Do We Need Automatic Indexing of Musical Instruments?

Alicja Wieczorkowska[2] and Zbigniew W. Raś[1,3]

[1] University of North Carolina, Department of Computer Science,
Charlotte, N.C. 28223, USA
[2] Polish-Japanese Institute of Information Technology
ul. Koszykowa 86, 02-008 Warsaw, Poland
[3] Polish Academy of Sciences, Institute of Computer Science,
ul. Ordona 21, 01-237 Warsaw, Poland
e-mail: awieczor@uncc.edu, ras@uncc.edu

Abstract. Increasing growth and popularity of multimedia resources available on the Web brought the need to provide new, more advanced tools needed for their search. However, searching through multimedia data is highly non-trivial task that requires content-based indexing of the data. Our research is focused on automatic extraction of information about the sound timbre, and indexing sound data with information about musical instrument(s) playing in a given segment. Our goal is to perform automatic classification of musical instrument sound from real recordings for broad range of sounds, independently on the fundamental frequency of the sound.

1 Sound Data

Automatic sound indexing should allow labelling sound segments with instruments names. Knowledge discovery techniques can be used here for that purpose. First of all, we discover rules that recognize various musical instruments. Next, we can apply these rules, one by one, to unknown sounds. By identifying so called supporting rules, we can point out which instrument is playing or is dominating in a given audio segment, and in what time instants this instrument starts and ends playing.

Generally, identification of musical information can be performed for the following data:

- For audio samples taken from real recordings, representing waveform, and
- For MIDI (Musical Instrument Digital Interface) data.

When we deal with the MIDI files, we have access to highly structured data. We are given information about the pitch (fundamental frequency), effects applied, beginning and end of each note, voices (timbres) used, and about every note that is present in a given time moment. Therefore, the research on MIDI

data may concentrate on higher level of musical structure, like key or metrical information.

In the case of recordings, we are dealing with, for each channel we only have access to one-dimensional data, i.e. to single sample representing amplitude of the sound. Any basic information like pitch (or pitches, if there are more than one sounds), timbre, beginning and end of the sound must be extracted via digital signal processing. There exist many methods of pitch extraction, mostly coming from speech processing. But even extraction of such simple information may produce errors and poses some difficulties. Even for singular sound, especially octave errors are common, and various errors for border frames, where 2 consequent sound of different pitch are analyzed. Pitch extraction for layered sounds is even more difficult, especially when spectra overlap. Basically, parameters of fundamental frequency trackers are usually adjusted to characteristics of the instrument that is to be tracked, but this cannot be done when we do not know what instrument is playing.

Identification of musical timbre is even more difficult. Timbre is rather subjective quality, defined by ANSI as the attribute of auditory sensation, in terms of which a listener can judge that two sounds, similarly presented and having the same loudness and pitch, are different. Such definition is subjective and not of much use for automatic sound timbre classification. Therefore, musical sounds must be very carefully parameterized to allow automatic timbre recognition. We assume that time domain, spectrum, and evolution of sound features must be taken into account.

2 Basic Parameterization of Musical Instrument Sounds and their Classification

Broader research on automatic musical instrument sound classification goes back to last few years. So far, there is no standard parameterization used as a classification basis. The sound descriptors used are based on various methods of analysis of time and spectrum domain, with Fourier Transform for spectral analysis being most common. Also, wavelet analysis gains increasing interest for sound and especially for musical sound analysis and representation. Diversity of sound timbres is also used to facilitate data visualization via sonification, in order to make complex data easier to perceive.

There exist numerous parameterization methods that have been applied to musical instrument sounds so far, see for instance (Brown, 2001), (Kaminskyj, 2000), and (Wieczorkowska, 1999). In our research, we decided to base our parameterization on MPEG-7 standard. This standard provides multimedia content description interface (ISO/IEC JTC1/SC29/WG11, 2003), and if this standard gains popularity, the use of MPEG-7 based representation should increase usability of our work.

MPEG-7 provides a universal mechanism for exchanging descriptors of multimedia data. MPEG-7 shall support at least the description of the following

types of auditory data: digital audio, analogue audio, MIDI files, model-based audio, and production data (Manjunath, Salembier and Sikora, 2002). The subclasses of auditory data covered by this standard include: sound track (natural audio scene), music, speech, atomic sound effects, symbolic audio representation, and mixing information. In MPEG-7, so-called Multimedia Description Schemes provide the mechanisms, by which we can create ontologies (Sowa, 2000), and dictionaries, in order to describe musical genre as a hierarchical taxonomy or identify a musical instrument from a list of controlled terms. Evolution of spectral sound features in time can be observed in MPEG-7 by means of Hidden Markov Models (HMM). Therefore, indexing a sound in this standard consists of selecting the best fit HMM in a classifier and generating the optimal state sequence (path) for that model. The path describes the evolution of a sound through time using a sequence of integer state indices as representation.

