A SURVEY COURSE ON COMPUTER AUDIO*

Stephen V. Rice
University of Mississippi
rice@cs.olemiss.edu

ABSTRACT

A new one-semester course for graduate students introduces them to the field of computer audio from a computer science perspective. Students gain an understanding of sound, speech, and music and their digital representation and processing in computer software. Programming assignments provide hands-on experience in playing, recording, mixing, synthesizing, and analyzing audio. Class lectures present algorithms and techniques for compression, watermarking, synthesis, sonification, pitch and beat detection, audio fingerprinting, speech and speaker recognition, and audio retrieval.

INTRODUCTION

Over the past 15 years, audio technology has migrated from the world of electronics and specialized hardware to computer software running on general-purpose computers. Today virtually all personal computers support audio, and PC users are accustomed to audio output from computer speakers. Many users are familiar with audio input using a microphone and with editing audio on a computer. The ability to collect and play song recordings on a computer is enjoyed by millions of music listeners; however, the rampant free trade of these files via peer-to-peer Internet systems, such as Napster and Kazaa, threatens the music industry. That computer audio has become mainstream there is no doubt.

Audio technology, if taught at all in a university, may be addressed as part of a classical digital signal processing (DSP) course offered by the electrical engineering department. The course content typically reflects the hardware heritage of the field, with a focus on processes that are realizable in hardware. However, an unlimited variety of audio-processing algorithms can be implemented in software. It is time for the computer science department to get involved.

---

* Copyright © 2005 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

This paper describes a survey course on computer audio designed and taught by the author to a class of graduate-level computer science students during the Fall 2004 semester. The class was a hands-on introduction to audio programming and provided a survey of advanced audio topics. The goal of the class was for students to become comfortable working with audio, from learning the details of software implementation to understanding the larger issues of the field. The class was taught from an algorithmic, computer science perspective. While there are many DSP textbooks available, no suitable computer science textbook could be found, so the materials for this course were assembled by the author from many sources. The list of references at the end of this paper may assist others looking for resources.

AUDIO BASICS

No audio class can begin without an introduction to the physics of sound and its characteristics including sound pressure level, frequency, wavelength, phase, and speed. Audio is not sound but is the representation of sound, which may be analog or digital. Going from sound to digital audio in a computer requires a microphone to convert sound to analog audio together with an analog-to-digital converter. The reverse direction requires a digital-to-analog converter and a loudspeaker to convert analog audio to sound. The digital representation of sound is characterized by the number of channels (e.g., mono or stereo), number of bits per sample per channel known as the *resolution* (typically 8, 16, or 24 bits), and the number of samples per second called the *sample rate* (e.g., 8,000 Hz or 8 kHz). The analog-to-digital conversion may result in quantizing distortion, clipping, and/or aliasing [8,23].

The *waveform display* is a 2-D graph of the amplitude or loudness of a digital audio signal over time (see Figure 1). Measures of amplitude include the energy, power, and root mean square. The *DC offset* is the average of the sample values. It should be zero or close to zero; if it is not, it can be corrected by subtracting it from each sample value. Multiplying each sample value by a constant greater than one amplifies the signal; multiplying by a positive constant less than one attenuates the signal. Multiplying by an increasing value causes the audio to "fade in"; multiplying by a decreasing value causes it to "fade out."

In 1801, Jean Baptiste Fourier discovered that sounds can be described as a summation of sine waves. The *discrete Fourier transform (DFT)* of a digital audio signal represents these component sine waves and thereby describes the relative strengths of the frequencies present in the signal. Direct computation of the DFT takes $O(n^2)$ time, where n is the number of sample values; however, the Fast Fourier Transform (FFT) algorithm computes the DFT efficiently in $O(n \log n)$ time [4,16]. The *spectrum* is a 2-D graph showing the strengths of each frequency. A *spectrogram* is a 3-D graph that shows how the spectrum varies over time.

When a musical instrument plays the "A" above "middle C," a *fundamental frequency* or *pitch* of 440 Hz is produced; however, integer multiples of the fundamental frequency, called *harmonics*, are also produced at 880 Hz, 1320 Hz, 1760 Hz, etc. The relative strengths of the harmonics give each instrument its characteristic sound or *timbre*. The harmonics appear as peaks in the spectrum (see Figure 1). The musical scale is

logarithmic: going up one octave doubles the frequency, going down one octave halves it. The Musical Instrument Digital Interface (MIDI) and the MIDI file format represent musical notes to be played by sound synthesizers including sound cards [10].

Figure 1. Waveform display (left) and spectrum (right) of a note played by a bassoon.

