# MPEG-7 Sound Recognition Tools

Michael Casey, *member* IEEE

***Abstract*— The MPEG-7 sound recognition descriptors and description schemes consist of tools for indexing audio media using probabilistic sound models. The descriptors provide containers for category labels as well as data structures for quantitative information about sound content. We describe the normative tools as well as informative methods for automatic description extraction and sound matching.**

***Index terms*—Sound recognition, taxonomy, pattern matching, spectral features, dimension reduction, finite state models.**

## I. Introduction

Audio media consists of a wide range of sound phenomena including environmental sounds, background noises, Foley sounds, animal sounds, speech sounds, non-speech utterances and music. The sound recognition tools are designed for searching media by automatically indexing a sound track, such as a film, with a description of the sound content. Recent progress in pattern recognition research has made such automatic indexing a plausible technology, [1][2][3][4][5].

The MPEG-7 sound recognition tools provide a unified interface for automatic indexing of audio using trained sound classes in a pattern recognition framework. Description is divided into two types; text-based description by category labels and quantitative description using probabilistic models.

Category descriptors provide qualitative information about the nature of the sound content. Descriptions in this form are suitable for text-based query applications, such as Internet search engines, or any processing tool that uses text fields. The quantitative descriptors consist of compact mathematical information about an audio segment that may be used for numerical evaluation of sound similarity. These latter descriptors are well suited to query-by-example search applications, as we shall discuss.

The normative tools have been designed to enable different extractor implementations with the underlying motivation of compatibility between the different extraction methods. In addition, informative tools have been developed that serve as examples for implementing sound recognition applications using MPEG-7 descriptors and description schemes.

*Author Affiliation:* Mitsubishi Electric Research Labs, Cambridge, MA, USA.

## II. Sound Recognition Descriptors and Description Schemes

### A. Qualitative Descriptors

Amongst the reasons for classifying sound segments is the wish to understand something about the content. For example, a scream in a film soundtrack may indicate danger, whereas laughter may indicate comedy. Structured indexing of sound segments with hierarchical categories gives both an absolute category concept for a sound as well as relative cues about the category. The *SoundCategory* and *ClassificationScheme* description schemes define category labels and relationships between the categories.

#### 1) SoundCategory

This description scheme is used for naming sound categories and it is derived from the *ControlledTerm* description scheme defined in the Multimedia Description Schemes (MDS) part of the standard. As an example, the sound of a dog barking may be given the category label “Dogs” with “Bark” as a sub-category. In addition, “Woof” or “Howl” may be desirable sub-categories of “Dogs”. The first two sub-categories are closely related, but the third is an entirely different sound event.

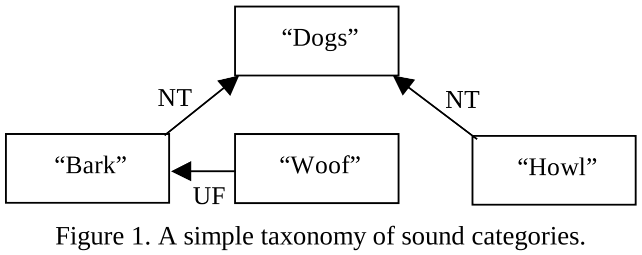


Figure 1. A simple taxonomy of sound categories.

Figure 1 shows four controlled terms that are organized into a taxonomy with “Dogs” as the root node. Each term has at least one relation link to another term in the taxonomy. By default, a contained term is considered a narrower term (NT) than the containing term. However, in this example, “Woof” is defined as being a nearly synonymous with, but less preferable than, “Bark”. To capture such structure, the following relations are available as part of the *ControlledTerm* description scheme:

- *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.

- *US* – **Use instead**. The related term is (nearly) synonymous with the containing term but the related term is preferred to the containing term.
- *UF* – **Use for**. The related term is (nearly) synonymous with the containing term but the containing term is preferred to the 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 following XML-schema code shows how to instantiate the *SoundCategory* description scheme for the category “Dogs” using the MPEG-7 description definition language (DDL):

```
<SoundCategory term=”1” scheme=”DOGS”>
    <Label>Dogs</Label>
    <TermRelation term=”1.1” scheme=”DOGS”>
        <Label>Bark</Label>
        <TermRelation term=”1.2”
scheme=”DOGS” type=”UF”>
            <Label>Woof</Label>
        </TermRelation>
    </TermRelation>
    <TermRelation term=”1.3” scheme=”DOGS”>
        <Label>Howl</Label>
    </TermRelation>
</SoundCategory>
```

The *term* and *scheme* attributes together provide unique identifiers that are used for referencing categories and taxonomies from other description schemes such as probabilistic models. The *Label* descriptor gives a meaningful semantic label for each category and the *TermRelation* descriptor describes relationships amongst terms in the taxonomy.

### 2) *ClassificationScheme*

Categories may be combined into classification schemes to make a richer taxonomy; for example, “Barks” is a sub-category of “Dogs” which is a sub-category of “Pets”; as is the category “Cats”. The following is an example of a simple classification scheme for “Pets” containing two categories: “Dogs” and “Cats”.

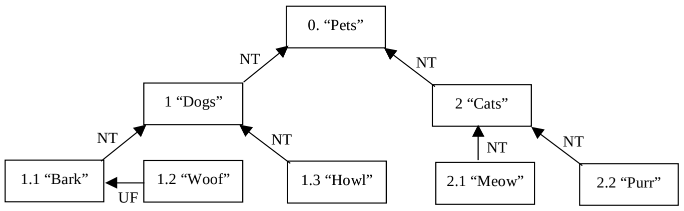


Figure 2. Combining categories into a larger taxonomy.