Classifiers used so far in the research on musical instrument sound classification include wide variety of methods, and the use of HMM is not obligatory in any way. We use MPEG-7 standard as a starting point only, taking sound descriptors as a basis for further processing and research. Low-level descriptors that we use are defined for easy automatic calculation purposes, and they may serve as a basis (for instance AudioSpectrumBasis descriptor) for extraction of new parameters, better suited to instrument classification purposes. High-level descriptors from this standard cannot be extracted automatically, but using low-level descriptors we can calculate new ones, including linear or logical combinations of lower level parameters. Therefore, we decided to choose low-level MPEG-7 descriptors as a research basis, and then search for the classifier.

3 TV-trees and FS-trees Used for Content Description Representation of Audio Data

(Wieczorkowska & Ras, 2001) used trees similar to telescopic vector trees (TV-trees) to represent content description of audio data. We briefly summarize the notion we refer to as TV-trees and also the notion of FS-trees. We outline the strategy for constructing TV-trees.

Each audio signal is divided into frames of length four times the fundamental period of the sound. Each frame is represented as a vector consisting of K acoustic descriptors. So, any collection of audio signals can be defined as a set of K-dimensional vectors. This set is represented as $(K \times N)$-matrix where N is the number of frames in all audio signals in our DB. If needed, K can be reduced to a smaller number by using Singular Value Decomposition method [14]. After introducing the notion of a distance (Minkowski's distance is the most popular) between K-dimensional vectors and setting up activity threshold values for all K dimensions, we partition our K-dimensional space into disjoint and dense clusters.

To define descriptor a as an active in a cluster, we require that the span of values of a in that cluster has to be below the activity threshold value. For

example, if $\{1, \ldots, 100\}$ is the domain of an attribute and its corresponding activity threshold value is $1/20$, then this attribute is active in a cluster if the distance between values of this attribute for any 2 vectors in that cluster is not greater than 5. Clearly, the activity threshold values are purely subjective and they predefine the notion of a cluster. In spite of the drawback of this subjective definition of a cluster, the freedom to define the domain of an attribute to be active is quite convenient from the application site and welcomed by users. Storage and retrieval of sound files is an example of such an application domain.

Now, we show how to construct TV-tree of order 2 with a goal to represent a set of N points as a collection of clusters associated with leaves of that tree. Initially, the set of N points (initial cluster) is divided into 2 clusters in a such a way that the total number of active dimensions in both clusters is possibly maximized. For each cluster we repeat the same procedure, again maximizing the total number of active dimensions in the corresponding sub-clusters. For instance, if $\{[5, 3, 20, 1, 5]$, $[0, 0, 18, 42, 4]$, $[0, 0, 19, 39, 6]$, $[9, 10, 2, 0, 6]\}$ is the initial cluster, $\{1, \ldots, 100\}$ is the domain of each attribute, and the activity threshold value is $1/20$, then the following two subclusters will be generated: $\{[0, 0, 18, 42, 4]$, $[0, 0, 19, 39, 6]\}$, $\{[5, 3, 20, 1, 5]$, $[9, 10, 2, 0, 6]\}$. The initial cluster has only the last dimension active. After split, the first subcluster has 5 dimensions active and second one has the last two dimensions active. We continue this procedure till all subclusters are relatively dense (all points are close to each other with respect to all dimensions). For instance, in the example above, the first subcluster is dense. The underlying structure for this method is a binary tree with nodes storing information about the center of a corresponding cluster, the smallest radius of a sphere containing this cluster, and its list of active dimensions $(d_1, d_2, ..., d_s)$.

The heuristic procedure to construct a binary TV-tree for a collection of audio signals is similar to the strategy used in *Rosetta* or *See*5 system for discretizing numerical attributes [10], [7]. For m-element domain of an attribute, $m - 1$ splitting points are considered (alternatively, the splitting points can be placed between consecutive dense groups of values of an attribute). Another words, if v_1, v_2 are neighboring values of the attribute a, then an audio signal with value of a less than or equal to $[v_1 + v_2]/2$ is placed in the left subtree and all other audio signals are placed in the right subtree. When this is done, the total number of active dimensions in both left and right subtree is checked. We repeat this step for each attribute and for its all splitting points mentioned above. A split which gives the maximal number of active dimensions, for both left and right sub-tree, is the winning split and it is used to build two children of the current node. The above procedure is recursively repeated at two children nodes just created.

As we have mentioned earlier, each audio signal is divided into frames of length four times the fundamental period of the sound. In practice, an activity usually spans across several contiguous frames. Thus it makes sense to store data in terms of contiguous sound segments of frames. Frame Segment tree (FS-tree)

[14] is a data structure which can be used for compact representation of a sound content.

