19

Sound Classification and Similarity

Michael A. Casey

General audio consists of a wide range of sound phenomena such as music, sound effects, environmental sounds, speech and nonspeech utterances. The sound recognition tools provide a means for classifying and querying such diverse audio content using probabilistic models. This chapter gives an overview of the tools and discusses applications to automatic content classification and content-based searching.

The sound classification and indexing tools are organized into low-level descriptors (LLD), AudioSpectrumBasis and AudioSpectrumProjection, and high-level description schemes (DSs), SoundModel and SoundClassificationModel, which are based on the ContinuousHiddenMarkovModel and ProbabilityClassificationModel DSs defined in the Multimedia Description Schemes (MDS) document. The tools provide for two broad types of sound description; text-based description by class labels and quantitative description using probabilistic models. Class labels are called terms and they provide qualitative information about sound content. Terms are organized into classification schemes, or taxonomies, such as music genres or sound effects. Descriptions in this form are suitable for text-based query applications, such as Internet search engines, or any processing tool that uses text fields. In contrast, the quantitative descriptors consist of compact mathematical information about an audio segment and may be used for numerical evaluation of sound similarity. These latter descriptors are used for audio query-by-example (QBE) applications. They can be applied to many different sound types because of the generality of the low-level features. We start by discussing these LLD.

19.1 SPECTRAL BASIS FUNCTIONS

The frequency spectrum is a widely used tool in audio signal processing applications. However, the direct spectrum is generally incompatible with automatic classification methods because of its high dimensionality and significant variance for perceptually similar signals. Consider, for example, the AudioSpectrumEnvelope descriptor in which each spectral frame is an \( n \)-dimensional vector. A one-fourth – octave spectrum requires around 32 dimensions per frame at a sample rate of 22.05 kHz, but probability classifiers require data to occupy 10 dimensions or fewer for optimal performance. A lower resolution
representation, such as the octave bandwidth spectrum, reduces the dimensionality to 8 coefficients but disregards the narrow-band spectral information essential to many sound types. The solution is a representation that performs a trade-off between dimensionality reduction and information loss. This is the purpose of the `AudioSpectrumBasis` and `AudioSpectrumProjection` low-level audio descriptors.

The `AudioSpectrumBasis` descriptor contains basis functions that are used to project spectrum descriptions into a low-dimensional representation. The basis functions are decorrelated features of a spectrum with the important information described much more efficiently than by the direct spectrum. The reduced representation is well suited for use with probability models because basis projection typically yields useful features that consist of 10 dimensions or fewer.

In the simplest extraction method, basis functions are estimated using the *singular value decomposition* (SVD). The SVD is a well-known technique for reducing the dimensionality of data while retaining maximum information content. The SVD decomposes a spectrogram, or collection of spectrograms belonging to a sound class, into a sum of vector outer products with vectors representing both the basis functions (eigenvectors) and the projected features (eigen coefficients). These basis functions and projection coefficients can be combined to form eigenspectra, which are, in themselves, complete spectrograms but they contain mutually exclusive subsets of the original information. The full complement of eigenspectra sum to the original spectrogram with no information loss.

A subset of the complete basis is selected to reduce spectrum dimensionality. The loss of information is minimized because the basis functions are ordered by statistical salience; thus, functions with low information content are discarded. Figure 19.1 shows

![Figure 19.1](image)  
*Figure 19.1* A four basis component reconstruction of a spectrogram of pop music. The left vectors are basis functions and the top vectors are the corresponding projected features.
a spectrum of 5 seconds of pop music reconstructed using the first 4 out of 32 basis functions.

The functions to the left of the spectrogram are the basis functions, those above the spectrogram are the projected features that are used for automatic classification. Here, 70% of the original 32-dimensional data is captured by the 4 sets of basis functions and projection coefficients. Figure 19.2 shows the extraction system diagram for both AudioSpectrumBasis and AudioSpectrumProjection. Note that for classification applications, the normalization coefficients are stored with the projected features thus increasing the dimensionality by 1.