In their first five assignments, the students were tasked with writing the following programs:

A1 – generate and play a sine wave
A2 – read a Wave file and play it
A3 – record sound captured by the microphone and save it as a Wave file
A4 – read two or more Wave files and mix them
A5 – compute the FFT of a Wave file and write the spectrum data to a file

Assignment A1 introduced the students to the "waveOut" routines for playing audio in Microsoft Windows, and Assignment A3 exposed them to the "waveIn" routines for recording sound; these routines are part of the Windows Multimedia library. Assignments A2 and A3 involved reading and writing audio files in uncompressed Wave format [10]. Assignment A4 introduced the students to audio mixing, which is achieved by averaging the sample values of two or more signals; each input was weighted to simulate the controls of a hardware mixing console. The students were given the source code to compute the FFT for Assignment A5; however, they needed to set up the FFT input and write the FFT output in a format suitable for graphing using Microsoft Excel.

AUDIO TRANSFORMATION AND SYNTHESIS

The goal of audio compression is to reduce the size of audio data. Each second of an uncompressed CD-quality recording requires 172K of storage. A lossy compression scheme offers reduced storage but at the cost of reducing the perceived audio quality. Popular lossy schemes, such as MP3, require only 4K to 40K of storage per second. However, the audio quality is acceptable to most listeners because the algorithms focus on discarding audio data that is not perceived. *Psychoacoustics* is the study of the human perception of sound [9]. From research in this field, it is known that a loud sound at one

frequency masks quieter sounds at neighboring frequencies. Perceptual-coding algorithms take advantage of this and other psychoacoustic principles to identify data to discard.

By contrast, *audio watermarking* is the process of embedding inaudible information in the audio data which identifies the origin and ownership of the recording. Its goal is to track copies of audio data and protect copyrights. The challenge is to embed the watermark in such a way that it cannot be removed either intentionally or unintentionally. A recent study of watermarking algorithms by the European Broadcasting Union indicates that this technology is in its infancy [14]. Compression algorithms seek to remove inaudible data so watermarking is at odds with compression.

Audio compression and watermarking seek to transform audio data without altering the perceived sound. Yet often the goal is to modify the sound, and there are many ways to accomplish it. The frequency content may be altered by applying a filter (low-pass, high-pass, band-pass, or band-reject) or a set of filters in a graphic or parametric equalizer. Audio-editing systems (also known as *digital audio workstations*) are to audio files what word processors are to document files. Working from a waveform display, the user may copy, paste, delete, mix, amplify, attenuate, fade, filter, and equalize any portion of an audio recording.

Techniques from the field of sound synthesis for creating and transforming sounds include modulation (amplitude, frequency, and ring), waveshaping, sound morphing, additive synthesis, physical modeling, and granular synthesis [2,15,19]. For Assignment A6, each student wrote a sound synthesis program that could utilize any combination of techniques, offering an opportunity for creative expression. The students demonstrated their programs in a mini-concert of interesting and unusual sounds.

Our look at sound synthesis would not be complete without also examining speech synthesis and sonification. The speech synthesizers available in Microsoft Windows XP were demonstrated and the challenge of producing natural-sounding speech was discussed. Sonification, also known as auditory display, is the synthesis of sounds to represent non-audio data, for example, weather or stock market data. The Geiger counter is the most well-known example [12]. We considered also algorithmic composition, in which a computer algorithm composes music. Such algorithms have been based on automata, grammars, Markov models, neural networks, and expert systems [19].

AUDIO RECOGNITION AND RETRIEVAL

There are many applications in which audio recordings must be analyzed, compared, classified, recognized, and searched. In Assignment A7, the students implemented a simple pitch-detection algorithm that estimates the fundamental frequency of an audio recording; such an algorithm could be used for tuning a musical instrument. We looked at algorithms for detecting beats, rhythm, and tempo in song recordings [5,6]. *Audio fingerprinting* is the process of identifying a song from a digital representation and is used to collect statistics about which songs have been broadcast and to detect unauthorized copies of copyrighted recordings [22].

The longest studied and best known of these applications is speech recognition. The accuracy of a speech-recognition system is affected by whether it must handle (1) a small

or large vocabulary; (2) a noisy environment; (3) discrete speech with unnatural pauses between words or continuous, natural speech; and (4) speech from many speakers with or without training or optimization for each speaker. Techniques used include vector quantization (VQ), template matching, and hidden Markov models (HMMs) [13,17,20]. The speech signal is typically represented by a time-based sequence of feature vectors and two sequences are optimally aligned using dynamic time warping [21]. In Assignment A8, students implemented a yes/no speech recognizer, a program that distinguishes whether a person says "yes" or "no." The *zero-crossing rate* is used to make the determination, which is the number of times per second the sequence of sample values crosses zero, i.e., changes its sign. The rate is higher for "yes" than "no" due to the high-frequency "s" in "yes."

