Proc. 2004 Int'l Computer Music Conference, U. of Miami, 11/04

Feature Extraction and Database Design for Music Software *

Stephen Travis Pope[1,2], Frode Holm[1,3], and Alexandre Kouznetsov[1,2]
1: Center for Research in Electronic Art Technology (CREATE)
University of California, Santa Barbara
2: FASTLab, Inc, Santa Barbara, California, USA
3: Predixis, Inc. Monrovia, California, USA
{stp, frode, alex}@create.ucsb.edu

ABSTRACT

Persistent storage and access of sound/music meta-data is an increasingly relevant topic to the developers of multimedia software. This paper focuses on the design of music signal analysis tools and database formats for modern applications. It is partly tutorial in nature, and partly a discussion of design issues. We begin with a high-level overview of the dimensions of music database (MDB) software, and then walk through the common g feature extraction techniques. A requirements analysis of several application categories will allow us to carefully determine which features might be most useful for them. This leads us to suggest concrete architectural and design criteria, and to close by introducing several of our recent implemented systems.

The authors believe that much current MDB software suffers due to ad-hoc design of analysis systems and feature vectors, which often incorporate only low-level features and are not tuned for the application at hand. Our goal is to advance the state of the art of music meta-data extraction and database design by fostering a better engineering practice in the construction of high-level feature vectors and analysis engines for music software.

1 Introduction

The careful design of music analysis frameworks for database-related musical software is too often overlooked. In many systems in the literature, the feature vector's design is not derived from the requirements of the application domain, or the desired database query criteria. Most prior work in sound or music databases has addressed a single kind of data (e.g., MIDI scores or sampled sound effects), and has pre-defined the types of queries that are to be supported (e.g., queries on fixed sound properties or musical features). This may indeed suffice for single-use MDB tools, but good engineering practice requires that we develop a set of application-domain-specific design patterns to foster reuse among feature extraction packages and databases. Furthermore, we believe that many recent implementations would benefit from a richer and higher-level feature vector.

In this paper, we present a high-level approach to the development of music analysis engines and database feature vectors that is based on requirements analysis of the target application. We will focus on the issues of meta-data feature extraction and feature vectors for areas such as user preference matching, genre classification, summarization, finger-printing, and feature mapping.

*This work is partially funded by a generous grant from the Matsushita Electric Industrial Company, administered by Miwa Fukino.

We will limit our discussion primarily to systems that extract musical meta-data from multi-instrument mixed musical sound (rather than notated scores, MIDI performances, sound effects, or speech), because we believe that these will increase in importance in the future, and that the technology is lagging behind that available to developers of other kinds of multimedia databases.

The structure of the paper is as follows. In section 3, we introduce some of the basic parameters of music database (MDB) applications. This is followed in section 4 by a discussion of the various domains of analytical features found in MDB systems, and then in section 5 by a presentation of the rough requirements of several common application domains. This then allows us to give a simple feature vector design decision tree in section 6. We close by describing several recent MDB systems we've implemented.

2 Background

Large-scale storage of sound and music data and meta-data has only become possible in the last decade. With this, and the new possibility for wide-area distribution of multimedia over the Internet, new requirements have arisen for flexible and powerful sound/music databases. A good overview of the field of music information retrieval (MIR) is presented by Jonathan Foote in (Foote 1999); George Tzanatakis' excellent tutorial (Tzanatakis 2002) is the best terse introduction to the signal processing of musical feature extraction.

The MIR literature shows musical meta-data being used (e.g.,) for applications requiring audio segmentation (Tzanatakis and Cook 1999), classification (Tzanatakis and Cook 2001), indexing and database query (Wold et al. 1999), content identification (Haitsma and Kalker 2002), summarization (Logan 2000). Current mature MIR/MDB systems often support several data formats, large and dynamic data volumes, query-by-example, analysis-on-demand, data replication or distribution, networked, distributed or p2p deployment, and web-oriented service architecture.

Since 1996, our work at CREATE has focused on database frameworks for multimedia applications, and on analysis and feature extraction techniques for music and sound databases used in a variety of application areas (Pope, Roy, and Orio 1999; Pope 2001; Pope and Ramakrishnan 2003). Our goal has been to develop a suite of analysis, storage, and query construction tools that scale to support large data volumes,

complex queries, and on-the-fly analysis in several flavors over several generations. Our projects have produced several database analysis and interaction tools that can be used together or separately, and address data types as diverse as scanned manuscript score pages, MIDI performance data, and 24/96-format 5.1-channel surround sound recordings. The most recent projects have been feature extraction systems based on taking mixed musical content (as MP3 or AIFF files) and producing databases of complex feature vectors for different applications.

3 Dimensions of MDB Applications

There is little unifying the field of music information retrieval other than the processing of some form of music-related data (sound, score, MIDI, etc.) in a software system that offers persistent storage and query capacity. In this text we will refer to the field itself as the domain of music meta-database (MDB) systems, since several classes of applications are not strictly "information retrieval" at all. Diverse applications such as user preference matching, music fingerprinting, or genre classification place very different requirements on the kinds of analysis performed, the format of the stored feature vectors, and the manner of making queries into the database.