To implement this classification scheme by extending the previously defined scheme, a second scheme, named “CATS”, is instantiated as follows:

```
<SoundCategory term=”2” scheme=”CATS”>
    <Label>Cats</Label>
    <TermRelation term=”2.1 scheme=”CATS”>
        <Label>Meow</Label>
    </TermRelation>
    <TermRelation term=”2.2” scheme=”CATS”>
        <Label>Purr</Label>
    </TermRelation>
</SoundCategory>
```

Now to combine these categories, a *ClassificationScheme*, called “PETS”, is instantiated that references the previously defined schemes:

```
<ClassificationScheme term=”0”
scheme=”PETS”>
    <Label>Pets</Label>
    <ClassificationSchemeRef scheme=”DOGS”/>
    <ClassificationSchemeRef scheme=”CATS”/>
</ClassificationScheme>
```

Now, the classifications scheme called “PETS” consists of all the category components of “DOGS” and “CATS” with the additional category “Pets” as the root. Whilst building a taxonomy may be sufficient for some indexing applications, it is just the first step toward the goal of automatic sound classification. The following sections outline quantitative descriptors for classification and indexing.

## B. *Quantitative Descriptors*

The sound recognition quantitative descriptors describe features of an audio signal to be used with statistical classifiers. The sound recognition quantitative descriptors can be used for general sound recognition including sound effects and musical instruments. In addition to the suggested descriptors, any other descriptor defined within the audio framework may be used for classification. However, use of the normative extraction techniques is required for reasons of compatibility between different recognition implementations.

### 1) *AudioSpectrumBasis*

Among the most widely used features for sound classification are spectrum-based representations, such as the power spectrum. However, spectrum-derived features are generally incompatible with probability model classifiers due to their high dimensionality. Each spectrum slice is an *n*-dimensional vector, with *n* being the number of spectral channels, that usually consists of between 64 and 1024 channels of data. A logarithmic frequency spectrum, as represented by the audio framework descriptor *AudioSpectrumEnvelope*, helps to reduce the dimensionality to around 32 channels but probability classifiers work best with fewer than 10 dimensions.

The *AudioSpectrumBasis* descriptor is a container for basis functions that are used to project a spectrum onto a lower-dimensional sub-space suitable for probability model classifiers. A basis is computed for each class of sound that captures the statistically most regular features of the sound feature space. Dimension reduction occurs by projection of spectrum vectors against a matrix of data-derived basis functions. The basis functions are stored in the columns of a matrix in which the number of rows corresponds to the length of the spectrum vector and the number of columns corresponds to the number of basis functions.

Basis projection is the matrix product of the spectrum and the basis vectors. Figure 3 shows a spectrogram reconstructed from four basis functions. The vectors on the left are combined with the vectors above the figure, using the vector outer product, each resulting matrix is summed to produce the final reconstruction.

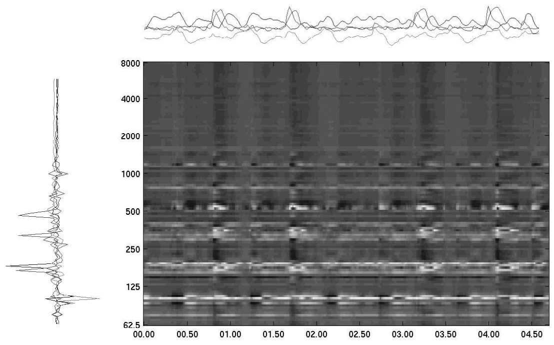


Figure 3. Reconstruction of a spectrogram from 4 basis functions, (pop music).

Basis functions are chosen to maximize the information in fewer dimensions than the original data. For example, basis functions may correspond to uncorrelated features extracted using principal component analysis (PCA) or a Karhunen-Loeve (KL) transform, or statistically independent components extracted by independent component analysis (ICA). For classification purposes a basis is derived for an entire class thus the classification space consists of the most statistically salient components of the class. The following DDL instantiation defines a basis projection matrix that reduces a series of 31-channel logarithmic frequency spectra to 5 dimensions.

```
<AudioSpectrumBasis loEdge="62.5"
hiEdge="8000" resolution="1/4 octave">
<Matrix dim="31 5">
       0.26 -0.05  0.01 -0.70  0.44
       0.34  0.09  0.21 -0.42 -0.05
       0.33  0.15  0.24 -0.05 -0.39
       0.33  0.15  0.24 -0.05 -0.39
       0.27  0.13  0.16  0.24 -0.04
       0.27  0.13  0.16  0.24 -0.04
       0.23  0.13  0.09  0.27  0.24
       0.20  0.13  0.04  0.22  0.40
       0.17  0.11  0.01  0.14  0.37
</Matrix>
</AudioSpectrumBasis>
```

The *loEdge, hiEdge* and *resolution* attributes give the lower and upper frequency bounds of the basis functions and the spacing of the spectral channels in octave-band notation. In the MPEG-7 classification framework, the basis functions for an entire class of sound are stored along with a probability model for the class.

