played once), they are functionally more powerful than the consumer products found in homes. For example, voice mail systems can be used within an organization to disperse voice memos to distribution lists specifying multiple recipients or to specify a future delivery date for a message. This verbal memo is recorded just like a telephone message, although the author may speak more carefully or rerecord the message several times because it will be heard by a wider audience.

Voice is slow and serial, and telephone-based voice mail interfaces do not offer the user the opportunity to save messages in a "folder" with a mnemonic name as do most electronic mail systems. Archiving old voice messages and recalling them later is tedious, and few users take advantage of such capability; consequently, recording and playback occur equally often, which eliminates the advantages of an asymmetrical encoder/decoder combination. Random access of the recorded speech file is necessary as well to support repeated listening to a portion of the message.

Because of the more casual nature of telephone-based voice messages over paper memoranda, editing capabilities need not be provided; it would be difficult to build an effective editor using an audio-only interface based on touch tones. Instead, the user may be offered the option of reviewing a message before sending it and perhaps may choose to rerecord it. This is in contrast to the system recordings provided by the voice mail vendor, which are recorded very carefully often, with the assistance of voice editing software.

During recording, **Automatic Gain Control (AGC)** may be employed in an attempt to record constant amplitude messages from variable quality telephone connections. AGC uses variable amplification during recording to normalize the level of the recorded signal independent of the source level but it has limitations. Boosting the gain for a quiet speaker in a noisy environment may not enhance intelligibility as both the signal and noise are amplified; a better strategy is to monitor recording levels and prompt the quiet talker to speak louder. Another problem with AGC is that it tends to increase the gain during quiet portions of the message, or pauses, which causes a "whoosh" as the background noise increases; this is evident on many home videotapes. AGC can be improved by normalizing the gain based on peaks in the speech signal averaging over a time period, but this technique is not in widespread use. Although each telephone call can have a different audio level, the level is usually consistent for the duration of a message. This allows gain to be applied after the recording by multiplying each sample by the same scale factor; this avoids the whoosh produced when the gain is varied during a recording.

Dictation

Dictation is an application of stored voice in which recording capability is required by the user, but the stored voice is never meant to be heard by the ulti-

mate recipient. Instead, a transcriptionist listens to the recorded voice and transcribes it into text, often as a letter, memo, or short report.

Dictation has a different set of interaction requirements than other voice applications. The originator of a dictation spends a significant amount of time thinking about content and so requires a start/stop control or push-to-talk button; this is somewhat in contrast to telephone messages, where the caller must speak all at once at the "beep." The transcriptionist uses a foot pedal to control playback. Transcribing requires the capability to repeatedly review a portion of the message to understand individual words, but minimal random access to jump or scan through large portions of the recorded voice. The ultimate recipient of the message reads the transcription, having no interaction with the stored voice.

Although dictation and transcription have traditionally been done using analog tape recorders, other methods are now available. Dictation in an office environment can be digitized with the transcriptionist using a computer to control playback (again using a foot pedal). Dictation can also be taken over the telephone. Although some companies market this as a special product, business travelers already frequently leave lengthy voice mail messages for their secretaries to transcribe.

Dictation is used to create text documents when users refuse to use keyboards or can afford the cost of transcription as offset by the faster speed of speaking over typing or writing. Digitized dictation can also be used as a temporary information storage medium while awaiting transcription. For example, a hospital's medical records system could apply such a hybrid system. Physicians usually dictate their reports, but hospital administrators as well as medical staff prefer medical records to be stored in computers for both easy access and archival storage. Once dictation tapes are transcribed the text is easily stored on line, but until then the information is inaccessible. If the dictation were stored as digital sound files, it could be accessed immediately from a voice-and-text workstation, even as transcriptionists slowly converted the sound files to text. But for the hospital staff accessing voice in the medical record before transcription, a user interface different from the transcriptionists' foot pedal is required; voice as a document type requires a greater degree of interactivity.

Voice as a Document Type

The use of voice for announcements or in toys does not require any recording capability after the announcement has been created, while both voice messages and dictation are intended to be heard only a few times before being discarded. In contrast, voice can also be utilized as a medium for the creation of documents which are intended to be more permanent or to reach a larger audience. As a document type, voice may be used as a means of annotating text or it may itself be the primary medium. Voice documents may also include other media such as text and images. More care will be taken during recording voice documents as the stored speech is intended for more formal use than a telephone message. In many cases multiple listeners will hear the recording, especially if it is part of a presentation.

Computer workstation-based multimedia applications such as these have several additional requirements for stored voice not encountered in the classes of applications discussed thus far. The first requirement is a visual representation of the stored speech appearing in a document and a graphical user interface to control playback. Because of the longevity and larger audience for voice documents, more care is taken in their creation so the second requirement is a voice editor that provides a means to manipulate stored voice, including cutting, pasting, and recording new material for insertion. Editing has implications for sound file storage since it is likely to produce many small snippets of sound that result in fragmented files, and certain methods of file storage are more compatible with the internal data structures used by the editor to manipulate sound data.

VOICE IN INTERACTIVE DOCUMENTS

The remainder of this chapter will focus on uses of stored speech as a document type. As mentioned above, voice may be used alone or composed with other media to form a document. This chapter focuses on composed multimedia documents employing a display and a graphical user interface. Chapter 6 treats the problems unique to audio-only documents accessed without any display. A voice editor is also usually provided to aid authoring of voice documents. Although voice may be imported into one application from another much as text is cut and pasted between windows, discussion of this capability is postponed to Chapter 12; here we consider only document creation and retrieval.

When both text and voice are available, the different media may be used for different types of messages. Voice is appropriate for casual and transient messages, while text is preferred for messages that the recipient will likely save as reference. Voice is also desirable for its expressiveness; voice can carry subtle messages far more easily than text. It is also faster to enter messages by voice. Nicholson reported the results of a usage study of a voice and text message system in [Nicholson 1985] that supports all these observations. Chalfonte *et al.* [Chalfonte *et al.* 1991] compared voice and text as a means of commenting on text for a collaborative writing task; they found that a greater proportion of voice annotations carried "deeper" comments on style and structure, while text markings related to surface features such as choice of words and syntax.

Voice may be used to annotate text documents; it overcomes some limitations of the two-dimensional page by allowing comments to be placed anywhere with a content-associated visual icon that requires less space than handwriting. Conversely, text may be used to comment on a voice recording such as appending text remarks to a voice mail message before forwarding it to another party. Applications to support voice and text require the creation and storage of voice and a user interface to access it while viewing a document.