The discussion of design methods for analysis and storage frameworks for MDB applications must incorporate both the expected I/O formats, the required kinds of indexing or queries, and the highest-level of abstraction or most general representational model that can be applied. The meta-data must allow the application to support its modes of data access given the expected inputs. The next sections describe some of the properties or defining characteristics that differentiate the classes of applications seen in the literature; these concepts will feed in to the design discussion that follows.

3.1 Content Format

MDB data formats may include musical scores (as images or encoded in some formal language), MIDI data (dead-pan or human-performed), one-voice monophonic melodies, and mixed multi-channel music. The input format is obviously a very important factor that determines the analysis that the system is able to perform. Much of the literature is dedicated to processing the simpler formats (e.g., MIDI or monophonic music), but in the systems we have built (and also see more frequently in the literature), we have generally concentrated on analysis of mixed and mastered multi-channel music in "mainstream" musical styles.

In any multi-stage system, there will be different source and meta-data storage formats, and analysis/clustering engines generally use multiple intermediate formats (feature vectors) internally. We differentiate the basic data categories of windowed, tracked, segmented, and averaged, which are illustrated in the examples that follow.

The determination of whether or not to keep the original source content gives rise to 1-, 2-, and multi-tier MDB architectures. Indexing applications such as MP3 players access compressed content using meta-data-based play-lists, whereas a classifier can discard the original content, keeping only the derived meta-data for clustering.

3.2 Low-level Analysis

Sound feature extraction generally starts with a mix of windowed low-level signal analysis and statistical operations on the window data. The goal is to take sound and generate a basic parametric signal representation based on short time windows (typ. 5 – 50 msec in length). An example would be performing windowed time-domain and frequency-domain analysis such as RMS envelope following and FFT-based spectral analysis, both using a medium-sized window (e.g., 1024 samples, ≈ 22 msec @ 44.1 kHz sample rate).

Given this lowest-level data, "medium-level" tools such as peak-finders and peak-trackers, inter-window signal derivatives, and a host of simple signal statistics can be derived from the windowed analysis to "add value" to the data. A more sophisticated system may perform multi-stage input analysis: filtering, data smoothing, and several kinds of time-/frequency-domain processing in one pass.

Additional DSP algorithms found in the recent literature (and described below) include linear prediction (LPC), pitch detection, wavelet analysis, and filter-bank analysis. These analysis stages produce feature data that can be stored directly, or used to compute higher-level features.

3.3 High-level Derived Features

Additional statistics that are "higher-level" in terms of the implied underlying model, are often be derived from the data delivered by the first-pass analysis functions. This process can be as simple as perceptual mapping of the low-level data (e.g., convert FFT spectral bins into weighted 1-octave spectra), or as complex as detailed signal segmentation and moving average distance functions between collections of the low-level features. In some cases, the high-level features are sufficient to perform the database search or classification tasks of the application.

These higher-level data can be constructed for a single time-slice (e.g., detailed spectral or formant analysis of a chosen frame), or added to the collection of time-sliced feature vectors that comprise a musical selection (e.g., inter-frame tempo tracking throughout a song). As we'll discuss below, instrument identification, tempo-tracking, and segmentation are three important higher-level techniques that operate on a combination of short-time and time-averaged feature vectors. Non-procedural programming techniques such as artificial neural networks, hidden Markhov models, or rule-based expert systems are often seen here, for example for use in tempo tracking or segmentation.

3.4 DB Design

There are obvious considerations that come into play as soon as the required data volume or throughput exceed certain limits. The choice of traditional relational database systems (RDBMS, e.g., Oracle, MySQL, or PostgreSql), as compared to more modern object-oriented database software (OODBMS, e.g., Objectivity, GemStone, ObjectStore, or Versant) can be made based on the current application's profile, or on the long-term data requirements analysis of a team or project family.

For MDBs, it is important to pay special attention to the selection of default indices for a group of tables or object sets, and to the techniques used for running stored procedures within

the database. Stored procedures can be especially helpful (i.e., efficient) if common distance metrics are known that help in clustering or searching the database.
Formal database software also provides features such as data replication, distribution, and enhanced data integrity, which cannot be overlooked as MDB systems are used in high-performance or high-value applications.

3.5 Application Flow

The profile of the application at hand determines the system-level subtasks that are required in an MDB system; most run-time applications rely on large pre-populated databases that can be used to search, match, build data distributions, apply various distance metrics for clustering, or perform classification. If audio queries are to be used to access the database contents, then run-time analysis is also required to generate a well-formed query (e.g., "query by humming" or sound-effect timbral similarity measures). In classification systems, a feature vector is generated from the input song, and then used to locate the nearest cluster or genre in the database.
The data flow within an analysis engine may be simple or quite complex and dynamic. Indexing applications often require no run-time analysis, and simply provide query-based access to a library or corpus. Streaming processing applications must clearly be able to "keep up" if they are to perform any kind of feature extraction or segmentation on a continuous stream. Constructing and populating a formal corpus database entails developing an analysis and segmentation engine, a clusterer, and a classification matcher, all in parallel.