While the SVD is sufficient for most applications, an optional independent component analysis (ICA) step performs a transformation of the SVD basis functions yielding maximum separation of features. The ICA transform preserves the structure of the SVD feature space while offering a statistically independent view of the data; independence is a stronger condition than the de-correlation constraint imposed by the SVD [1–4].

The following code example shows an instance of the AudioSpectrumBasis descriptor for a 32-dimensional spectrum using a subset of 4 basis functions:

```xml
<AudioDescriptor xsi:type="AudioSpectrumBasisType"
    loEdge="62.5" hiEdge="8000" octaveResolution="1/4">
    <SeriesOfVector totalNumOfSamples="1" vectorSize="32 4">
        <Raw dim="32 4">
            0.082 -0.026 0.024 -0.093
            0.291 0.073 0.025 -0.039
            0.267 0.062 0.030 -0.026
            0.267 0.062 0.030 -0.026
            0.271 -0.008 0.039 0.007
            0.271 -0.008 0.039 0.007
            0.269 -0.159 0.062 0.074
            <-- more values here ... -->
            0.010 -0.021 0.063 -0.103
        </Raw>
    </SeriesOfVector>
</AudioDescriptor>
```
The next code example shows an instance of the `AudioSpectrumProjection` descriptor representing features derived from the basis vectors in the previous example:

```xml
<AudioDescriptor xsi:type="AudioSpectrumProjectionType">
  <SeriesOfVector hopSize="PT10N1000F" totalNumOfSamples="263" vectorSize="4">
    <Raw dim="263 4">
      0.359 -0.693 0.345 -0.145
      0.364 -0.690 0.308 -0.147
      0.353 -0.656 0.382 -0.175
      <!-- more values here ... -->
      0.998 -0.342 0.569 0.592
      1.000 -0.324 0.562 0.601
    </Raw>
  </SeriesOfVector>
</AudioDescriptor>
```

### 19.2 SOUND CLASSIFICATION MODELS

The first step toward automatic classification is to define a set of categories and their relationships. The `SoundClassificationModel` DS consists of sound models, each with an associated category term, organized into a hierarchical tree; for example, `people`, `musical instruments` and `animal` sounds. Each of these classes can be broken into narrower categories such as: `people:female`, `animals:dog` and `instruments:violin`.

Figure 19.3 shows musical instrument controlled terms that are organized into a taxonomy with ‘Strings’ and ‘Brass’. Each term has at least one relation link to another term. By default, a contained term is considered a narrower term (NT) than the containing term. In this example, ‘Fiddle’ is defined as being a nearly synonymous with, but less preferable than, ‘Violin’. To capture such structure, the following relations are available as part of the `ControlledTerm` DS:

- **BT** – *Broader Term*: The related term is more general in meaning than the containing term;
- **NT** – *Narrower Term*: The related term is more specific in meaning than the containing term;

![Figure 19.3](image_url)  
*Figure 19.3: Part of a Musical Instrument classification scheme*
• **US – Use**: The related term is (nearly) synonymous with the current term but the related term is preferred to the current term;

• **UF – Use For**: Use of the current term is preferred to the use of the (nearly) synonymous related term;

• **RT – Related Term**: Related term is not a synonym, quasi-synonym, broader or narrower term, but is associated with the containing term.

The purpose of the classification scheme is to provide semantic relationships between categories. As the scheme gets larger and more fully connected the utility of the category relationships increases. Figure 19.4 shows a larger classification scheme including animal sounds, musical instruments, people and Foley (sound effects for film and television). By descending the hierarchical tree we find that there are 19 leaf nodes in the taxonomy. By inference, a sound segment that is classified in one of the leaf nodes inherits the category label of its parent node in the taxonomy. For example, a sound classified as a dog:bark also inherits the label animals.