The Diamond project at BBN (Bolt, Beranek, and Newman) [Thomas *et al.* 1985] was an early example of incorporating voice with text. Diamond was designed to support voice and graphics in both electronic mail and more formal

documents. Nearly 10 years old, Diamond used LPC speech coding to minimize audio storage requirements. But the LPC coder was an external computer peripheral that was not readily available. This, combined with the low quality of LPC, limited the use of Diamond and it remained basically a prototype system. But many aspects of Diamond have reappeared in *Slate,* a multimedia mail system currently available as a product from BBN [Crowley 1991, Lison and Crowley 1989].

Slate documents can include a variety of media and Slate is designed to work in concert with a computer conferencing system, MMConf. Slate uses a commercially available speech coding card or the audio capabilities of the workstation itself. In addition to speech, Slate incorporates still images (bitmaps), geometric graphics (lines, curves, etc.), motion video, spreadsheets, and charts. Slate indicates the presence of voice in a document with a small icon that can include a line of text as shown in Figure 4.1. By default, the text contains the author's name and recording information, but the author can change the label. The reader clicks on the icon to play the associated stored voice. Slate supports a small voice editor to allow the author to modify a recording during its creation as shown in Figure 4.2.

Another example of voice annotation of text documents was Quilt at Bellcore (Bell Communications Research) [Leland, Fish, and Kraut 1988, Fish, *et al.* 1988]. Quilt was a tool to allow collaborative document creation, editing, and review. Multiple parties could add notes to the document being edited using a

Figure 4.1. Slate shows an annotated icon to indicate speech in a document.

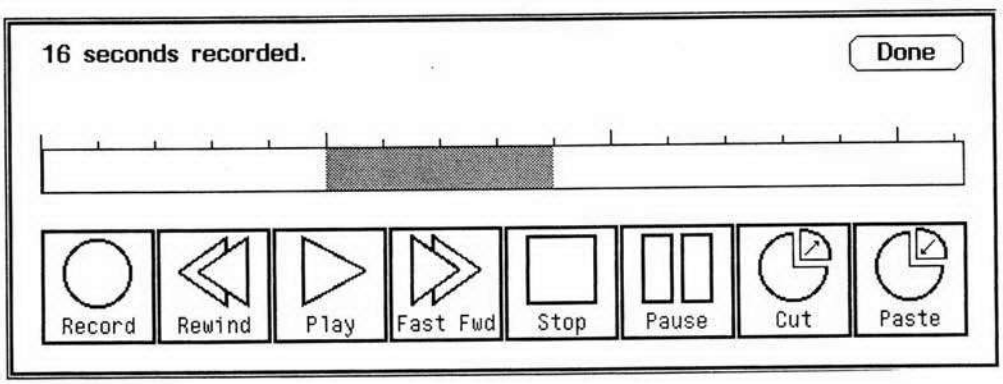

Figure 4.2. The Slate voice editor.

Courtesy of Telcordia. Used with permission.

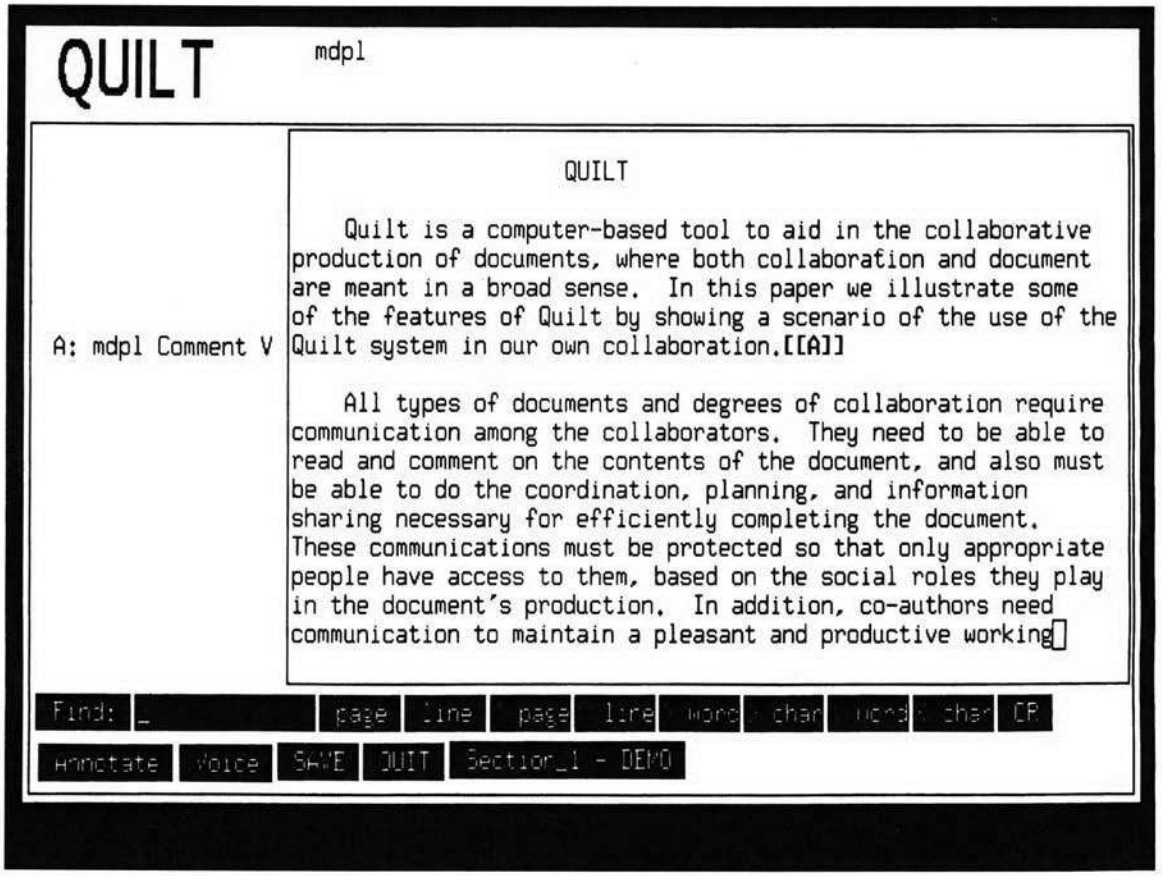

Figure 4.3. Quilt used **V** symbols to indicate the presence of voice comments in a text document.

simple voice annotation scheme. Voice recordings were indicated in the text with a "V" symbol as shown in Figure 4.3. Quilt was never widely deployed so it did not receive significant use, but it motivated preliminary studies with tape recorders that indicated that voice annotations were lengthier and more penetrating than text.