4 Domains of Feature Extraction

In this section, we discuss each of the fundamental domains of sound signal analysis and feature extraction, and describe how they might be used for an exemplary MDB application. The goal is not to present the mathematics, but to investigate how they each map signal characteristics onto perceptually useful numerical features.

4.1 Time-domain Analysis

Windowed time-domain signal amplitude analysis, such as RMS or peak envelope extraction, and beat- or tempo-tracking, can be both computationally simple and musically effective. In fact, this data alone is sufficient to derive meta-data for many simple song segmentation, summarization, or matching applications. The processing of this analysis consists of selecting short segments (windows) of the content data, possibly applying a weighting function (e.g., a triangular window) to the samples, and then deriving some single statistic from the array, typically the peak value, weighted root-mean squared (RMS) value, or median rectified value.
Even this stage offers great flexibility, though; for example, one can split the musical signal into separate frequency bands (e.g., using a four-band filter-bank) and derive the RMS envelope and beat pattern from each of them separately. In popular musical styles, the time envelopes of the lowest and highest bands may correspond to the bass drum and cymbals, respectively.
Figure 1 below shows the windowed RMS and peak envelopes for the first ten seconds of a popular song with a very strong beat in the four-measure intro (first half of the window), followed by a first verse with loud sustained vocals and more constant spectral centroid (center-right of the screen dump). The screen shot is taken from the MDB visualization tools written by one of the authors (Pope) for the FMAK analysis framework (2004).

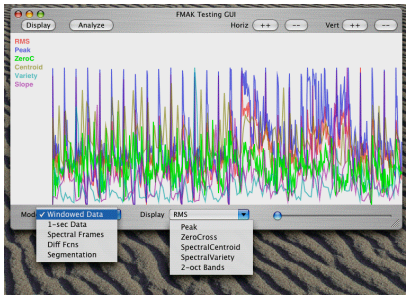

Figure 1: Windowed time-domain analysis example of the first ten seconds of a pop song: the list on the left shows the time- and frequency domain features. The first four measures (left half of the graph) show the strong 4/4 drum beat; the vocals enter near the middle of the screen with the sustained loud beats.

More musically meaningful and higher-level features can be derived from the core time-domain data; tempo curves and tempo transitions, the base time signature (and its changes), overall or section-by-section dynamic range, and fade-in/out times are all easily determined.
Moving to the next highest level, musical selections (esp. ones human-performed on acoustical instruments) can often be effectively segmented on the basis of tempo and dynamic range changes alone. Most of the technology to perform this analysis is well understood and stable, though there are still competing techniques for the derivation of the high-level tempo features, and their use in segmentation, and for the mapping of time-domain features to application-specific meta-data.
Latter-stage analysis often relies on time-domain tracking of frequency-, LPC-, or wavelet-domain data peaks. The same time-domain processes of tempo tracking or segmentation are applied to features derived from other domains. Conversely, in a multi-stage analysis engine, the initial time-domain segmentation of a selection can be used to direct and filter the further analysis stages.

4.2 Frequency-domain Analysis

Spectral analysis for feature extraction typically entails the windowed short-time Fourier transform (STFT) and/or linear predictive coding (LPC), possibly augmented by spectral peak extraction and inter-frame peak tracking.
Given a low-level spectral analysis, one can warp the linear-frequency-scale spectrum into one of several perceptually weighted spectra (1-octave bands, phon-weighted bands, etc.), or scale the spectral bands according to perceptual parameters.
As with other kinds of windowed data, an analysis engine will typically perform spectral peak detection and tracking. The peak location process within a frame may involve a simplistic histogram of the spectral data, or it may apply geometry-based techniques such as parabolic interpolation to achieve better estimates of peak frequencies. By the same token, the peak

tracking between frames (and the handling of peak births and deaths) can be quite simple, or very sophisticated, adaptive, and configurable.
Figure 2 below shows the LPC-derived spectrum of an eight-bar selection in which guitar chords can be tracked as spectral formants (lighter horizontal lines in the mid-spectrum). The screen shot is taken from the interactive tools written by one of the authors (Holm) for the FASTLab 1 framework (2000).

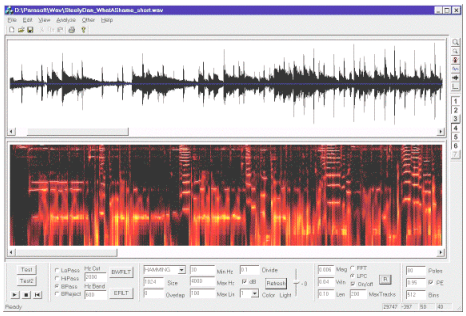

Figure 2: Frequency-domain analysis: the upper view shows the time-domain signal, and the lower the spectral display with tracked peaks visible as light horizontal lines in the spectrum.