The following code example shows a number of SoundClassificationModel DS instances corresponding to the classification scheme shown in Figure 19.4. The classifiers are organized hierarchically with high-level classes preselecting the classifiers for low-level classes. Category terms are contained inside the SoundModel DS instances, see below. Here, references to SoundModel instances are used for brevity.

```xml
<!-- General Audio Classifier, highest level categories -->
<AudioDescriptionScheme xsi:type="SoundClassificationModelType"
 id="IDClassifier:GeneralAudio">
 <SoundModel SoundModelRef="IDPeople"/>
 <SoundModel SoundModelRef="IDMusicalInstruments"/>
 <SoundModel SoundModelRef="IDAnimals"/>
 <SoundModel SoundModelRef="IDFoley"/>
 </AudioDescriptionScheme>

<!-- Human sound classes -->
<AudioDescriptionScheme xsi:type="SoundClassificationModelType"
 id="IDClassifier:People">
 <SoundModel SoundModelRef="IDSpeech:Male"/>
 <SoundModel SoundModelRef="IDSpeech:Female"/>
 <SoundModel SoundModelRef="IDCrowds:Applause"/>
 <SoundModel SoundModelRef="IDPeople:FootSteps"/>
 <SoundModel SoundModelRef="IDPeople:Laughter"/>
 <SoundModel SoundModelRef="IDPeople:ShoeSqueaks"/>
</AudioDescriptionScheme>
```

Figure 19.4  A hierarchical taxonomy consisting of animals, music, people and Foley classes
SOUND CLASSIFICATION AND SIMILARITY

314

19.3 SOUND PROBABILITY MODELS

Spectral features of a sound vary in time and it is this variation that gives a characteristic fingerprint for classification. Sound recognition models divide the sound feature space into a number of states and each state is defined by a continuous probability distribution. The states are labeled 1 to \( k \), and sound is indexed by the most probable sequence of states for a given sound model. Figure 19.5 shows four states in two dimensions and a sound trajectory in the space. The dimensions correspond to the basis vectors contained in \( \text{AudioSpectrumBasis} \) and the trajectory corresponds to a sequence of spectral frames projected into an \( \text{AudioSpectrumProjection} \) descriptor.

The \( \text{SoundModel} \) DS is derived from the \( \text{ContinuousHiddenMarkovModel} \) DS defined in MDS. In addition to Markov model parameters, the \( \text{SoundModel} \) DS stores \( \text{AudioSpectrumBasis} \) functions that define the dimensions of the probability space, see the discussion of \( \text{AudioSpectrumBasis} \) above. The multidimensional Gaussian distribution is used for defining state densities. Gaussian distributions are parameterized by a \( 1 \times n \) vector of means, \( \mathbf{m} \), and an \( n \times n \) covariance matrix, \( \mathbf{K} \), where \( n \) is the number of features (columns) in the observation vectors. The \( \text{GaussianDistribution} \) DS stores the inverse covariance matrix and the determinant of the covariance matrix along with the vector of means for each state. Therefore, the expression for calculating probabilities for a random vector, \( \mathbf{x} \), given a mean vector, \( \mathbf{m} \), covariance matrix inverse and the covariance matrix determinant is:

\[
f_x(x) = \frac{1}{(2\pi)^{n/2}|K|^{1/2}} \exp \left[ -\frac{1}{2} (x - m)^T K^{-1} (x - m) \right]
\]
The dynamic behavior of a sound model through the state space is described by a $k \times k$ transition matrix defining the probability of transition to each of the states from any current state, including the probability of self-transition. For a transition matrix, $T$, the $i$th row and $j$th column entry is the probability of a transition to state $j$ at time $t$ given state $i$ at time $t - 1$. The transition matrix is a stochastic matrix that constrains the entries in each row to sum to 1. An initial state distribution, which is a $1 \times k$ vector of starting probabilities that also sum to 1, is required to complete the model. The $k$th element in the vector is the probability of being in state $k$ in the first observation frame. The following example code shows an instance of a `SoundModel` for the `Trumpet` class that is contained in the `SoundClassificationModel` code example shown above. Floating-point numbers have been rounded to three decimal places for illustrative purposes.

```xml
<SoundModel id="IDInstrument:Trumpet">
  <SoundClassLabel>
    <Term id="ID16">Instrument:Trumpet</Term>
  </SoundClassLabel>
  <Transitions dim="6 6">
    1.000 0.000 0.000 0.000 0.000 0.000
    0.000 0.994 0.000 0.000 0.000 0.006
    0.000 0.000 0.993 0.007 0.000 0.000
    0.014 0.000 0.095 0.818 0.000 0.074
    0.000 0.000 0.000 0.005 0.995 0.000
    0.056 0.000 0.000 0.000 0.000 0.944
  </Transitions>
</SoundModel>
</DescriptorModel>
<Descriptor xsi:type="mpeg7:AudioSpectrumProjectionType"/>
<Field xsi:type="mpeg7:SeriesOfVector"/>
</State>
</Label>
  <Term id="IDState1">State1</Term>
</Label>
  <ObservationDistribution xsi:type="mpeg7:GaussianDistributionType">
```