### 2) *SoundRecognitionFeatures*

The features used for sound recognition are collected into a single description scheme that can be used for a variety of different applications. These descriptors have been shown to perform well in classification of many sound types. However, the description scheme also allows for alternative classifier implementations that may use additional descriptors if required.

```
  <complexType
name="SoundRecognitionFeaturesType">
    <complexContent>
      <extension base="mpeg7:AudioDType">
        <sequence>
          <element name="AudioPower"
type="mpeg7:AudioPowerType"/>
          <element
name="AudioSpectrumBasisProjection"
type="mpeg7:SeriesOfVectorType"/>
          <sequence minOccurs="0">
            <element
type="mpeg7:AudioDType"/>
          </sequence>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

The base features are derived from an *AudioSpectrumEnvelope* extraction process. For each spectral slice, the power is computed and instantiated using the *AudioPower* descriptor. The *AudioSpectrumBasisProjection* descriptor is a container for dimension-reduced features that are obtained by projection of a spectrum envelope against a set of basis functions, as described above. Figures 4a and 4b show a 3 basis components and the resulting basis projection for the spectrogram in Figure 5.

In addition to the base descriptors, an unlimited sequence of alternative descriptors may be used to define classifiers that use special properties of a sound class, such as the *harmonic envelope* and *fundamental frequency* features that are used for musical instrument classification. One convenience of dimension reduction is that any descriptor based on a scalable series may be appended to spectral descriptors with the same sampling rate and a suitable basis may be computed for the entire set of extended features in the same manner as a spectrum-only basis.

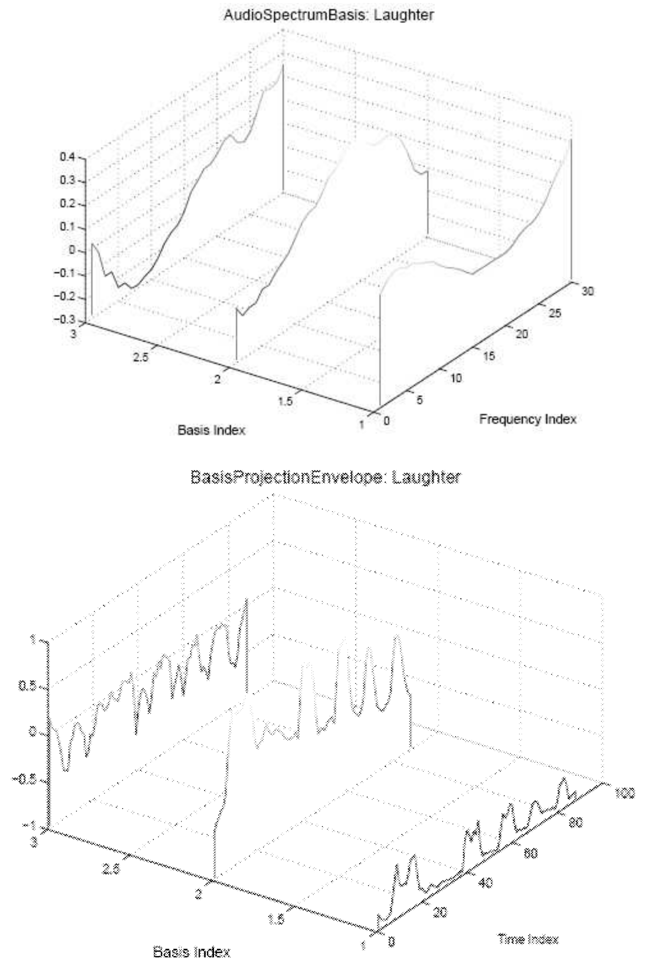


Figure 4. (a) Spectral basis functions for the laughter class, (b) basis projection of a laughter spectrogram.

### 3) *Spectrogram Summarization with a Basis*

An alternative application for the *SoundRecognitionFeatures* desription scheme is efficient spectrogram representation. *AudioSpectrumBasisProjection* and *AudioSpectrumBasis* features may be used as a very efficient storage mechanism for spectrograms for visualization and summarization purposes. In order to reconstruct a spectrogram we use Equation (2) which builds a two-dimensional spectrogram from the cross product of each basis function and its corresponding spectrogram basis projection. Figure 5 shows a spectrogram displayed along with a low-dimensional basis reconstruction. The data storage requirements for each representation are as follows; the log spectrogram has 88 time points with 30 channels thereby requiring 2640 floating point values. The basis functions require three channels of the 88 time points and 3 of the 30-point spectral vectors thus requiring 354 floating point values to reconstruct the spectrogram as shown in the figure, this constitutes data reduction by a factor of approximately 7.5 in this case.

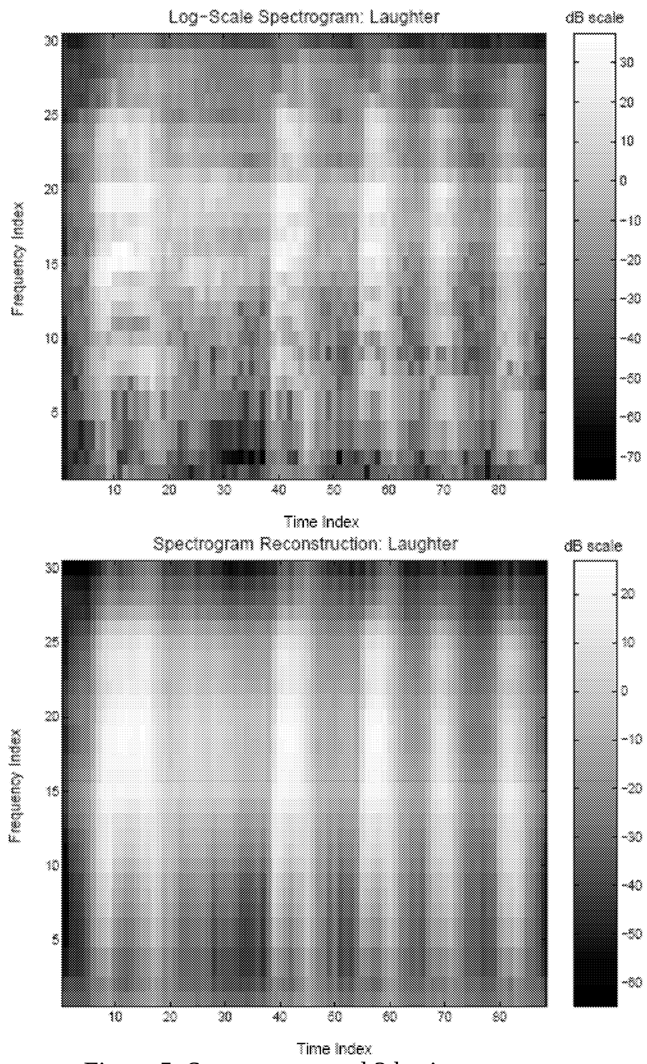


Figure 5. Spectrogram and 3 basis-component reconstruction of laughter.

## C. *Probability Model Description Schemes*

The probability model description schemes are defined in the MDS part of MPEG-7. The motivation for standardization is driven by the cost of designing and training a robust classifier. Applications that use statistical models for classification of content will benefit significantly from a standard interface to probability models. With these standardized schemes in hand, it is possible to share pre-trained probability models between applications even if the specific extraction methods vary. The following sections outline the use of probability model description schemes for sound recognition.

### 1) *FiniteStateModel*

Sound phenomena are dynamic. The spectral features vary in time and it is this variation that gives a sound its characteristic fingerprint for recognition. MPEG-7 sound-recognition models partition a sound class into a finite number of states based on the spectral features; individual sounds are described by their trajectories through this state space. Each state is modeled by a continuous probability distribution such as a Gaussian.

The dynamic behaviour of a sound class through the state space is modeled by a *k* x *k* transition matrix that describes the probability of transition to each of the *k* states in a model given a current state. For a transition matrix, **T**, the *i*th row and *j*th column entry is the probability of transitioning to state *j* at time *t* given state *i* at time *t-1*.

An initial state distribution, which is a *k* x *1* vector of probabilities, is also required for a finite-state model. The *k*th element in the vector is the probability of being in state *k* in the first observation frame.

### 2) *GaussianDistributionType*

The multi-dimensional Gaussian distribution used for modeling states in sound classification. Gaussian distributions are parameterised by a *1 x n* vector of means, **m**, and an *n x n* covariance matrix, **K**, where *n* is the number of features in each observation vector. The expression for computation of probabilities for a random vector, **x**, given the Gaussian parameters is:

$$f_{\mathrm{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}\left|\mathbf{K}\right|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T \mathbf{K}^{-1}(\mathbf{x}-\mathbf{m})\right].$$

### 3) *ContinuousHiddenMarkovModel*

A continuous hidden Markov model is a finite state model with a continuous probability distribution model for the state observation probabilities. The following DDL instantiation is an example of the use of probability model description schemes for representing a continuous hidden Markov model with Gaussian states; in this example floating-point numbers have been rounded to 2 decimal places for display purposes only.