The Xerox PARC (Palo Alto Research Center) Etherphone project[1] supports voice annotation of text. It uses a "comic strip balloon" as an indicator of voice stored as part of a text document as shown in Figure 4.4. The capability to mix

[1]Etherphone is discussed in more detail in Chapter 11, and its voice editor is discussed later in this chapter.

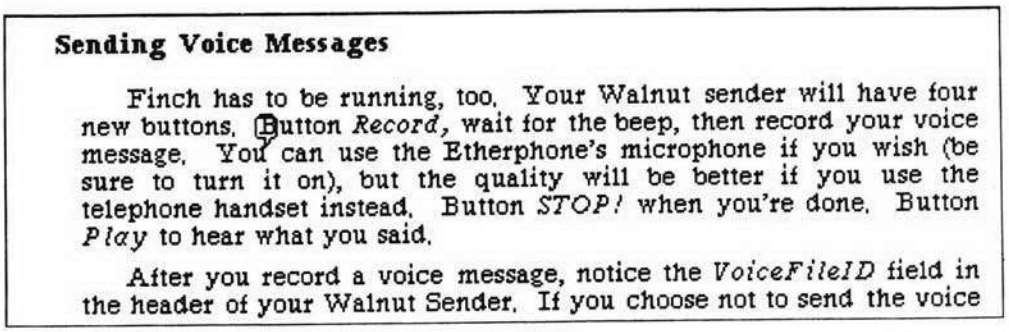

Figure 4.4. Voice is indicated by a "balloon" at Xerox PARC. (Copyright ©1986, Xerox Corporation. Reproduced with permission of Xerox Corporation.)

voice with text was extended by Zellweger to **scripted documents,** which sequentially play back (or display) portions of a multimedia document [Zellweger 1989]. A scripted document is a form of presentation with the management of time and sequence an added dimension of authoring; the script provides a single path through the document.

Stored speech is also used in multimedia documents that provide many paths for presentation under control of the "reader." For example, the MUSE system from M.I.T.'s Project Athena [Hodges *et al.* 1989] uses recorded video, audio, and textual annotation for the interactive presentation of educational material.[2] The student in a foreign language course may be asked to make decisions on the basis of a spoken dialogue played from a videodisc. Text may be employed during playback to define new vocabulary items. The student's response is indicated using the mouse, and the system responds with the next appropriate sequence.

An entirely different approach is taken in documents which consist only of voice, with nonvisual user interfaces. Muller defines an architecture for voice documents as a network of short audio segments and points out some of the timing issues with synchronizing user inputs with whichever segment is currently playing [Muller and Daniel 1990]. Resnick built a telephone-based groupware application, Hypervoice, that allowed callers to browse a community bulletin board of voice entries [Resnick 1992a]. Each entry consisted of a number of short voice recordings such as an event's time, description, and a contact person's telephone number; users navigated with touch tones. Arons' Hyperspeech used speech recognition to navigate a hypermedia database consisting of audio inter-

[2]MUSE is both the presentation system as well as the underlying language and editor used for creating the presentations.

views with five user interface researchers [Arons 1991b]. VoiceNotes is a user interface to a handheld digital audio storage device; users employ speech recognition and mechanical buttons to traverse and edit lists of reminders [Stifelman *et al.* 1993].

VOICE EDITING

While adding voice to a document or presentation users will occasionally make mistakes and must correct them. To change a very short voice recording it is easy to simply record the speech again; an example is the outgoing message on an answering machine. Other applications, which use longer recordings, can benefit from the addition of a voice editor. An editor allows a user to delete or move portions of a recording and to insert new audio into an existing recording; these operations are controlled with a graphical user interface. An editor can also be used to extract a portion of sound from a longer voice recording to be used in another application. The rest of this chapter considers several architecture issues for voice editors and illustrates editor user interfaces with a number of examples.

Temporal Granularity

Audio editors may provide different levels of temporal granularity for specifying the position within a sound at which a sound manipulation operation is to occur. The granularity of an editor is determined both by the application's requirements as well as by the nature of the stored sound to be edited. The temporal granularity also places requirements on the editor for display of the sound being edited.

A **fine-grained** editor allows edit operations to be performed at any point in the sound. It makes no assumptions about the content of the sound and gives the user complete control of editing. Fine-grained editors usually display sound as a waveform (see Figure 4.5) or **average magnitude** (see Figure 4.6). Average magnitude is the sum of the absolute value of some number of audio samples divided by the number of samples; samples are taken over a large enough window to average multiple pitch periods. A mechanism may be provided to zoom in and out of the sound view for local or global editing; the zooming changes the ratio of pixels to time for display purposes.

A fine-grained editor is most useful for studio work such as producing music or motion picture sound tracks. In these situations a great deal of flexibility is required as a variety of sounds may be manipulated and very careful edits are needed. For example, the Sound Droid digital audio editing system [Snell 1982] was used in the motion picture industry and emulated existing analog studio equipment both in user interface as well as basic functionality.

A **coarse-grained** editor provides much less temporal control but abstracts sound into "chunks" based on periods of speech and silence. Coarse granularity is more suitable for editing speech because speech consists of a series of semanti-