Figure 19.5  Four probability model states in a two-dimensional vector space. Darker regions have higher probabilities. A sound trajectory is shown by the line. The state parameters are chosen to maximize the probability of the states given a set of training data.
19.4 TRAINING A HIDDEN MARKOV MODEL (HMM)

Figure 19.6 shows the flow diagram for a HMM. The states are called hidden because the state sequence is not known but the data generated by the states are known. The observable data must be used to infer the position and extent of the states in the feature space, as well as the parameters for the transitions and initial state distribution. Once these parameters are known, a HMM can be used to convert a sequence of feature vectors into an optimal sequence of states.

Models are acquired by statistical analysis of training data – this process is called model inference. The first, and most important, step in training a HMM is collecting a representative set of exemplars for a given sound class; for instance, violins. For the best results, a number of violin example sequences should be collected that are recorded in differing environments and at many pitches. Ideally, the training set should contain all of the variation for the sound class. A set of 100 sounds is a good guide for training a sound class. However, some sound types exhibit more variation, such as speech, and therefore need more training examples to create a successful model.
States

s1

s2

s3

s4

Observations

o1

o2

o3

o4

Time

Figure 19.6  HMM system showing a sequence of states generating a sequence of observations. HMM inference performs the inverse mapping from training data to state parameters.

During training, all of the parameters for a HMM must be estimated from the feature vectors of the training set. Specifically, these parameters are the numbers of states, the mean vector and covariance matrix for each of the states, the initial state distribution and the state transition matrix. Conventionally, these parameters are obtained using the well-known Baum – Welch algorithm. The procedure starts with random initial values for all of the parameters and optimizes the parameters by iterative reestimation. Each iteration runs over the entire set of training data in a process that is repeated until the model converges to satisfactory values. For a detailed introduction to designing and training HMMs [5].

19.5 INDEXING AND SIMILARITY USING MODEL STATES

Indexing a sound consists of selecting the best-fit HMM in a classifier and generating the optimal state sequence, or path, for that model. The state path is an important method of description since it describes the evolution of a sound through time using a very compact representation; specifically, a sequence of integer state indices. Figure 19.7 shows the state path for a dog barking; there are clearly delimited onset, sustain and termination or silent states. In general, states can be inspected via the state-path representation and a useful semantic interpretation can often be inferred for a given sound class.

Dynamic time warping (DTW) and histogram sum-of-squared differences are two methods for computing the similarity between state paths generated by a HMM. DTW uses linear programming to give a distance between two functions in terms of the cost of warping one onto the other. We may apply DTW to the state paths of two sounds in order to estimate the similarity of their temporal structure [5].