A frame sequence is a pair $[i, j)$, where $1 \leq i \leq j \leq n$. The interval $[i, j)$ represents the set of all frames between i and j. In other words, $[i, j) = \{k : i \leq k < j\}$. Integer i is the start of the frame sequence $[i, j)$ and j is the end. For example, the frame sequence $[8, 12)$ denotes the set of frames $\{8, 9, 10, 11\}$.

We define a partial ordering $\sqsubseteq$ on the set of all frame sequences as follows: $[i_1, j_1) \sqsubseteq [i_2, j_2)$ iff $i_1 < j_1 \leq i_2 < j_2$. Intuitively, it means that the sequence of frames $[i_1, j_1)$ precedes the sequence of frames $[i_2, j_2)$.

Frame Segment tree (FS-tree) is a binary tree constructed as follows:

- Each node represents a frame sequence $[x, y)$, starting at frame x and including all frames up to, but not including, frame y.
- All leaves are at the same level. The leftmost leaf denotes the interval $[z_1, z_2)$, the second from the left represents the interval $[z_2, z_3)$, the third from the left represents the interval $[z_3, z_4)$, and so on. If N is a node with two children representing the intervals $[p_1, p_2)$, $[p_2, p_3)$, then N represents the interval $[p_1, p_3)$.
- A set of indexes is assigned to each node. Each index is used to denote a single activity or fact associated with the entire frame sequence assigned to that node. Thus, for example, if a node N represents the frame sequence $[i, j)$, and the activity α occurs in all frames in $[i, j)$, then the label α is assigned to node N.

Now, let us assume that QAS is based both on a TV-tree and an FS-tree and user submitting an audio query to QAS is looking for audio files satisfying certain properties expressed in terms of indexes used in FS-trees. User's audio query contains also a sub-query represented by an incomplete K-dimensional vector a of acoustical descriptors which structure is similar to vectors stored in a TV-tree. For this type of queries, FS-tree is searched first for audio segments satisfying desired properties (index-based search). Next, the TV-tree is searched to identify which segments retrieved from FS-tree have also properties expressed by sub-query a or properties close to them.

Starting from the root, TV-tree is searched recursively checking at each node if:

- its active dimensions cover the complete dimensions of vector a,
- all complete dimensions of vector a, which are active at this node, are close to its center (with respect to l).

If the first condition is satisfied, we stop the search. Otherwise the search is continued.
If both conditions are satisfied, each vector from the cluster associated with that node is checked if it is within the corresponding thresholds values and if so, it is returned as the answer to the query.

TV-tree is a structure originating from textual databases. This structure have been adopted for audio data [14] because segments in audio data are built from frames described by descriptors. The match between descriptors based queries and audio segments does not have to be exact to get successful answer.

4 Hierarchical Classification of Musical Instrument Sounds

The MPEG-7 descriptors extracted for consequent analyzing frame are treated as a starting point for further data processing. In order to trace evolution of sound features, we elaborate intermediate descriptors that provide internal representation of sound in our recognition system. These descriptors characterize temporal patterns, specific for particular instruments or instrument groups. The groups may represent instrument family, or articulation (playing technique) applied to the sounds. This is why our system will apply hierarchical classification of musical instrument sounds. The family groups include basically aerophones and chordophones, according to Hornbostel and Sachs classification (Hornbostel and Sachs, 1914). In case of chordophones, we are going to focus on bowed lutes family that includes violin, viola, cello and double bass. The investigated aerophones will include flutes, single reed (clarinet, sax) and double reed (oboe, bassoon) instruments, sometimes called woodwinds, and lip-vibrated, brass instruments, with trumpet, trombone, tuba, and French horn. The articulation applied includes vibrato, pizzicato, and muting.

Hierarchical classification is also one of the means to facilitate correct recognition of musical instrument sounds. Also, obviously classification on the family level yielded better results, as reported in the research performed so far, for instance in (Martin and Kim, 1998) and (Wieczorkowska, 1999). Another argument for hierarchical classification is that for the user, information about the instrument family or articulation may be sufficient. For example, non-expert user may just look for brass-performed theme, or melody played with sweet vibration, delicate pizzicato motif and so on. Not to mention that some of the users simply may not be familiar with sounds of all instruments, and they may not know how the particular instrument sounds like.

Since we deal with real recordings, the audio data may contain various kinds of sounds, including non-pitched percussive sounds, and in further development of the system, also singing or speech. Therefore, our system should start with classification of type of the signal (speech, music, pitched/non-pitched), performing auditory scene analysis and recognition (Peltonen, Tuomi, Klapuri, Huopaniemi and Sorsa 2002), (Rosenthal and Okuno, 1998), for instance (Wyse and Smoliar, 1998). Then, for pitched musical instrument sounds, the system will proceed with further specification, to get as much information as possible from the audio signal.