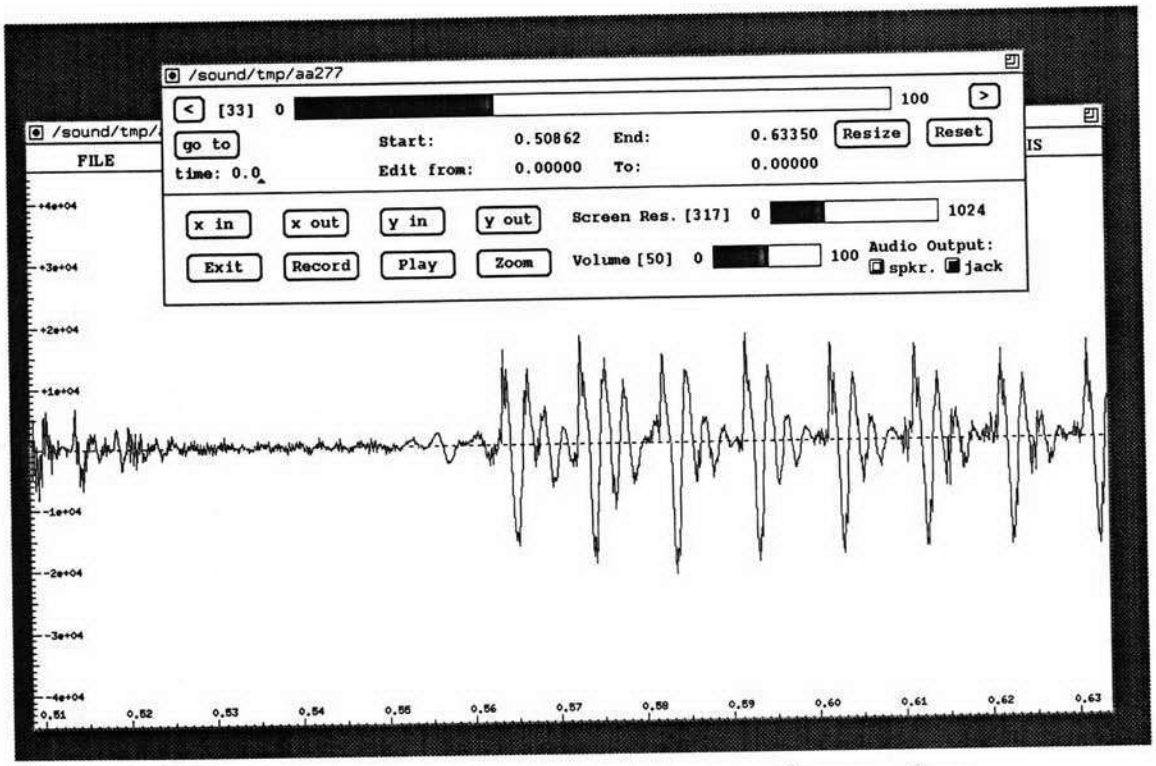

Figure 4.5. The MixView audio editor displays sound as a waveform.

cally meaningful phrases; a coarse-grained editor facilitates the user's identifying phrases and editing them as units. The abstraction of sound into speech and silence intervals also allows more succinct visual representations with reduced screen space requirements.

The coarse-grained editor is well suited to removing spurious remarks, cleaning up false starts from a recording, and replacing a sentence; it is not suited to modifying individual words. In a study of users of an early voice editor, Allen noted that subjects desired to edit individual words frequently [Allen 1983], but in practice it is nearly impossible to modify a single word in a sentence without having the result sound broken, with very unnatural prosody. Since users will not be able to perform successful fine-grained editing of voice, coarse-grained editors make no attempt to support this function, and use a simple binary representation to simplify the user interface.

Manipulation of Audio Data

An audio editor must manipulate stored sound data in response to the user's commands. As users are likely to make changes and review them several times before achieving the desired result, it is useful if the editor allows preview before com-

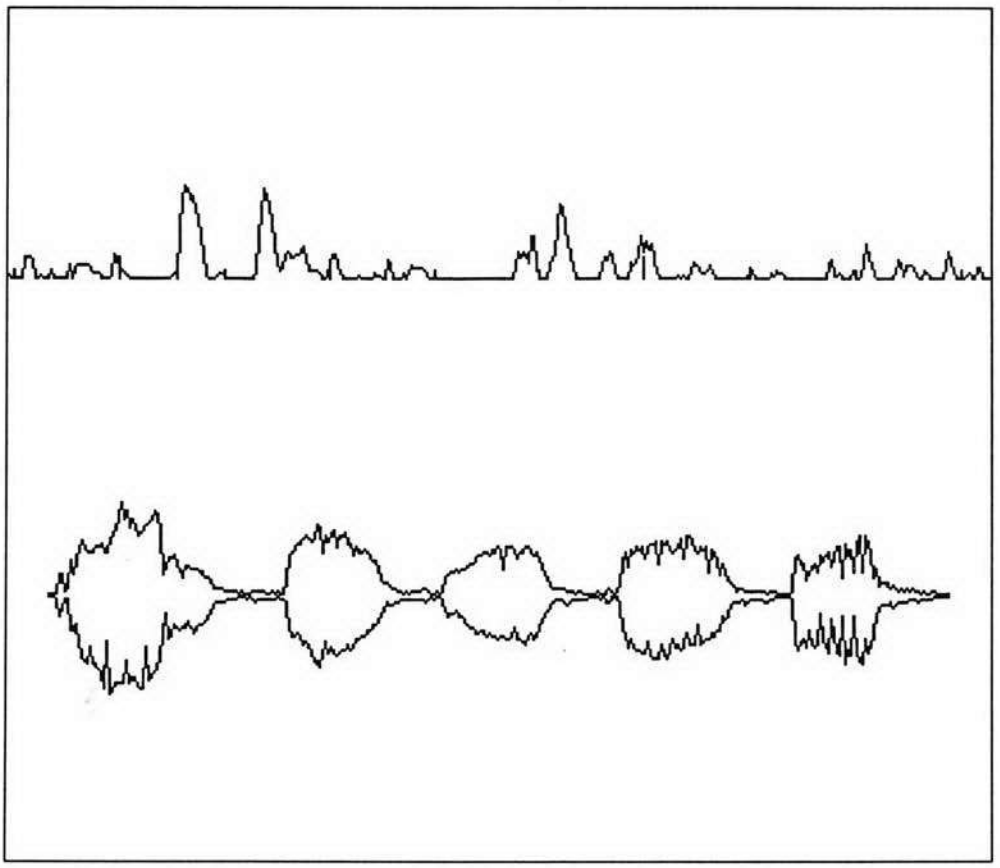

Figure 4.6. Average magnitude display of a sound (top) and an envelope display showing both positive and negative peaks (bottom).

mitting the changes to the original sound file. Although this could be done simply by editing a copy of the sound file, most editors instead operate on an internal representation of the sound that allows users to edit and review without changing the file at all. When finished the user may choose to save the edited sound at which point the file is changed. Many text editors employ similar techniques.

More importantly, editing an in-memory representation of the file offers significant performance improvements over editing a file directly. For example, if the user deletes the first character of a text file or the first 100 milliseconds of an audio file, it is time consuming to move the entire remainder of the file.[3]

[3]Under some operating systems this requires copying the entire file as well.

Instead, the editor just modifies its description of the sound to note that the very beginning of the file should be skipped over during playback. At the end of the editing session the disk file can be changed in a single operation avoiding slow file operations during editing. This is even more important in the editing of sound than in text as sound files are much longer than most text files. A chapter of this book may takes roughly 40,000 bytes of text storage, but for many speech coding algorithms this would represent less than 10 seconds of sound.