However, there are many cases in which the temporal evolution is not as important as the relative balance of occupied states between sounds. This is true, for example, with sound textures such as rain, crowd noise and music recordings. For these cases it is preferable to use a temporally agnostic similarity metric such as the sum-of-squared differences between state-path histograms. The SoundModelStatePath descriptor provides a container for these histograms. Frequencies are normalized counts in the range 0 to 1.
obtained by dividing the counts for each state by the total number of samples in the state sequence:

$$
\text{hist}_a(j) = \frac{N(j)}{\sum_{i=1}^{K} N(i)}, \quad 1 \leq j \leq K
$$

where $K$ is the number of states, $N(j)$ is the count (frequency) for state $j$ for the given audio segment. Similarity is computed as the absolute difference in the relative frequency of each state between different sound instances. The differences are summed to give an overall distance metric. For two sound models $a$ and $b$, the distance is defined as:

$$
\delta(a, b) = \sum_{j=1}^{k} \sqrt{[\text{hist}_a(j) - \text{hist}_b(j)]^2}
$$

This similarity method applies for all classes of sound and therefore constitutes a generalized sound similarity framework [6, 7]. The following section gives examples of these methods applied to automatic sound classification and QBE searches.
19.6 SOUND MODEL APPLICATIONS

19.6.1 Automatic Audio Classification

Automatic audio classification finds the best-match class for an input sound by presenting it to a number of HMMs and selecting the model with the highest likelihood score. A combination of HMMs used in this way is called a classifier. To build a classifier, a set of individual SoundModel DSs are trained, one for each class in a classification scheme, and combined into a SoundClassificationModel. Given a query sound, the spectrum envelope is extracted and the result is presented to each sound model. The spectrum is projected against the model’s basis functions producing a low-dimensional feature representation represented by the AudioSpectrumProjection descriptor. The Viterbi algorithm is then used to compute the SoundModelStatePath and likelihood score and the HMM with the maximum likelihood score is selected as the representative class for the sound. The algorithm also generates the optimal state path for each model given the input sound. The state path corresponding to the maximum likelihood model is stored and used as an index for query applications. Figure 19.8 illustrates the method for automatic audio classification using a set of models organized into a SoundClassificationModel DS. For a detailed description of the Viterbi algorithm see [5].

In the following example, 19 HMMs were trained corresponding to the leaf nodes of the classification scheme shown in Figure 19.4 above. The database consisted of 2,500 sound segments divided into 19 training and testing sets. 70% of the sounds were used for training the HMMs and 30% were used to test the recognition performance. For details on the training techniques see [7–10].

The results of classification on the testing data are shown in Table 19.1. The results indicate good performance for a broad range of sound classes. Of note is the ability of the classifier to discriminate between speech sounds and nonspeech sounds, it can also distinguish between male and female speakers. This speech and nonspeech classifier can be used to increase the performance of automatic speech recognition (ASR) in the context of nonspeech sounds and to generate labels and indexes for all of the classes defined by the classifier.

Figure 19.8 Maximum Likelihood Classification with the SoundClassificationModel DS
Table 19.1 Performance of 19 classifiers trained on 70% and cross-validated on 30% of a database consisting of 2,500 sound clips between 1 second and 30 second duration

<table>
<thead>
<tr>
<th>Model name</th>
<th>% Correct classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] AltoFlute</td>
<td>100.00</td>
</tr>
<tr>
<td>[2] Birds</td>
<td>80.00</td>
</tr>
<tr>
<td>[3] Piano</td>
<td>100.00</td>
</tr>
<tr>
<td>[4] Cellos (Pizz and Bowed)</td>
<td>100.00</td>
</tr>
<tr>
<td>[5] Applause</td>
<td>83.30</td>
</tr>
<tr>
<td>[6] Dog Barks</td>
<td>100.00</td>
</tr>
<tr>
<td>[7] Horn</td>
<td>100.00</td>
</tr>
<tr>
<td>[8] Explosions</td>
<td>100.00</td>
</tr>
<tr>
<td>[9] Footsteps</td>
<td>90.90</td>
</tr>
<tr>
<td>[10] Glass Smashes</td>
<td>92.30</td>
</tr>
<tr>
<td>[11] Guitars</td>
<td>100.00</td>
</tr>
<tr>
<td>[12] Gun shots</td>
<td>92.30</td>
</tr>
<tr>
<td>[13] Shoes (squeaks)</td>
<td>100.00</td>
</tr>
<tr>
<td>[14] Laughter</td>
<td>94.40</td>
</tr>
<tr>
<td>[15] Telephones</td>
<td>66.70</td>
</tr>
<tr>
<td>[16] Trumpets</td>
<td>80.00</td>
</tr>
<tr>
<td>[17] Violins</td>
<td>83.30</td>
</tr>
<tr>
<td>[18] Male Speech</td>
<td>100.00</td>
</tr>
<tr>
<td>[19] Female Speech</td>
<td>97.00</td>
</tr>
</tbody>
</table>