Tracked peak birth/death statistics can very useful for tempo analysis and segmentation, since peaks in track births will generally correspond to note onsets. The ratio of the number of peaks to the number of tracked peaks also gives a good running indicator of the "noisiness" of a signal. Software developers should be especially cautioned about the use of such fine-grained spectral analysis on lossy-compressed material such as MP3 files, where much of the (masked) spectral content is filtered out.
If a full FFT-based analysis is not required, using filter banks for spectral analysis can be computationally simpler, and allows one to more easily tune the filter center frequencies and bandwidth to fit the requirements of the application at hand.
Linear prediction (LPC) analysis delivers smoothed spectral slices. LPC can be used for poly-instrumental music, but the data interpretation is much different than for its common usage in speech applications—spectral peaks do not represent the formats of a single instrument, and the LPC residual is not well-behaved, i.e., one cannot assume a single base pitch. All that being said, LPC can deliver useful data for mixed sources in the cases that formant-like spectral features are expected or need to be followed over time.
Further analysis often tries to characterize specific windows of a signal, for example in an attempt to identify the spectral signature of the solo instrument (or the main percussion instruments) in a musical selection.
Pitch tracking is another form of frequency-domain analysis. While the general problem of robust polyphonic pitch detection is still considered intractable, pitch detection techniques such as autocorrelation, FFT(FFT) cepstrum, or spectral peak interpretation are often used on one-voice signals, or on specially filtered versions of mixed signals (e.g., to follow a bass line). In melody databases (or in general-purpose databases that need to support melodic search or "query by humming"), this may be the only form of analysis performed. An analysis engine can use other features (e.g., segmentation) to guess where a melodic line can be expected, and then filter the signal to a selected frequency range and attempt pitch detection.
Bass pitch tracking is useful for many kinds of music, and enables the derivation of "harmonic key histograms" that relate to musical structure. Figure 4 below illustrates the results of pitch-tracking on a severely low-pass filtered signal that accurately captures the bass line of a 12-bar blues in C.

4.3 The Wavelet Domain

Wavelet-domain signal decomposition is gaining popularity as we learn better how to interpret and use collections of wavelet coefficients. We consider it separately from the time-domain and spectral-domain because it combines features of both other analysis domains in a single representation. This transform generally convolves the input signal with each member of a family of related signal windows (called the wavelet kernel); the members of the family share the same shape but have different time-scales and amplitudes. The convolution serves to decompose the signal into a summation of "grains" at different time/frequency scales.
The statistical characteristics of the wavelet coefficients, the "constant-Q" time/frequency profile of grains, makes this the preferred choice as a hierarchical time/frequency representation of a signal. The process can be applied to derive tempo estimates, pitch estimates, or to characterize the "noisiness" of a signal.
Figure 3 below (from Kling and Roads 2004) illustrates the difference between FFT-based spectral analysis and wavelet decomposition. The top graph is the time-domain waveform of a single short note, the middle the wavelet decomposition, and the bottom the 256-sample-window STFT. Both time- and frequency-smearing are apparent in the STFT decomposition.

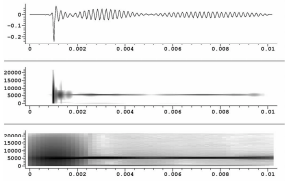

Figure 3: Time-, wavelet-, and FFT-domain version of a signal with a strong transient; note the excellent time and frequency localization of the wavelet components in the middle plot.

4.4 Cross-domain Analysis

It is not uncommon to combine the basic analysis processes into multi-stage analysis procedures, for example to use the results from the time-domain analysis (a guess at the basic song structure) to tune further analysis in other domains (e.g., to take a close look at the spectrum and formants of the first note of the solo instrument or vocal track that can be located easily near the middle of Figure 1 above). By the same token, the output of the peak tracker illustrated in Figure 2 (regularly spaced periods of high levels of track births/deaths) could be fed back and added to the time-domain information in a segmenter. The choice of possible analysis paths suggests a black-board- or agent-based meta-analysis "expert system" to drive the analysis engine.

Some applications profit from (or indeed require) the identification of certain "genre-indicator" instrument signatures (e.g., banjo, pedal steel guitar, pipe organ, harpsichord, grunge guitar, wah-wah guitar, latin percussion, etc.) or may add ensemble analysis (i.e., a guess at what and how many instruments are playing) or recording production and spatial analysis (i.e., guess how the sound was recorded and mastered). Related to this is the frequent desire for voice-location, pitch tracking, and vocal style analysis, which is simple for single-voice well-isolated recordings, but still very challenging for general-purpose mixed and mastered content. There is still much to be done in these areas, both at the signal processing level and the higher-level feature derivation task.