An audio editor may represent a sound being edited as a **buffer** or **playback list.** This can be implemented as a linked list where each entry in the list points to a sound file and indicates start and end points. Figure 4.7 shows how such a representation is employed during editing. When a file is initially read into a buffer (shown in the top portion of the figure), the buffer includes the entire sound. The edit list contains a single entry with a start point of zero and stop point at the end of the file; duration is represented by shading in the figure. A "cut" operation removes a portion of the sound (shown in the middle of the figure). Now the buffer list has two entries. The first entry points to the initial part of the file at offset zero and extending up to the beginning of the cut section. The second buffer entry points to the same file but starts after the cut section and extends to the end of the sound. When the edit is reviewed, the editor plays the first chunk of the file and immediately skips the middle part of the file to play only the final segment. When the buffer is then written out to a file, a single homogeneous file is created by merging the two sections of sound (shown in the bottom of the figure).

To insert a new sound into the middle of a buffer, the buffer must be split into two list entries with the new sound added to the list in between them as shown

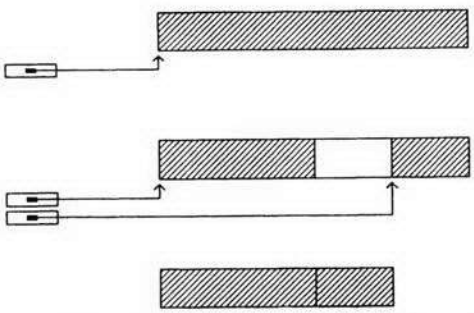

Figure 4.7. Edit buffers. At the top, a single buffer node points to an entire file. In the middle, after removing a portion of the sound, two buffer nodes are required to point to the two segments of the file. At the bottom is the correct representation after the buffer has been written to a file; the two portions of the original file have been merged into a single continuous file.

in Figure 4.8. With each edit operation performed by the user, one or more entries are likely to be added to the buffer list. After a number of operations, the resulting representation is complex (see Figure 4.9), but it is hidden completely from the user. Depending on how sound is stored in disk files, the complexity may be largely hidden from the editing program as well; some sound file systems

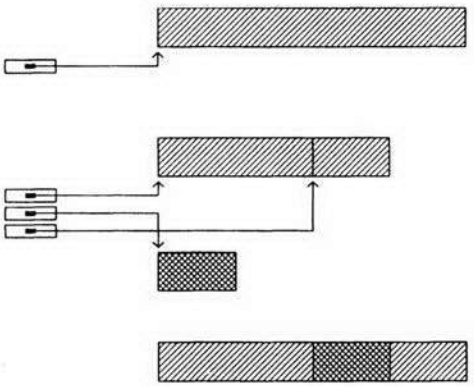

Figure 4.8. Edit buffers showing insertion of a new sound into an existing sound. The original sound at the top must be divided into two segments, each with its own buffer node and a new node inserted in their midst. The resulting buffer in the middle may then be written to a single file at bottom.

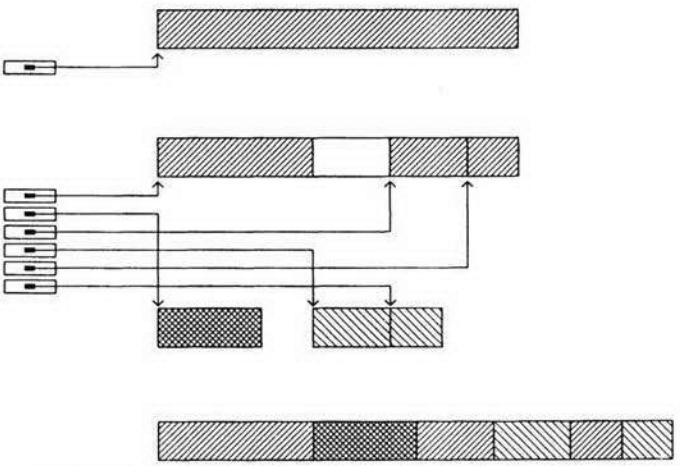

Figure 4.9. After a number of edit operations, the resulting buffer may become rather complicated.

provide transparent sequential access to chunks of sound stored in different locations on the disk.

EXAMPLES OF VOICE EDITORS

All sound editors include a user interface to the editing functions. These functions include selection of a sound file for editing, playback of the sound, deleting part of the sound, and inserting new sound. The appearance of the user interface depends on the window system, graphical toolkits, and style conventions for the particular operating environment in which it runs. The editor may be modeled after existing tools with which users are familiar, such as text editors for the office and audio equipment for the studio.

One of the earliest voice editors was built by Maxemchuk at Bell Laboratories as part of an experimental speech storage facility [Maxemchuk 1980]. This editor had a primitive interface based on an alphanumeric terminal; a subsequent version used a more sophisticated display. This section briefly describes a number of more recent audio editors; they exhibit a wide range of user interfaces and sound representations yet share many aspects of underlying software architecture.

Intelligent Ear, M.I.T.

The Intelligent Ear was an early voice editor from the Architecture Machine Group (M.I.T.) [Schmandt 1981]. It displayed a sound as an amplitude envelope using average magnitude to modulate both height and brightness as shown in Figure 4.10. The entire sound was displayed on the screen; long sounds were shown with less temporal detail than short sounds because of the compression required to fit them on the screen. The Intelligent Ear also used speech recognition to attempt to detect keywords in the sound and displayed these as text with brightness indicating the degree of certainty with which the word was recognized.

This editor used a touchscreen for input. When the user touched a location on the sound display, a cursor moved to that location. When the *play* button was touched, playback began at that point and the sound display changed color while the cursor moved in synchronization with playback. This motion was provided to aid the user's mental mapping of the visual cue of average magnitude to the actual contents of the sound. Other editing functions included *delete* and *record,* which enabled the user to specify an insertion point at which to record additional speech. Edit operations were buffered in that they did not change any sound file data, but this buffering was only one operation deep. After an insertion or deletion the user could play through and listen to the result; only after accepting this result was the sound file actually changed on disk. An edit operation was either accepted or rejected by touching a red or green button which appeared after the edit, but the user was not able to make multiple edits and preview the result without modifying the file.