5 Conclusion

Automatic indexing of multimedia databases is of great importance, and ISO/IEC decided to provide MPEG-7 standard for multimedia content description. However, this standard does not comprise the extraction of descriptors (nor search algorithms). Therefore, there is a need to elaborate extraction of sound descriptors that would be attached to sound files.

In recent years, there has been a tremendous need for the ability to query and process vast quantities of musical data, which are not easy to describe with mere symbols. Automatic content extraction is clearly needed here and it relates to the ability of identifying the segments of audio in which particular instruments are playing. It also relates to the ability of identifying musical pieces representing different types of emotions, which music clearly evokes, or generating human-like expressive performances. Automatic content extraction may relate to many different types of semantic information related to musical pieces. Some information can be stored as metadata provided by experts, but some has to be computed in an automatic way. We believe that our approach based on KDD techniques should advance research on automatic content extraction, not only on identifying the segments of audio in which particular instruments are playing, but also in identifying the segments of audio containing other, more complex semantic information.

References

1. Brown, J. C., Houix, O., McAdams, S., "Feature dependence in the automatic identification of musical woodwind instruments", in *J. Acoust. Soc. of America*, 109, 2001, 1064-1072
2. Hornbostel, E. M. V., Sachs, C., "Systematik der Musikinstrumente. Ein Versuch", in *Zeitschrift fur Ethnologie*, Vol. 46, No. 4-5, 1914, 553-90, available at http://www.uni-bamberg.de/ppp/ethnomusikologie/HS-Systematik/HS-Systematik
3. ISO/IEC JTC1/SC29/WG11, "MPEG-7 Overview (version 9)", Pattaya, March 2003, available at http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm
4. Kaminskyj, I, "Multi-feature Musical Instrument Classifier", MikroPolyphonie 6, 2000 (online journal at http://farben.latrobe.edu.au/)
5. Manjunath, B. S., Salembier, P., Sikora, T. (Eds.), "Introduction to MPEG-7. Multimedia Content Description Interface", J. Wiley & Sons, 2002
6. Martin, K. D. and Kim, Y. E., "Musical instrument identification: a pattern-recognition approach", in *Proceedings of 136th Meeting of the Acoustical Society of America*, Norfolk, VA, October, 1998
7. Øhrn, A., Komorowski, J., Skowron, A., Synak, P. The design and implementation of a knowledge discovery toolkit based on rough sets: The ROSETTA system. In: Polkowski, L., Skowron, A. (Eds.), *Rough Sets in Knowledge Discovery 1: Methodology and Applications*, number 18 in Studies in Fuzziness and Soft Computing, chapter 19, Physica-Verlag, Heidelberg, Germany (1998) 376–399

8. Opolko, F. and Wapnick, J., "MUMS - McGill University Master Samples", CD's, 1987
9. Peltonen, V., Tuomi, J., Klapuri, A., Huopaniemi, J., Sorsa, T., "Computational Auditory Scene Recognition", *International Conference on Acoustics Speech and Signal Processing ICASSP 2002*, Orlando, Florida, May 2002
10. Quinlan, J. R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California (1993)
11. Rosenthal, D., Okuno, H. G., (Eds.) "Computational Auditory Scene Analysis", *Proceedings of the IJCAI-95 Workshop*, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1998.
12. Slezak, D., Synak, P., Wieczorkowska, A., Wroblewski, J., "KDD-based approach to musical instrument sound recognition", *Foundations of Intelligent Systems*, Proceedings of ISMIS'02, Lyon, France, LNCS/LNAI, No. 2366, Springer, 2002, 29-37
13. Sowa, J.F. (2000) Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks/Cole Publishing Co., Pacific Grove, CA.
14. Subrahmanian, V. S.: *Multimedia Database Systems.* Morgan Kaufmann Publishers, San Francisco, CA (1998)
15. Wieczorkowska, A, "The recognition efficiency of musical instrument sounds depending on parameterization and type of a classifier", PhD. thesis (in Polish), Technical University of Gdansk, Poland, 1999
16. Wieczorkowska, A., Ras, Z., "Audio content description in sound databases", in *Web Intelligence: Research and Development*, *Proceedings of WI'01*, Maebashi City, Japan, LNCS/LNAI 2198, Springer-Verlag, 2001, 175-183
17. Wyse L, Smoliar, S. W., "Toward Content-Based Audio Indexing and Retrieval and a New Speaker Discrimination Technique", in Rosenthal, D., Okuno, H. G., (Eds.), *Computational Auditory Scene Analysis*, Proceedings of the IJCAI-95 Workshop, Lawrence Erlbaum Associates, Mahwah, New Jersey, 1998.