Mean recognition rate 92.646

The second classification example describes an experiment in music genre classification using the feature extraction and training methodology outlined above. Several hours of material were collected from compact discs and MPEG-1 (Layer III) compressed audio files corresponding to eight different musical genres. The data was split into 70/30% training/testing sets and HMMs were trained in the same manner as described above. Each sound file was split into several chunks consisting of a maximum of 30-second segments. Thus, the models were tuned to capture localized structures in the sound data. Table 19.2 shows the results of classification into music genres for the novel testing data. These results indicate that the classification system generalized to recordings consisting of mixed sources.

19.6.2 Audio QBE

In a QBE system, feature extraction is performed on a query sound as shown in Figure 19.8 above. The model class and state path is stored and these results are compared against the state paths stored in a precomputed sound index database using the sum of square differences in relative state frequencies also discussed above. Figure 19.9 shows a screen shot of an application called SoundSpotter that uses the sound classification tools to search a large database by categories and finds the best matches to the selected query sound using state-path histograms. Figure 19.10 shows the query and best-match state-path histogram, as represented by the SoundModelStateHistogram DS. The frequency
Table 19.2  Performance of eight classifiers using a 70/30% training/testing split for music genre classification

<table>
<thead>
<tr>
<th>Model name</th>
<th>% Correct classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] Bluegrass</td>
<td>96.8</td>
</tr>
<tr>
<td>[2] Reggae</td>
<td>92.5</td>
</tr>
<tr>
<td>[3] Rap</td>
<td>100.0</td>
</tr>
<tr>
<td>[4] Folk</td>
<td>92.3</td>
</tr>
<tr>
<td>[5] Blues</td>
<td>98.7</td>
</tr>
<tr>
<td>[6] Country</td>
<td>88.9</td>
</tr>
<tr>
<td>[8] NewAge</td>
<td>98.3</td>
</tr>
</tbody>
</table>

Mean recognition rate 95.4%

Figure 19.9  The SoundSpotter application finds the best matches to a highlighted query in a selected class. The 10 best matches are returned by the application on the basis of the distance between query and target state-path histograms. The categories on the left of the figure were automatically assigned by HMM classification values are normalized to sum to 1 for each histogram, thus, sound segments of different lengths can be compared using these methods.

The examples given above organize similarity according to a taxonomy of categories. However, if a noncategorical interpretation of similarity is required a single HMM can be trained with many states using a wide variety of sounds. Similarity may then
proceed without category constraints by comparing state-path histograms in the large generalized HMM state space.

19.7 SUMMARY

This chapter outlined the LLD and high-level DSs for automatically classifying and querying sound content. The descriptors consist of low-dimensional representations of audio spectra that are extracted using linear basis methods. The high-level tools are based on classification schemes and HMM classifiers, both defined in MDS, and they are used to generate category labels for audio content and to index the content by HMM state indices.

One of the major design criteria for the tools was to represent a wide range of acoustic sources including textures and mixtures of sound. As such, the tools presented herein exhibit good performance on musical sound recordings as well as traditionally nonmusical sources such as vocal utterances, animal sounds, environmental sounds and sound effects. Use of the tools for automatic classification and QBE was discussed as well as extensions to generalized audio similarity.

In conclusion, the descriptors and DSs outlined in this paper provide a consistent framework for analyzing, indexing and querying diverse sound types. The ability to automatically classify sounds, and the ability to search for sounds-like candidates in a large
database, independent of the source type, will be valuable components in new Internet music software, professional sound-design software, composers tools, audio-video search engines and many yet-to-be-discovered applications.

REFERENCES