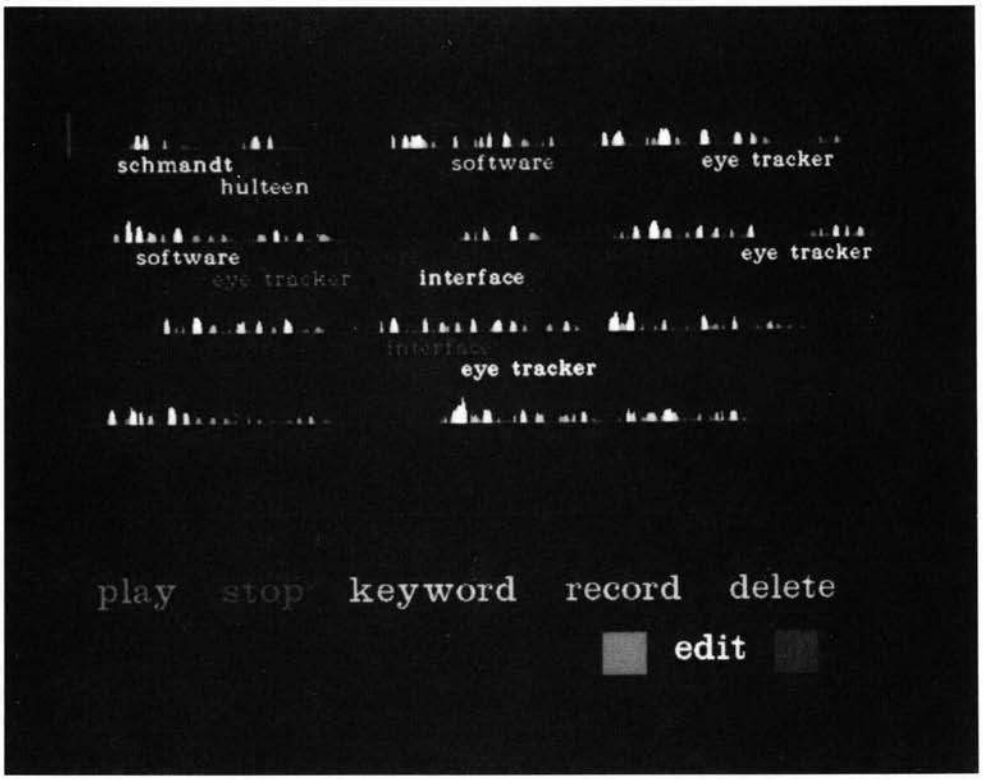

Figure 4.10. The touch-sensitive user interface to the Intelligent Ear editor.

The Intelligent Ear was a demonstration project which saw only limited internal use by its authors to edit voice prompts in later research projects. It was novel in its early use of a color graphical interface and touchscreen for voice editing.

Tioga Voice, Xerox PARC

The Tioga Voice audio editor at Xerox PARC was modeled after the Tioga text editor's user interface and underlying data representations with the intention of making editing voice similar to editing text [Ades and Swinehart 1986]. Tioga Voice was designed to support audio in a variety of applications and could be used as a listening tool as well as an editor.

The Tioga Voice user interface shown in Figure 4.11 supported a number of features. Portions of speech and silence were displayed by a binary representation in a small bar (or "capillary tube" as described by its creators), and text annotation could be added as well. On a color screen, recent changes or additions appeared brighter than older sound segments. During recording the display changed dynamically showing the new sound as a string of arrows both to keep a sense of

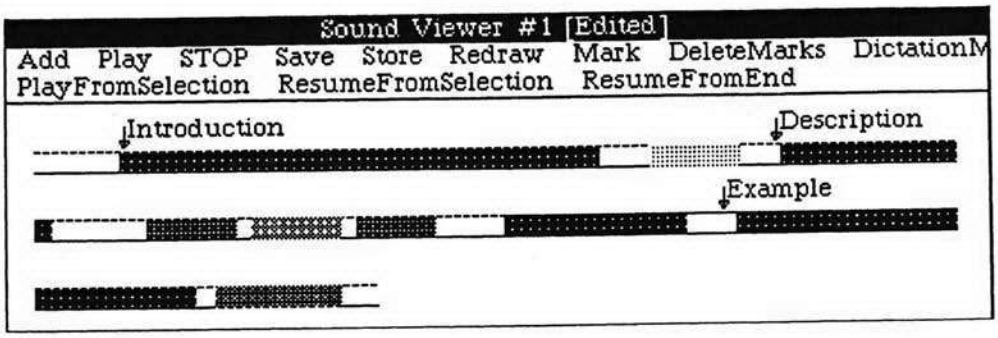

Figure 4.11. The Tioga Voice editor. (Copyright ©1986, Xerox Corporation. Reproduced with permission of Xerox Corporation.)

Courtesy of PARC. Used with permission.

its relationship to the whole recording as well as to remind the user that the recording was still in progress. During playback, a marker moved across the sound bars indicating the current play position.

To facilitate the manipulation of sound files, Tioga Voice kept track of the current *selection,* i.e., the piece of sound most recently referenced. During dictation mode, the selection corresponded to the most recent portion of speech recorded to allow for quick review or cancelling erroneous input. If the user began recording, coughed, and then stopped recording, this audio segment could be deleted by a single mouse click.

Tioga Voice was built on top of the Etherphone system, which provided sound storage and voice services through server processes running on other computers. Although voice segments were stored as sequential files on disk, they were accessed through the file server as a list of segments referred to as a *voice rope* [Terry and Swinehart 1988]; text was stored in a similar fashion, also referred to as a *rope*. A complete sound consisted of the rope database plus raw sound storage (see Figure 4.12). The voice ropes were similar to the edit buffer schemes described above except that they were saved on disk as part of the storage mechanism provided by the server for all sound access. Each rope represented a path through portions of one or more sound files; editing simply created more ropes. Because editing simply created more ropes instead of actually modifying the underlying sound files, periodic garbage collection was required to recover unreferenced disk files.

PX Editor, Bell Northern Research

Another example of a sound editor is that which was developed for the PX (Personal eXchange) project at Bell Northern Research's Computer Systems Research Laboratory [Kamel, Emami and Eckert 1990, Bowles *et al.* 1991]. PX is discussed in greater detail in Chapter 11; in this section we consider only its sound editor.