A related problem is speaker recognition. In *speaker verification*, it must be determined whether the speaker is who he or she claims to be. This is an example of a biometric used to control access to a facility, computer system, or account. In *speaker identification*, the person speaking must be identified from among a group of known speakers. In *text-dependent* speaker recognition, a specific word or phrase must be uttered for comparison, whereas in *text-independent* speaker recognition, a voice match can be obtained by comparing different phrases; however, the former is more accurate than the latter [11,20].

Sequential listening to audio recordings is a tedious way to search them. Text descriptions of recordings can be searched by keyword to locate recordings of interest. However, it is burdensome to enter such descriptions; moreover, sounds are difficult to describe in words. Searching audio recordings based on the sounds they contain is called *content-based audio retrieval*. In *spoken document retrieval*, speech recordings are converted to text transcripts using speech recognition, and these transcripts are searched using text information retrieval techniques [1]. In *music information retrieval*, collections of song recordings are searched for similar melodies or genres [3,7]. Sound effects can be searched by a *sounds-like search*: given any sound as input, the system locates similar sounds [18].

CONCLUSION

The students in the class became comfortable and knowledgeable working with computer audio, and some have been motivated to pursue further studies in this area. For the author, developing the course without a textbook was a challenge.

In recent years, computer scientists have become more involved in this field; however, many of the books available today reflect the hardware orientation and background of their authors. Moreover, the books are focused on specific sub-topics of the field. A computer audio textbook for computer science students is needed which surveys the field from an algorithmic perspective.

REFERENCES

1. A. Coden, E.W. Brown, and S. Srinivasan, eds., *Information Retrieval Techniques for Speech Applications*, LNCS, vol. 2273, Springer-Verlag, 2002.

2. C. Dodge and T.A. Jerse, *Computer Music: Synthesis, Composition, and Performance*, 2nd edition, Schirmer, 1997.

3. J.S. Downie, "Music Information Retrieval," *Annual Review of Information Science and Technology*, B. Cronin, ed., vol. 37, Information Today, 2003, 295–340.

4. P.M. Embree and B. Kimble, *C Language Algorithms for Digital Signal Processing*, Prentice Hall, 1991.

5. J. Foote and S. Uchihashi, "The Beat Spectrum: A New Approach to Rhythm Analysis," *Proc. IEEE Int'l Conf. on Multimedia and Expo*, 2001, 1088–1091.

6. M. Goto and Y. Muraoka, "A Beat Tracking System for Acoustic Signals of Music," *Proc. Second ACM Int'l Conf. on Multimedia*, 1994, 365–372.

7. W.B. Hewlett and E. Selfridge-Field, eds., *Melodic Similarity: Concepts, Procedures, and Applications*, MIT Press, 1998.

8. T. Holman, *Sound for Film and Television*, Focal Press, 1997.

9. D.M. Howard and J. Angus, *Acoustics and Psychoacoustics*, Focal Press, 1996.

10. T. Kientzle, *A Programmer's Guide to Sound*, Addison-Wesley, 1998.

11. R.L. Klevans and R.D. Rodman, *Voice Recognition*, Artech House, 1997.

12. G. Kramer et al., *Sonification Report: Status of the Field and Research Agenda*, NSF report, 1997, www.icad.org/websiteV2.0/References/ nsf.html.

13. J.A. Markowitz, *Using Speech Recognition*, Prentice Hall, 1996.

14. A.J. Mason, "EBU Tests of Commercial Audio Watermarking Systems," presented at the 116th AES Convention, May 2004, preprint 6011.

15. E.R. Miranda, *Computer Sound Design: Synthesis and Programming Techniques*, 2nd edition, Focal Press, 2002.

16. J.G. Proakis and D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd edition, Prentice Hall, 1996.

17. L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

18. S.V. Rice and S.M. Bailey, "Searching for Sounds: A Demonstration of FindSounds.com and FindSounds Palette," *Proc. Int'l Computer Music Conf.*, 2004, 215–218.

19. C. Roads, *The Computer Music Tutorial*, MIT Press, 1996.

20. R.D. Rodman, *Computer Speech Technology*, Artech House, 1999.

21. D. Sankoff and J. Kruskal, eds., *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, 1983.

22. G.R. Schmidt and M.K. Belmonte, "Scalable, Content-Based Audio Identification by Multiple Independent Psychoacoustic Matching," *J. Audio Engineering Society*, vol. 52, no. 4, 2004, 366–377.

23. J. Watkinson, *The Art of Digital Audio*, 2nd edition, Focal Press, 1994.