The general trend is away from small fixed feature vectors and towards more complex hierarchical analysis engines that incorporate a set of heuristics as to what kinds of analysis to perform on what kinds of data. To the extent that one can characterize (and limit) a database's content *a priori*, one can still design minimal analysis engines, but more and more frequently, systems are being built and databases populated with the hope that the systems will support diverse kinds of content and multiple domains of indexing or query.

4.5 Data Smoothing

Natural audio signals, and derived features as well, generally contain significant levels of noise; we have already mentioned a number of uses for data smoothing, and the desire for time-function data reduction, or "forgiving" peak tracking. The smoothing/averaging technique chosen must be appropriate to the profile and noise characteristics of the signal. As an example, look at the fluctuating and noisy plot in the left-hand graph of Figure 4, the pitch tracker output for the bass line of a 12-bar blues tune taken from the FMAK analysis framework. There is obviously a consistent pattern, with various kinds of blips and out-of-range errors. A statistical smoothing function that combines running average differentiation with constant-range extension produces the smoothed plot in the right-hand view; these data can easily be justified to well-tempered pitches or even correlated with the attacks in the RMS level to produce a MIDI bass line.

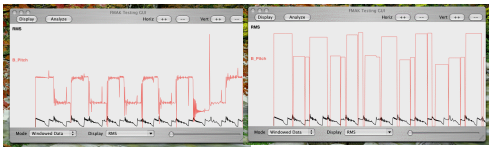

Figure 4: Data smoothing at work: before and after bass pitch plots of a 12-bar blues recording (C/G/C/G … C/F/G/C) that show the statistical cleaning process. The smaller-scale data is the RMS envelope of the filtered notes, which correlates quite well to the pitch changes.

4.6 Segmentation and Musical Form

Signal segmentation is both an application in itself and a technique that has proven very valuable as a component in other applications. The simplest segmenters use windowed low-level time- and/or frequency-domain features to locate significant changes in the signal by using one or more inter-window distance metrics, functions that take multiple features (see e.g., the combination of curves shown in Figure 1) and calculate how different each frame or frame group is from the next or previous ones. One can think of this process as taking the time derivative of the signal using a weighting function in the multi-dimensional feature space.

. This can be very effective for processing speech, or for locating very significant and abrupt changes in a signal (e.g., silence detection or voice/music switches), but it is still difficult to take an arbitrary selection of music and reliably determine its musical structure, to guess where the sections or verses start.

If the signal is assumed to include regularly spaced segment boundaries (i.e., repetitions of segments of similar lengths as in the verses of a song), then the segmenter may use autocorrelation of time-averaged inter-frame difference functions to guess at the longest common segment length. The data about segment length and regularity can be very useful in classification and matching tools, for example.

Figure 5 below shows the output of a simple speech segmenter. In the lower pane one can see the signal spectrogram and the RMS amplitude overlaid on it. In the upper view, the reduced 1-octave spectrum is shown, and the bold vertical lines show where the segmenter placed inter-phoneme segment points. The utterance is "ice melts," and the system correctly segments it into its constituent phonemes. The screen shot is taken from the display tools written by one of the authors (Pope) for the 8S analysis framework (2002).

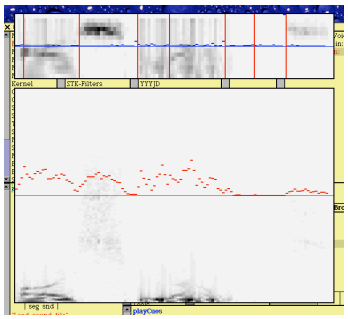

Figure 5: Signal segmentation: the bottom graph shows the spectrogram and RMS envelope of the utterance "ice melts." The top region is the 1-octave reduced spectrum with discovered inter-phoneme segmentation points shown as vertical bars.

4.7 Very Large Feature Vectors

If the requirements are well understood at the start of database population, the feature vector can be limited to include only those features deemed necessary to the application. Increasingly, however, some MDB systems are designed to support a wide variety of applications and query types, meaning that a large and rich feature vector will be needed. In our experience, the database volume for feature vector storage can easily exceed the volume of the original content, at least in the intermediate stages of analysis processing.

This leads to an explicit stage of "feature vector pruning," removing parts of the analysis results that are no longer needed or have been deemed unimportant, i.e., they are masked or included in higher-level averages.

There is also an important difference between very complex and large feature vectors, and very high-level (and therefore potentially compact) feature vectors. The literature is tending towards the derivation of ever-higher-level (more structured)

perceptual and psychological features (i.e., "mood"), even though these are often very hard features to derive in any reliable manner, or to map onto other feature spaces.

The final stage of an MDB analysis engine may involve storing a reduced and pruned feature vector in a database (e.g., during the database population phase), or using it to generate a query into a pre-populated database (e.g., in a run-time matching or indexing application).

5 Application Requirements

Given the introduction of the basic parameters of music database applications, and the preceding outline of signal analysis techniques, we can now proceed to a discussion of application-specific feature vector design. We will start here with a description of the requirements of several common application areas, and then present some general guidelines for feature vector developers.