```
<ProbabilityModel
xsi:type="ContinuousMarkovModelType"
numberStates="7">
<Initial dim="7">
0.04  0.34  0.12  0.04  0.34  0.12  0.00
</Initial>
<Transitions dim="7 7">
0.91  0.02  0.00  0.00  0.05  0.01  0.01
0.01  0.99  0.00  0.00  0.00  0.00  0.00
0.01  0.00  0.92  0.01  0.01  0.06  0.00
0.00  0.00  0.00  0.99  0.01  0.00  0.00
0.02  0.00  0.00  0.00  0.97  0.00  0.00
0.00  0.00  0.01  0.00  0.00  0.98  0.01
0.02  0.00  0.00  0.00  0.00  0.02  0.96
</Transitions>
<State><Label>1</Label></State>
<!—State 1 Observation Distribution -- >
<ObservationDistribution
xsi:type="GaussianDistributionType">
<Mean dim="6">
5.11 -9.28 -0.69 -0.79  0.38  0.47
</Mean>
```

```
<Covariance dim="6 6">
1.40 -0.12 -1.53 -0.72  0.09 -1.26
-0.12  0.19  0.02 -0.21  0.23  0.17
-1.53  0.02  2.44  1.41 -0.30  1.69
-0.72 -0.21  1.41  2.27 -0.15  1.05
0.09  0.23 -0.30 -0.15  0.80  0.29
-1.26  0.17  1.69  1.05  0.29  2.24
</Covariance>
<State><Label>2</Label></State>
<!—Remaining states use same structures-- >
<\PobabilityModel>
```

*ProbabilityModel* is declared as an abstract type in the standard and must therefore be instantiated with an xsi:type reference. In this example, *ProbabilityModel* is instantiated as *GaussianDistributionType*, which is derived from the base *ProbabilityModel* class.

## D. *Sound Recognition Model Description Schemes*

The structures outlined so far are isolated tools that have no application structure on their own. The following data types combine these descriptors and description schemes into a unified framework for sound classification and indexing. Sound segments may be indexed with a category label based on the output of a classifier and, additionally, the probability model parameters may be used for indexing a sound in a database. Indexing by model parameters, such as states, is necessary for query-by-example applications in which the query category is unknown or when a narrower match criterion than the scope of a category is required.

### 1) *SoundRecognitionModel*

The *SoundRecognitionModel* description scheme specifies a probability model of a sound class, such as a hidden Markov model or Gaussian mixture model. The following example is an instantiation of a hidden Markov model of the “Barks” sound category. A probability model and associated basis functions for the sound class must be defined in the same manner as the previous examples.

