A Content-Aware Sound Browser

Douglas Keislar, Thom Blum, James Wheaton, and Erling Wold
Muscle Fish, LLC
2550 Ninth St., Suite 207 B, Berkeley, CA 94710, USA
1-510-486-0141 inquiries@musclefish.com
http://www.musclefish.com

ABSTRACT: The SoundFisher™ browser is a cross-platform, single- or multi-user sound-effects database application. It incorporates an audio analysis engine that permits retrieval of sounds based on their acoustical similarity, as well as on traditional keywords and file attributes. Databases can be constructed from sounds on a local filesystem and/or the World Wide Web.

1. Introduction

When studios amass collections of sound effects and other sound files totaling hundreds of gigabytes, intuitive means of organizing and retrieving sounds become imperative. Traditional approaches have required time-consuming manual classification and organization. Describing the sound with keywords, while important, is not enough. What is needed is a system that can automatically compare, classify, and retrieve sounds. A number of researchers have investigated the problem of how to automatically classify and retrieve non-speech audio. (See references.)

This paper describes an audio engine and an end-user application, both already implemented, that permit retrieval of sounds based not only on traditional methods (keywords, soundfile header information, creation date, etc.), but also on the sounds' content—i.e., acoustic attributes. The end-user application is a multiplatform, multiuser sound-effects browser, called SoundFisher. (This paper does not mention the many features of the engine that are unused by SoundFisher. See Wold et al. 1999 for more information.)

2. Audio feature analysis and comparison

This section summarizes our technique for analyzing audio signals in a way that facilitates audio classification and search. For each frame of audio data (25 ms long, with a hop size of 10 ms) we measure a number of acoustic features of each sound. The analysis produces, over the course of the entire sound, a time series where each element is a vector of floating-point numbers representing the instantaneous values of the features. This sort of analysis works best when the sound is homogeneous in character, e.g., a door slam or rain. When analyzing a longer heterogeneous recording, e.g., a news broadcast, one can automatically segment the recording and compute a feature vector for each segment.

2.1 Frame-Level Features

The following features are currently extracted from each frame: loudness, pitch, brightness, bandwidth, and mel-filtered cepstral coefficients (MFCCs). The first three features were discussed in Keislar et al. (1995). Bandwidth is computed as the magnitude-weighted average of the differences between the spectral components and the centroid. A vector of MFCCs is computed by applying a mel-spaced set of triangular filters to the STFT and following this with the discrete cosine transform.

Since the dynamic behavior of a sound is important, the low-level analyzer also computes the instantaneous derivative (time differences) for all the aforementioned features.

2.2 Higher-Level Features

From the time series of frame values, we extract higher-level information. We compute the mean and standard deviation of the frame-level time series for each parameter, including the parameter derivatives. When computing the mean and standard deviation, the frame-level features are weighted by the instantaneous loudness so that the perceptually important sections of the sound are emphasized.

The user can present the system with a single sound for comparison, or with examples of a class of sound. In the latter case, we can infer something from the variability of the parameters across the different recordings. For example, there may be several samples of oboe tones, each at a different pitch. If one of these is presented as an example of an oboe sound to the system, the system has no a priori way of determining that it is the timbre of the sound that determines the class, rather than the particular pitch of this sample. However, if all the samples are presented to the system, the variability of the pitch can be noted across the samples and then used to weight the different parameters in comparison. This information can then be used when comparing new sounds to this class.
variability is stored in the standard deviation portion of the class’s feature vector.

For this single-Gaussian statistical model, the distance measure used is essentially the Euclidean distance between the two sounds’ feature vectors, with each dimension scaled by its standard deviation. The user is given the ability to apply additional weights to the different features, which is useful when certain features are known to be more (or less) pertinent to the task at hand.

3. The SoundFisher Sound-Effects Browser

This section presents the SoundFisher audio browser, which runs on the Macintosh, Windows (95, 98, and NT), and UNIX (Solaris, SGI) platforms. Written in Java, the SoundFisher GUI communicates with an included audio-analysis and database engine written in C.

Figure 1 shows the GUI for the application after a search has been performed. The row of buttons across the top provides functions such as “forward” and “back,” allowing navigation to previously displayed query results. Below the buttons is the query area, and the bottom portion of the window displays the query results.

A database is built up by adding URLs to it (either local files or Web addresses.) Directories can be added recursively; in a single step one can add all the sound files on a given disk, for example. The supported audio file formats include WAV, AIFF, AU, and Sound Designer II. When sounds are added, the engine analyzes the audio in the file or URL and stores the resulting feature vector in the database. Long sound files can be automatically segmented. In addition, “thumbnails” of sounds can optionally be generated. A thumbnail is a low-resolution, optionally truncated version of the source sound file. Thumbnails are useful for auditioning search results when the original sounds are offline.

The data record for each sound includes not only the acoustic feature vector but also soundfile information (sample rate, format, number of channels, duration, etc.), date, and textual keyword and comment fields. The text fields can be applied recursively when adding a directory. In addition, the user can define and add new text fields. Text fields can be edited at any time.

Users can create hierarchical categories of sounds. The default categories mimic the filesystem organization of the added directories, but the category names and hierarchy can be easily edited using a familiar graphical paradigm (e.g., Windows Explorer or the Macintosh Finder’s List view). These categories are arbitrarily defined by the user. The user can also create “classes”: sets of sounds whose acoustical feature vectors are close to sounds that the user has provided as a training set.

Below the top row of buttons is the query area, which is reminiscent of the Find File utility on Mac and Windows. Multiple criteria can be combined with a Boolean AND operation. A query is formed using a combination of constraints on the various fields in the database schema as well as “query by example” (comparison to a selected sound). For example, Figure 1 illustrates a query based on similarity to a selected sound (the noise of a crowd), in combination with a constraint based on a data field (duration). As indicated, the search can operate over the entire database, or it can apply to the currently displayed or currently selected records.

The bottom portion of the window displays the current records (often, the result of a query). Sounds can be auditioned by double-clicking, and multiple selections are possible. These results can be viewed in one of three ways: as a list (Figure 1), hierarchically by category, or as a 2-D plot (Figure 2). In the 2-D plot, the axes can be various acoustic attributes or the begin and end times of the sounds. The begin and end times are useful for automatically segmented sound files; by choosing begin time for an axis, one can view the temporal trajectory of a particular acoustic feature.

For more information, see the Muscle Fish Web site, www.musclefish.com.

References


Figure 1. Results of a content-based query, displayed as a list.

Figure 2. The same query results, displayed as a 2-D plot.