5.1 Players and Indexing

The most prevalent MDB application today is certainly the MP3 player such as iTunes, running on a PC or a portable device. Compressed audio content and manually generated textual meta-data are stored separately and used for indexing and the creation of play-lists. State-of-the-art systems (Pachet and Roy 1999; Predixis 2004) use on-line analysis and extended meta-data for "smart" automatic play-list generation.

The core processing of indexing applications is typically a matching engine that searches a local database based on a distance metric weighting function derived from user input (e.g., a set of selections). To support a large volume of local data, the stored feature vectors must be compact, but they must also allow genre-based, sylistic, or "sounds-like" queries and matching. This is one of the most challenging areas for the next generation of MDB analysis tools because one needs a very compact feature vector (due to storage and WAN connection constraints) that maps well to style, mood, or user preference, and normally has to run the analysis on limited hardware.

5.2 Database Queries and Search Engines

Applications and web services for accessing databases of mixed multi-channel music often still rely on text annotation of musical genres, or on classifications added manually. The first method is problematic because music providers generally supply only very vague or general genre classifications (and generally give only one genre per album or CD, rather than per-song), and the second method fails because it relies on an "army of ants" that listen to short excerpts of music selections and classify them manually.

Search engines that support text-based queries generally use such manually provided meta-data (e.g., GraceNote), whereas melody-based queries (query by humming) systems require analysis of user input to generate the initial database query (Ghias et al. 1995). Search engines are also found that support timbral queries (sound effect location, "sounds like") (Wold et al. 1999).

For the case where a limited number of features suffice to execute the database queries, a relatively simple, non-segmented feature vector can be used. Melodic databases are a special case since they sometimes place special requirements on the source material (one-voice, carefully recorded) and the batch-phase and run-time analysis (monophonic pitch tracking). We can expect to see software soon that couples sung and pitch-tracked input with automatically derived main melody summarizations of mixed music.

Successful genre-similarity (e.g., preference-matching systems) search engines require a complex feature vector, and need good discriminator functions or distance metrics to be useful. Search-on-genre systems that are independent of text meta-data are rarer (Predixis 2004), but we believe users will come to expect this kind of functionality.

5.3 Summarization

The goal of music summarization (or "thumb-nailing") systems is to locate the "most typical" short (in the 2-5 second range) segment of a musical selection (Logan 2000). This typically requires segmentation at the song-structure level, but it need not be very sophisticated if the system is limited to mainstream musical styles. Summarization may concentrate on the identification of a "typical" section or verse, as could be done using the time-domain feature data illustrated in Figure 1.

Given the choice of the most representative segment of a selection, one might store the original content excerpt, or just a single feature vector, for later use. More detailed analysis may be used to try to locate and track the main melody in this segment, or to identify what the lead instrument or voice is.

5.4 Content Identification and Matching

A system that performs content matching or music finger-printing could theoretically use an algorithm as simple as asking "what's the value of sample number 1000?" or "what are the peaks of a single spectral slice selected from the middle of the song?" To be robust in the face of transformations of the material, encoding,, and even deliberate spoofing, however, often requires a much more complex feature vector (Haitsma and Kalker 2002). The feature vector must be chosen so as to be invariant across the range of expected transformations. A reliable finger-printing system would incorporate high-level perceptual properties, and possibly also song segmentation and melody instrument tracking, in the feature vector.

5.5 Clustering and Classification

Clustering means grouping a database's contents into a number of data "clusters" based by finding a feature vector weighting that divides the data along some set of axes in higher-dimensional space. Once found these data clusters may or may not correspond to actual musical genres (e.g., speech vs music, or classical/jazz/rock), and labeling tools are often provided to pick and label clusters that are related 1:1 to genres. This exercise qualifies as "heavy lifting" in the data mining literature, given the large data volumes seen in MDB systems, and the complexity of our feature vectors. We hope to see improved results in this area in the coming years.

After clustering, the database may be pruned, discarding most of the data but maintaining those items that best define the clusters. A classifier system now has to take an input song, run a (possibly simplified) analysis engine, and locate the cluster in the pruned database that most closely matches the song's weighted feature vector.

Preference-matching systems, for example, are required to do genre-signature (or artist-signature, or song-signature) matching, but do not need labeled genre clusters. Any detailed genre classification requires a much more complex feature vector, often involving the identification of song form, genre-indicative instrument signatures, and detailed tempo and rhythm tracking.