```
<SoundRecognitionModel id="sfx1.1"
SoundCategoryRef="Bark">
<ExtractionInformation term="Parameters"
scheme="ExtractionParameters">
<Label>NumStates=7,
NumBasisComponents=5</Label>
</ExtractionInformation>
<ProbabilityModel
xsi:type="ContinuousMarkovModelType"
numberStates="7">
       ... <!-- see previous example -->
</ProbabilityModel>
<SpectrumBasis loEdge="62.5" hiEdge="8000"
resolution="1/4 octave">
       ... <!-- see previous example -->
</SpectrumBasis>
</SoundRecognitionModel>
```

### 2) *SoundModelStatePath*

This descriptor refers to a finite-state probability model and describes the dynamic state path of a sound through the model. Sounds are indexed by segmentation into model states or by sampling of the state path at regular intervals. In the first case, each audio segment contains a reference to a state, and the duration of the segment indicates the duration of activation for the state. In the second approach the sound is described by a sampled series of indices that reference the model states. Sound categories with long state-durations are efficiently described using the one-segment, one-state approach. Sounds with short state durations are more efficiently described using the sampled series of state indices. Figure 6 shows a spectrogram of a dog-bark sound accompanied by the *SoundModelStatePath* sequence of states for the bark model.

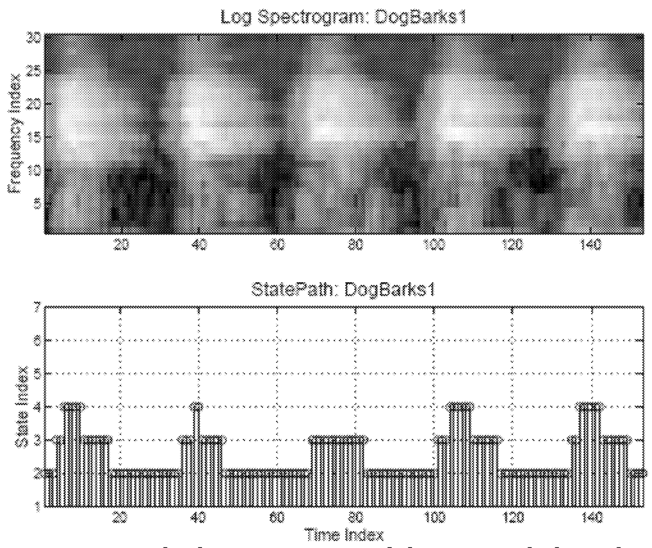


Figure 6. Dog bark spectrogram and the state path through a continuous hidden Markov model.

### 3) *SoundRecognitionClassifier*

This description scheme directly supports sound classification applications by providing a single container for all the necessary components of a classifier. A *SoundRecognitionClassifier* describes relationships between a number of probability models thus defining an ontology of classifiers. For example, a hierarchical recognizer can be described that classifies broad sound classes, such as *animals*, at the root nodes and finer classes, such as *dogs:bark* and *cats:meow*, at the leaf nodes. The scheme defines a map between an ontology of classifiers and a taxonomy of sound categories using the Graph DS structure thereby enabling hierarchical sound models to be used for extracting category descriptions for a given taxonomy.

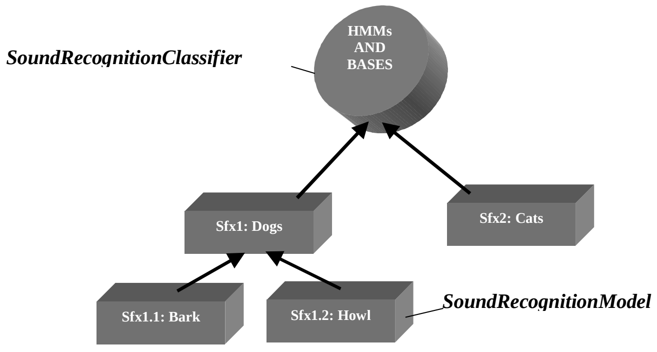


Figure 7. Combining sound recognition models into a classifier ontology.

With this scheme we conclude the discussion on description schemes and turn our attention to extraction tools and applications.

## III. Building a Sound Recognition Classifier

The model building process implemented in the MPEG-7 experimental model (XM) is outlined in the following sections. A stand-alone tool has been developed that performs feature extraction from a directory of WAV format audio files and trains a hidden Markov model with these features. A directory of sound examples is required for each sound class and the hierarchical directory structure defines an ontology that corresponds to a desired taxonomy. One hidden Markov model is trained for each of the directories in the ontology.

### 1) *HMM Model Training*

Much of the effort in designing a classifier is spent collecting and preparing the training data. The range of sounds should reflect the scope of the sound category. For example, dog barks may include individual barks, multiple barks in succession, or many dogs barking at once. The model extraction algorithm adapts to the scope of the data, thus a narrower range of examples produces a more specialized classifier.