The PX editor, shown in Figure 4.13, ran on Apple Macintosh computers with an external hardware interface for speech input. This editor supported several

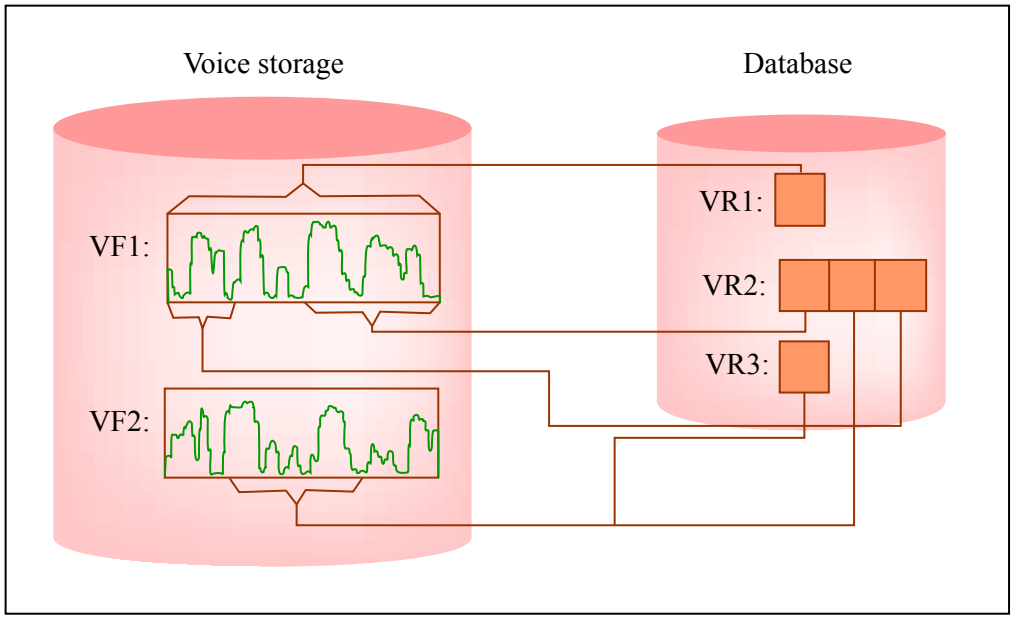

Figure 4.12. Voice ropes, showing the rope database which references portions of linear sound files. Figure by MIT OpenCourseWare, after Terry and Swinehart, 1988

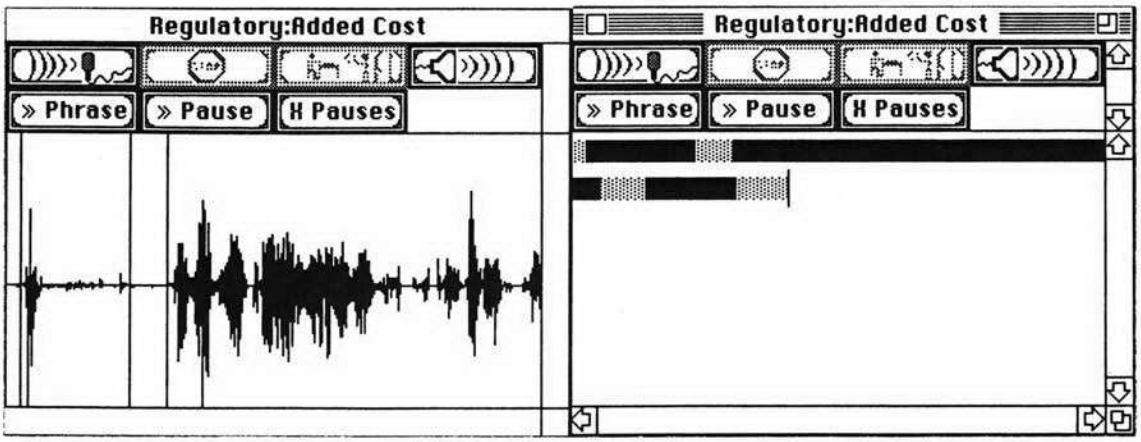

Figure 4.13. The PX sound editor showing two sound representations. The left view is a waveform envelope, while the right view shows speech and silence intervals. (©1990. Reprinted by permission of BNR.)

representations of sound. Figure 4.13 shows the same sound displayed in two windows: one using a waveform envelope and the other showing speech and silence intervals much like the Tioga Voice editor. This simple editor was meant to be a common tool used in a variety of applications such as creating a voice mail greeting and annotating text. In addition to the playback and recording functions described previously in this chapter, the PX editor allowed the user to skip to the next phrase during rapid playback.

PX utilized small segments of sound file storage called "voice clips," which were stored and accessed much like Xerox's voice ropes. Unlike Etherphone, however, the PX editor was implemented entirely on a single computer; there was no server process involved. Voice clips described a series of portions of a file to play in sequence, and every edit operation could produce multiple new voice clips. A sound file held a number of voice clips followed by the reference audio data, which was played as indicated by the clips. In contrast to Xerox's voice ropes, the voice clip database was stored in the same file as the raw audio so it could refer only to data in that file. If a new sound was to be made by merging two other sounds, these would have to be copied into the data area of the new sound file.

Sedit, Olivetti Research Center, and M.I.T. Media Laboratory

A similar audio editor, sedit, was originally written at Olivetti Research Center as part of the VOX audio server project [Arons *et al.* 1989].[4] It was subsequently extended substantially at the Media Laboratory where it is in use today. Sedit uses the X Window System for its graphical user interface and employs a sound editor widget.[5] The sound editor widget (see Figure 4.14) uses black and white bars to display periods of speech and silence; this display is similar to that used by Tioga Voice. A scroll bar beneath indicates how much of the entire sound file is visible and allows the user to choose a portion of the sound to view in detail. As the sound plays, a cursor (a vertical bar) moves across the display synchronously with playback; if necessary, the scroll bar also moves to keep the current location in view.

Sedit can edit multiple sounds simultaneously allowing data to be cut and pasted between sound files; Figure 4.14 shows several sounds active within the editor. The horizontal lines in the figure indicate the portion of a buffer selected for the next operation. Selection is performed by dragging the cursor across the sound with the mouse. Once a region is selected, the edit operation is indicated by clicking on the appropriate button.

[4]VOX is described in detail in Chapter 12; discussion here is limited to support for editing.

[5]In X Windows a "widget" is a basic user interaction object such as a button or scroll bar that can be instantiated multiple times in multiple applications.