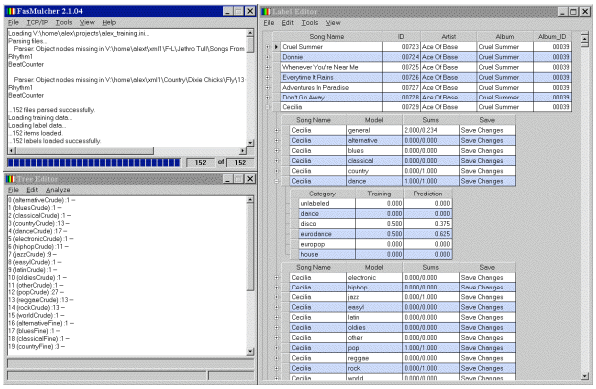

Figure 6: Database clustering and genre labeling: tool for training a CART expert system to recognize musical styles to a fine-grained (> 100 genres) model with a segmented feature vector. One the left are the reasoning log and current style lattice views, and the right shows the selected cluster members for audition

In Figure 6 above, one sees a tool that uses Categorization and Regression Trees (CART) to help the user locate and label clusters in a large database of popular songs. The screen shot is taken from the expert system training tools written by one of the authors (Kouznetsov) for the FASTLab 1 analysis framework (2000).

5.6 Mapping

In most of the above applications, the feature vector is only used for a single query or match. Feature mapping systems, however, use the feature vector and genre classification to plan signal processing such as post-production mastering, 2-to-5-channel up-mixing, or surround sound spatialization. In these cases, the feature vector must support fine-grained genre classification as well as capturing details of the mixing and production of the content (e.g., analysis of the instrumental ensemble, the stereo/surround mixing, the hall's reverberation time, and mastering dynamic range compression).

6 MDB Application Design

The preceding sections introduced some of the considerations in the principled design of feature extraction software for music. As an initial step in the formal design process, we will discuss the most basic decisions that come up in MDB construction.

Scope. What is the unit of musical content that we need to address? Are we capturing whole selections (songs) or not? Is the underlying content to be served, or just the meta-data? To what extent can we predefine or limit the source content? The answers to these questions will determine the initial scope of the analysis system and required infrastructure.

IO format(s). What is the input format? Is the input static or streamed? Do we need to maintain the original content after analysis? Do we get the define the output (DB) format from scratch? Has the input ever been lossily compressed?\ The I/O format has central effect on the choice of overall architecture.

Low-level features. Which analysis domains will be most important: time, FFT, LPC, wavelet, etc.? How much can we limit the analysis process (in the interest of performance)? The first-pass analysis engine design is derived from the basic properties of the simplest feature vector required.

Derived features. What, if any, higher-level features are required to be derived from the low-level analysis results? Should we track any data between analysis frames? Do we need to identify instrument signatures, or locate style-indicative instruments? Should we try to extract tempo curves or track harmonic change? A multi-pass analyzer or manager expert system can tune the use of further signal analysis or statistical routines.

Perceptual features. Are there special perceptual features (e.g., warped spectra) that we have to derive? Psychoacoustically derived feature statistics can be applied to the lower-level features.

Segmentation. Does the application suggest the need for accurate content segmentation or not? Do we store one or multiple feature vectors per selection? Segmentation invites reduction or pruning of the windowed feature data.

Back-end database. What will we end up storing? Do we need a content corpus of music, or clustered labeled genre specifications, etc.? Do we store the entire feature vector, or just a pruned subset of it? Database insert/select statements can generally be easily defined on the basis of the chosen feature vector or feature collection.

We can use these starting parameters to drive our design decisions while planning and implementing an MDB system, or incorporating MDB features into an existing application. Going back to the application discussion of section 5 and the overview of signal analysis techniques in section 4, one can put together a formal algorithm for analysis engine and database design for future MDB applications.

7 Examples: 8S and FMAK

In this section, we'll discuss the design of two music feature extraction systems we have built that targeted very different application areas (see [Pope, Roy, and Orio 1999] for background). The goal is to show the design principles we've introduced in action.

7.1 The 8S Segmenter and Database in Siren

The "Siren Speech Segmenter for *Sensing/Speaking Space* and *Sleeping Sword*" (**8S**, Pope 2001) is a two-stage speech/music segmenter and analysis database. In the first stage a simple feature vector is derived using only RMS envelope extraction and a 1-octave band-reduced spectrum analysis. The goal of this stage is to segment a monophonic speech signal into individual phonemes. The analysis meta-data and the resulting segmentation points were illustrated in Figure 5 above.

In the second stage, a single more complex and perceptually weighted feature vector is derived for each phoneme; this involves detailed noise analysis, LPC formant tracking, and

envelope template matching. These stored features are then used by compositional tools to search the database using different phoneme similarity functions (distance metrics that combine the features stored in the database). The tools use distance metric functions (kept as stored procedures in the database back-end) to collect groups of similar phonemes efficiently, and then use the sound file pointer information in the database to locate the actual sound cue corresponding to a given phoneme. These can be mixed together to create new speech, or many kinds of "warped" speech using phoneme replacement.

7.2 The FASTLab Music Analysis Kernel

The initial target application for the FASTLab Music Analysis Kernel (FMAK) was a mapping expert system that uses the derived features to plan several stages of signal processing for music mastering and post-production. Due to the complexity of the task, and the need for very robust segmentation and fine-grained classification, we decided on a very rich feature vector, with the resulting increase in database volume, improved flexibility in building distance metrics for segmentation, and complex and slow clustering.