### 2) *Audio Feature Extraction*

The system diagram in Figure 8 shows an extraction scheme for *AudioSpectrumBasis* and *SoundRecognitionFeatures*. An audio waveform is windowed into frames and a short-time logarithmic-in-frequency spectrum is extracted. If spectrum extraction used the power spectrum, as standardized by *AudioSpectrumEnvelope*, then at this stage the spectrum is converted from a power spectrum to a decibel-scale amplitude spectrum which is better suited to classification. Each spectrum slice is divided by its power and the resulting normalized spectrum envelope is passed to a basis extraction process.

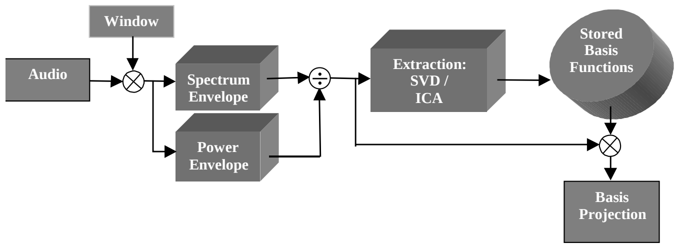


Figure 8. Extraction of low-level audio features for sound recognition classification.

The spectrum basis functions are computed from the normalized spectrum envelope using a basis decomposition algorithm such as the singular value decomposition (SVD), which essentially computes a PCA, or independent-component analysis (ICA). Well-known ICA algorithms, such as *INFOMAX*, *JADE or FASTICA*, are suitable for basis extraction, [6][7][8]. The spectrogram matrix is decomposed into a linear combination of basis components of which a small number are retained that correspond to the desired dimensionality of the output features; typically between three and ten is sufficient for recognition.

The specific choice of basis functions is non-normative but it is suggested, for reasons of maximum compatibility between applications, that the basis be chosen to have columns with unit *L2-norm* and that they maximize the information in *k* dimensions with respect to other possible basis functions. Basis functions may be orthogonal, as given by PCA extraction, or non-orthogonal as given by ICA extraction. Basis projection and reconstruction are described by the following analysis-synthesis equations,

$$\mathbf{Y} = \mathbf{XV} \qquad (1)$$

$$\mathbf{X} = \mathbf{YV}^{+} \qquad (2)$$

where $\mathbf{Y}$ is the extracted $m$ x $k$ observation matrix, $\mathbf{X}$ is the $m$ x $n$ spectrum data matrix with spectral vectors organized row wise and $\mathbf{V}$ is a $n$ x $k$ matrix of basis functions arranged in the columns. Equation (1) corresponds to feature extraction and equation (2) corresponds to spectrum reconstruction, where $\mathbf{V}^{+}$ denotes the pseudo inverse of $\mathbf{V}$ for the non-orthogonal case.

### *3) Model Acquisition*

An unsupervised clustering algorithm is used to partition an *n*-dimensional feature space, populated by the reduced-dimension observation vectors, into *k* states. The algorithm determines the optimal number of states for the given data by pruning a transition matrix given an initial guess for *k*, experiments showed that between 5 and 10 states are sufficient for good classifier performance. Training uses a variant of the well-known Baum-Welch algorithm that is extended by use of an entropic prior and a deterministic annealing implementation of the expectation maximization (EM) algorithm. Details on the HMM extraction algorithm can be found in [9] and [10]. Once trained, each sound recognition HMM is saved to disk, along with its basis functions, in DDL format using the *SoundRecognitionModel* description scheme, see Figure 9.

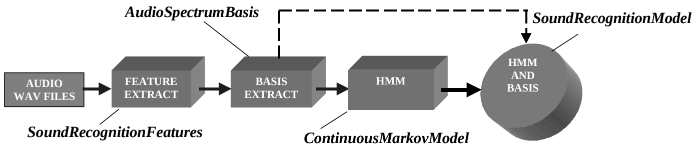


Figure 9. Extraction of hidden Markov model and basis functions and storage in a DDL representation.

When a number of sound models have been trained, corresponding to an entire taxonomy of categories, the HMMs are collected together into the larger *SoundRecognitionClassifier* data structure thereby generating an ontology of models as illustrated in Figure 7.

## IV. SOUND DESCRIPTION

The ontology is used to index new sounds with qualitative and quantitative descriptors. Figure 10 illustrates an automatic extraction system for indexing sound in a database using a pre-trained classifier saved as a DDL file. Sounds are read from a media source format, such as a WAV file, and projected into a separate classification space for each model. They are then passed through a Viterbi decoder that gives both a best-fit model and the state path through the model during the course of the sound. Each sound is then indexed by its category, model reference and the model state path and the descriptors are written to a database in DDL format. The indexed database is ready at this stage for query and search applications using any of the stored descriptors.