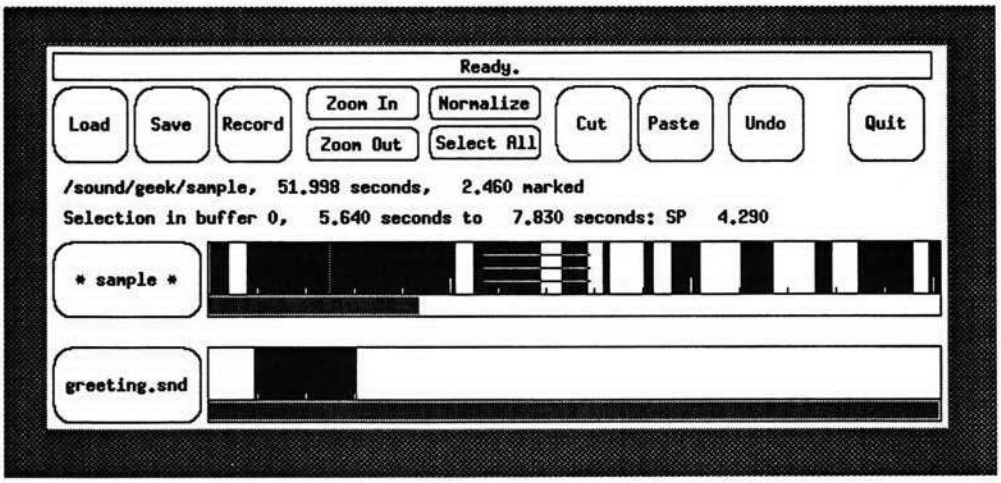

Figure 4.14. The *sedit* audio editor.

A year after the demise of the VOX project, sedit was revived at the MIT Media Lab. Capabilities that formerly had been implemented in the server were included in the editor and new functionality was added. One new function normalizes audio level among all buffer segments; this is useful when editing together source material recorded with different microphone gain levels. Other functions allow changing the time scale of the sound display widget and interaction with a selection mechanism allowing sound to be moved between applications (see Chapter 12). This is the version of sedit shown in Figure 4.14.

Pitchtool, M.I.T. Media Laboratory

Pitchtool was a special-purpose editor written at the Media Laboratory to aid in the study of intonation. It is mentioned here as an example of an editor interface designed specifically for a particular application. Pitchtool was used as much to analyze recordings as to change them and so included such capabilities as variable playback speed, textual annotations, and spatial scaling of the sound to show extended portions of dialog or detailed views of individual phrases.

Pitchtool, shown in Figure 4.15, displayed pitch, energy, and segment duration of a digitized speech file. In the figure, the solid line represents pitch, while the broken line represents energy. Pitchtool also provided for modification of these parameters to allow hand-editing of the pitch and energy for experiments in the synthesis of intonation. The mouse could be used to "paint" a new pitch track or two points could be selected with pitch interpolated linearly between them.

Because it was designed to manipulate the pitch of sounds, Pitchtool used Linear Predictive Coding (see Chapter 3) for its speech storage. LPC encoding was

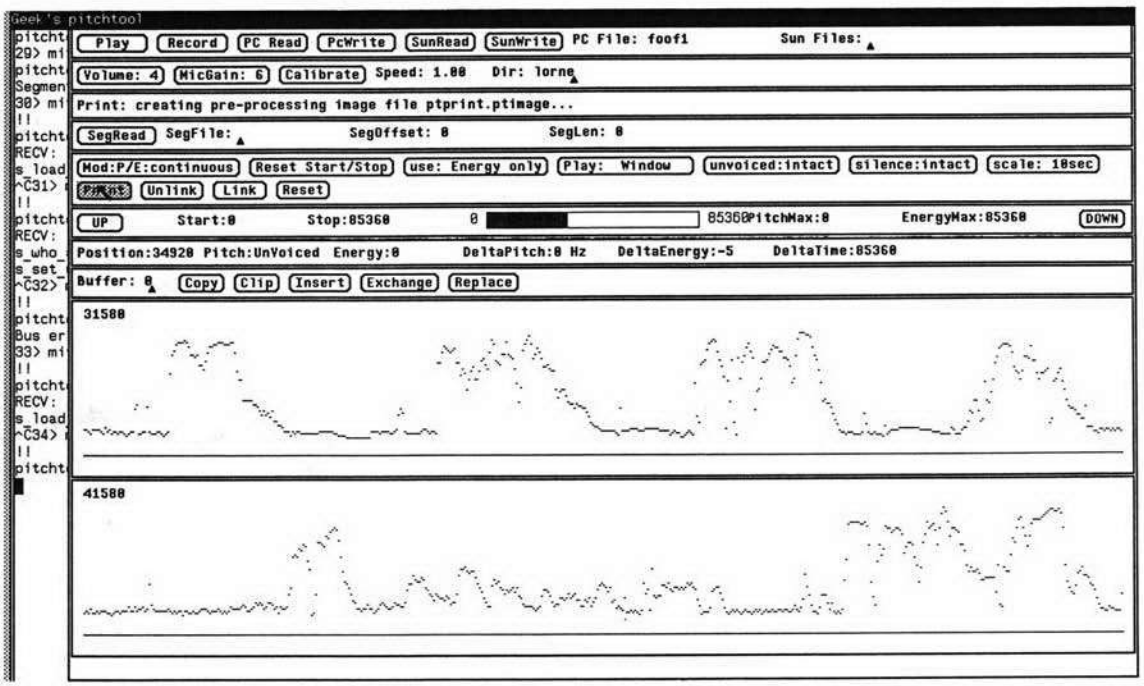

Figure 4.15. Pitchtool allowed editing of pitch and energy.

performed by a server, which included a signal processing card and small computer with local storage, communicating to the host workstation via a serial interface (this server is described in more detail in Chapter 12). As with the original sedit editor, all manipulation of sound data was implemented in the server; Pitchtool merely provided the user interface.

SUMMARY

This chapter described classes of applications for stored speech with emphasis on the underlying system and coder requirements for each. For some of these classes, speech is played only while others require recording as well. Recordings can be casual as in voice mail messages or more formal for presentations or audio memoranda. Recording for dictation may be short lived before it is converted to text. Each class of operation places different requirements on audio storage in terms of the quality required for data compression algorithms, speed to access voice data, and the synchronization of playback with a graphical user interface or other presentation media.

Particular emphasis was placed on interactive audio documents because they highlight requirements for screen-based user interfaces and sound editors. The

range of example systems and editors has revealed a variety of graphical user interfaces and representations of sounds. Editors generally employ in-memory buffers to describe sound file segments during editing, but this capability may be included in the set of primitives provided by the sound file management system. Similar storage issues arise with file formats and editors for all multimedia data that require very large files that may become fragmented as a result of editing operations.