The FMAK feature extraction process proceeds through several standard analysis stages; in the time domain, it performs windowed RMS envelope extraction at several time scales and frequency bands (see Figure 1), as well as gathering stereo and surround spatial information. FFT spectra are derived, and spectral peaks are extracted and tracked between frames. The peak extractors and trackers are configurable, and simple heuristic rules tune the processing as necessary to get meaningful results. The common FFT-based statistics are provided: spectral centroid, flatness measure, spectral variety, etc. LPC coding is performed with inter-frame tracking of formant peaks, and the LPC residual noise estimate is maintained. For the trackers, track birth and death statistics are kept, as these can be useful for segmentation. The discrete wavelet transform (DWT) is then used to get a noise estimate and some idea of the musical beat structure. We store the data for each window that is delivered by the analysis in a feature table object. The analysis engine supports different window and hop sizes for each stage of the low-level feature extraction, and results in a huge volume of data; the initial windowed feature collection is typically larger than the uncompressed audio file.

In the second stage of FMAK, a segmenter tries to discover the structure of the selection; in mainstream musical styles, this will be the verse/chorus structure. The segmenter tries several distance metrics to achieve well-delineated sections (i.e., choose a distance function that gives a good dynamic range), and then tries to find a periodicity among the segment boundaries. The result is a list of segment break points, and a confidence factor for the segmentation.

If the segmentation process was successful, we can now use the segmentation data to "prune" the huge collection of per-window feature data, leaving only an average and possibly a peak feature vector for each "verse" of the musical selection. We can also derive several additional structure-related features based on the segmentation, such as statistics about the fade out or repeated sections in the song.

The result is what we call the feature collection, an object (and a corresponding row in a database table) that contains the meta-data (name, artist, title, album, etc.), the song's average, peak and/or typical feature table data, a single-note feature table for some selected "solo instrument" window, the segment-length vector and confidence, and the structural features described above. This processing is used for database population as well as for run-time analysis for query generation. Separate tools perform database clustering, labeling, and pruning, and support run-time genre classification.

We hope to be able to use the database we have constructed using the FMAK analysis kernel for other applications, such as 2-to-5-channel surround sound up-mix, or more traditional "music information retrieval" applications in digital media libraries.

8 Conclusions

The variety of MDB applications calls for a more detailed design practice for feature extraction systems and feature vector and database designs. A formal engineering discipline of feature vector design is needed for the field of music/sound databases to progress beyond its present ad-hoc stage.

In this paper, we have surveyed the domains of MDB systems, looked at the basic features that can be extracted from musical signals and how they can be used, and developed a rough set of questions to aid the designers of future feature extractors. We concluded by discussing two of our recent developments in terms of their design for very different applications.

References

Foote, J. 1999. "An Overview of Audio Information Retrieval." Multimedia Systems, 7(1): 2-11.

Ghias, A, et al. 1995. "Query By Humming: Musical Information Retrieval in an Audio Database." *Proc. ACM Multimedia 95.*

Haitsma, J., and T. Kalker. 2002. "A Highly Robust Audio Fingerprinting System." *Proc. ISMIR 2002.*

Logan, B. 2000. "Music Summarization using Key Phrases." *Proc. ICASSP 2000.*

Kling, G., and C. Roads. 2004. "Audio Analysis, Visualization, and Transformation with Matching Pursuits." *Proc. DAFX-04.*

Pachet, F. and P. Roy. 1999. "Automatic Generation of Music Programs." *Proc. CP'99.*

Pope, S. T., P. Roy and N. Orio. 1999. "Content Analysis and Queries in a Sound and Music Database." *Proc. ICMC 1999.*

Pope, S. T. 2001. "Music and Sound Processing Using Siren." In M. Guzdial and K. Rose, eds. 2001. *Squeak: Open Personal Computing and Multimedia.* Prentice-Hall, pp. 377-421.

Pope, S. T., and C. Ramakrishnan. 2003. "The CREATE Signal Library: Design, Issues, and Applications." *Proc. 2003 ICMC.*

Predixis, Inc. 2004. *Predixis MusicMagic Mixer/Browser User's Manual.* http://music.predixis.com/4_help_usermanual.html

Tzanetakis, G. 2002. "Tutorial: MIR for Audio Signals." *Proc. ISMIR 2002.*

Tzanetakis, G., and P. Cook. 1999. "Multifeature Audio Segmentation for Browsing and Annotation." *Proc. IEEE WASPAA99.*

Tzanetakis, G., and P. Cook. 2000 "Audio Information Retrieval (AIR) Tools." *Proc. ISMIR 2000.*

Tzanetakis, G., and P. Cook. 2001. "Automatic Musical Genre Classification of Audio Signals." *Proc. ISMIR 2001.*

Wold, E., T. Blum, D. Keislar, and J. Wheaton. 1999. "Classification, Search and Retrieval of Audio." in Furht, B., ed., *Handbook of Multimedia Computing*, pp. 207-225, CRC Press.