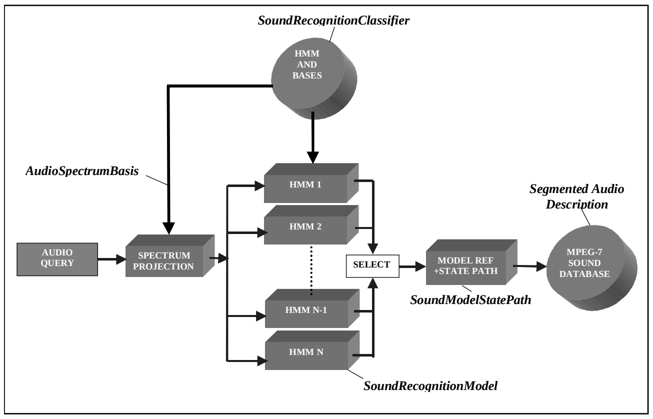


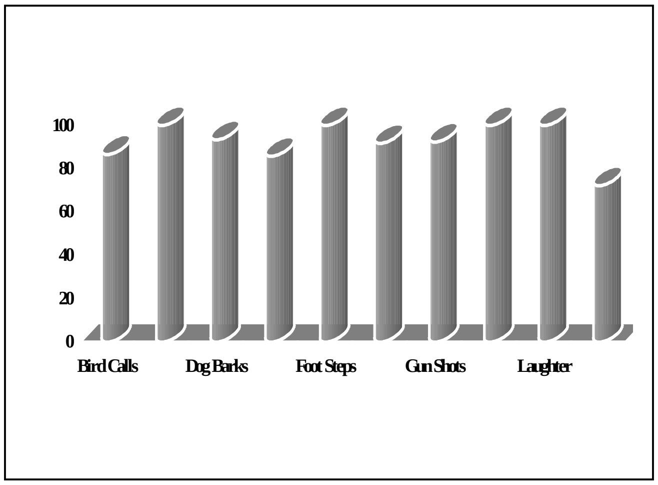
Figure 10. Extraction of sound indexes using a sound-recognition classifier. The model reference and state path is stored.

Figure 11 shows classification performance for 10 sound classes using the extraction scheme outlined above. Performance was measured against a ground truth using the label of the source sound as specified by a professional sound-effect library. The results shown are for novel test data, not used in the training of the classifiers, and therefore demonstrate the generalization capabilities of the classifier.

Figure 11. Performance of a sound effect classifier on a database of 10 sound classes. Performance is measured for novel data.

## V. Example Search Applications

The following sections give examples of how to use the description schemes to perform searches using both DDL-based queries and media source-format queries.



### A. *Query by Example I*

In the first search example, a sound query is presented to the system using the *SoundModelStatePath* description scheme in DDL format. The system reads the query and populates internal data structures with the description information. This description is compared to descriptions taken from a sound database stored on disk, a matching algorithm is applied, and a sorted result list consisting of the closest matches is returned, see Figure 12.

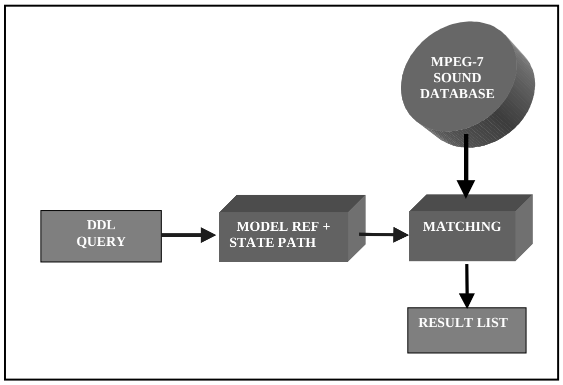


Figure 12. An example search application utilizing a query in DDL format.

The state-path matching algorithm is non-normative, however, a simple example matching algorithm is the sum of square errors between state-path histograms. This matching algorithm requires little computation and can be computed directly from the stored state-path descriptors.

State-path histograms are computed as the total length of time a sound spends in each state divided by the total length of the sound, thus giving a discrete probability density function with the state index as the random variable. The sum-of-squared-errors (SSE) between the query sound histogram and that of each sound in the database is calculated and used as a distance metric. A distance of zero implies an identical match and increased non-zero distances are more dissimilar matches. This distance metric is used to rank the sounds in the database in order of similarity, then the desired number of matches is returned. Figure 13a shows the state path and state path histogram for a Laughter sound query, 13b shows the state paths and histograms for the 5-best matches to the query; all the matches are from the same class as the query which illustrates a successful implementation of the classification system.

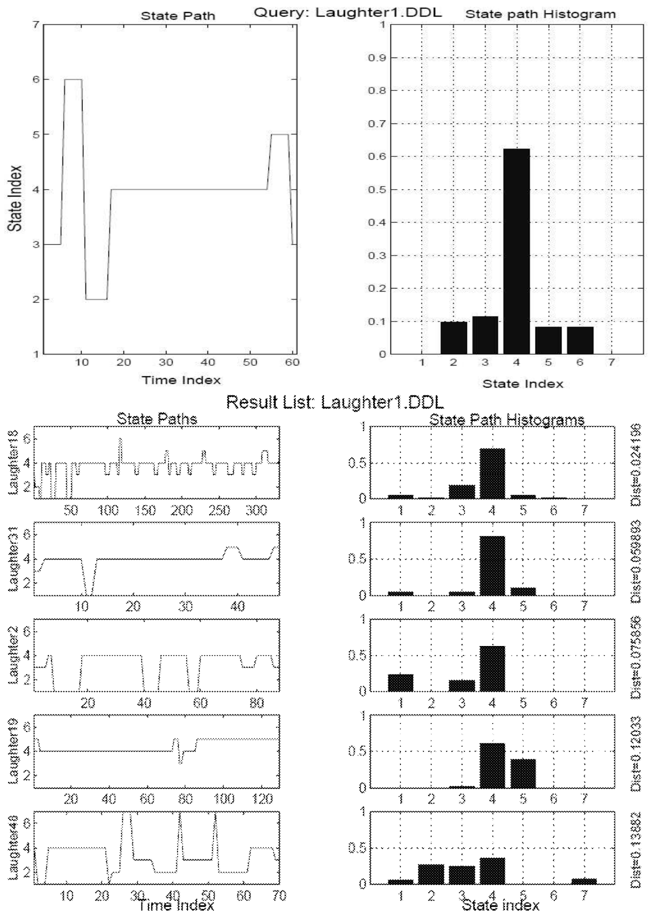


Figure 13 (a) Laughter sound query showing state-path representation and state path histogram. (b) 5-best matches to the Laughter query using the state-path histogram sum-of-squared-errors distance metric.

To leverage the structure of the ontology, sounds within equivalent or narrower categories, as defined by a taxonomy, are returned as matches. Thus the 'Dogs' category will return sounds belonging to all categories related to 'Dogs' in a taxonomy.

### *B. Query-by-Example II*

The system shown in Figure 14 implements a more sophisticated version of the query-by-example application, using an audio query instead of a DDL description-based query. In this case, the audio feature extraction process must be performed, namely spectrogram and envelope computation followed by projection against a stored set of basis functions for each model in the classifier. The resulting dimension-reduced features are passed to a Viterbi decoder for the given classifier and the HMM with the maximum-likelihood score for the given features is selected. The viterbi decoder essentially functions as a model-matching algorithm for the classification scheme. The model reference and state path are recorded and the results are matched against a pre-computed database as in the first example.

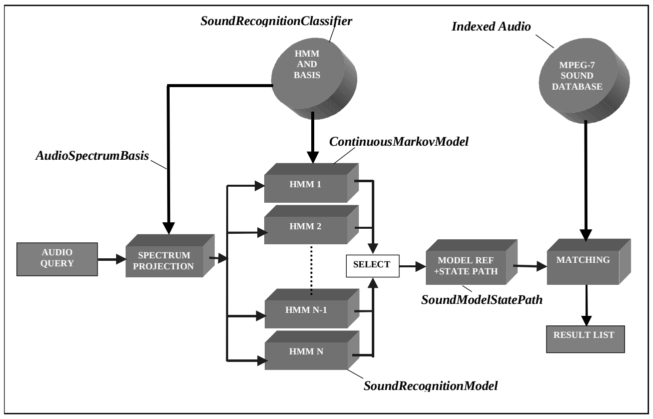


Figure 14. Query-by-example application with a query in media source form. Features must be extracted and projected into the classification space for each model in order to match against the database.

## VI. CONCLUSIONS

This final example illustrates that it is necessary to store basis functions and HMM parameters in addition to the categoy, model-reference and state-path descriptors for a sound-effect database. The *SoundRecognitionClassifier* scheme functions as a header file that is required so extraction processes will produce descriptions that are compatible with descriptions stored in the associated database. The header data enables classification and indexing that is compatible across different extractor implementations and therefore does not require standardization of specific types of classifier.

The description schemes and extractor methodologies outlined in this paper provide a consistent framework for indexing and querying sounds from a wide range of different classes. A number of the multimedia description schemes (MDS) have been discussed in addition to the audio-specific descriptors. Methods for naming categories and defining classification models were outlined as well as example applications with details of specific descriptor matching algorithms.

The tools described in this paper constitute a robust framework for general sound classification that is adaptable to a wide range of applications that have an audio component.

## References

[1] Wold, E., Blum, T., Keislar, D., and Wheaton, J. (1996). Content-based classification, search and retrieval of audio. *IEEE Multimedia,* pp.27-36, Fall.

[2] Martin, K. D. and Kim, Y. E. (1998). Musical instrument identification: a pattern-recognition approach. Presented at the 136th Meeting of the Acoustical Society of America, Norfolk, VA.

[3] Zhang, T. and Kuo, C. (1998). Content-based classification and retrieval of audio. SPIE 43$^{rd}$ Annual Meeting, *Conference on Advanced Signal Processing Algorithms, Architectures and Implementations VIII,* San Diego, CA.

[4] Boreczky, J.S. and Wilcox, L.D. (1998). A hidden Markov model framework for video segmentation using audio and image features, in *Proceedings of ICASSP'98,* pp.3741-3744, Seattle, WA.

[5] Casey, M.A., and Westner, A. (2000). Separation of mixed audio sources by independent subspace analysis. *Proceedings of the International Computer Music Conference,* ICMA, Berlin.

[6] Bell, A. J. and Sejnowski, T.J. (1995) An information-maximization approach to blind separation and blind deconvolution. *Neural Computation,* 7:1129-1159.

[7] Cardoso, J.F. and Laheld, B.H. (1996). Equivariant adaptive source separation. *IEEE Trans. On Signal Processing,* 4:112-114.

[8] Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. On Neural Networks,* 10(3):626-634.

[9] Brand, M. (1999). Pattern discovery via entropy minimization. In *Proceedings, Uncertainty'99.* Society of Artificial intelligence and Statistics #7. Morgan Kaufmann.

[10] Brand, M. (1999). Structure discovery in conditional probability models via an entropic prior and parameter extinction. *Neural Computation.*