SYNTHESIZED POLYPHONIC MUSIC DATABASE WITH VERIFIABLE GROUND TRUTH FOR MULTIPLE F0 ESTIMATION

Chunghsin Yeh
IRCAM / CNRS-STMS
Paris, France
Chunghsin.Yeh@ircam.fr

Niels Bogaards
IRCAM
Paris, France
Niels.Bogaards@ircam.fr

Axel Roebel
IRCAM / CNRS-STMS
Paris, France
Axel.Roebel@ircam.fr

ABSTRACT

To study and to evaluate a multiple F0 estimation algorithm, a polyphonic database with verifiable ground truth is necessary. Real recordings with manual annotation as ground truth are often used for evaluation. However, ambiguities arise during manual annotation, which are often set up by subjective judgements. Therefore, in order to have access to verifiable ground truth, we propose a systematic method for creating a polyphonic music database. Multiple monophonic tracks are rendered from a given MIDI file, in which rendered samples are separated to prevent overlaps and to facilitate automatic annotation. F0s can then be reliably extracted as ground truth, which are stored using SDIF.

1 INTRODUCTION

F0 (fundamental frequency) is an essential descriptor for periodic signals such as speech and music. Multiple F0 estimation aims at extracting the fundamental frequencies of concurrent sources. In the field of MIR (Music Information Retrieval), the multiple F0s serve as low-level acoustic features for building up high-level representation of music notes. In recent years, quite a few multiple F0 estimation algorithms have been developed for music signals but no common database (corpus + ground truth) for evaluation exists. In order to study and to evaluate a multiple F0 estimation algorithm, a polyphonic music database with reliable ground truth is necessary.

There are mainly three types of polyphonic signals used as evaluation corpus: mixtures of monophonic samples, synthesized polyphonic music and real recordings. Mixtures of monophonic samples allow a diversity of combinations among different notes and instruments [1]. The ground truth is extracted by means of single F0 estimation, which can be verified more easily. The concern, however, is that the final mixtures may not have the same statistical properties as those found in real music. To increase the relevance of the test corpus for real world applications, the corpus should take into account musical structures. Synthesized polyphonic music can be rendered from MIDI [2] files by sequencers with sound modules or samplers. Real recordings can be recordings of multitracks or stereo/mono mix-down tracks. Despite the wide availability of music corpora, establishing their ground truth remains an issue. We propose a systematic method to synthesize polyphonic music that allows to use existing single F0 estimation algorithms to establish the ground truth. In addition, the availability and the interchangeability of ground truth data together with the corpus is a main concern for evaluation. Therefore, we plan to distribute this polyphonic database with ground truth available in the SDIF (Sound Description Interchange Format) format.

This paper is organized as follows. In section **2**, we present innovative tools developed within IRCAM's **AudioSculpt** application [3] for manual annotation or score alignment and discuss the issues. In section **3**, we present a systematic method for creating a synthesized polyphonic music database. The method is reproducible and the ground truth is verifiable. The format in which to store the ground truth is described in section **4**. Finally, we discuss the evaluation concerns and the possibility of extending this method for different MIR evaluation tasks.

2 MANUAL ANNOTATION OF REAL RECORDINGS

Nowadays, more and more evaluations use real recordings of mix-down tracks with manually annotated ground truth [4] [5]. The annotation process usually starts with a reference MIDI file and then aligns the note onsets and offsets to the observed spectrogram. Under the assumption that the notes in the reference MIDI file correspond exactly to what have been played in the real performance, the annotation process is in fact "score alignment" with audio signals. At IRCAM, innovative tools in AudioSculpt (see Figure 1) have been developed to facilitate verification and modification of signal analysis and manual annotation.

Given a reference MIDI file and a real recording of the same musical piece, we first align the MIDI notes to the real recording automatically [6]. Then, details like note offs, slow attacks, etc., are manually corrected using AudioSculpt according to the following procedure:

1. Overlay MIDI notes on the spectrogram as a pianoroll like representation. Adjust *MIDI note grid* by tuning for the best reference frequency at note A4.

2. Generate time markers by automatic onset detection [7] and adjust the probability threshold according to the spectrogram.

3. Verify and adjust note onsets detected around transient markers visually and auditively. In addition to the waveform and spectrogram, the *harmonics tool*, *instantaneous spectrum* (synchronous to the navigation bar), etc., provide visual cues for the evolution of harmonic components. The *diapason* allows accurate measurement and sonic synthesis at a specific time-frequency point. *Scrub* provides instantaneous synthesis of a single FFT frame, which allows users to navigate auditively at any speed controlled by hand. Users can also listen to arbitrarily shaped time-frequency zones.

4. Align markers automatically to the verified transient markers using *magnetic snap*.

5. If any inconsistency is found between the MIDI file and the real performance, missing notes can be added and unwanted notes eliminated.

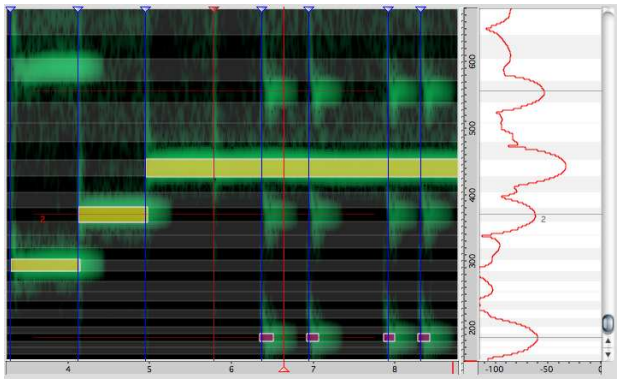

Figure 1. Screenshot of AudioSculpt during annotation showing MIDI notes, MIDI note grids, onset markers, the instantaneous frequency spectrum and the harmonics tool.

Despite all the powerful tools for manual annotation, timing ambiguities need to be resolved based on subjective judgements. Above all, for reverberated recordings, reverberation extends the end of notes and overlaps with the following notes in time and in frequency.

If one aims to find when a musician stops playing a note, a scientific description of reverberation [8] is necessary to identify the end of the playing. Due to reverberation, real recordings of monodic instruments usually appear to be polyphonic, which requires a multiple F0 tracking [9]. To our knowledge, the description of reverberation is not yet available for polyphonic recordings in a reverberant environment.

On the other hand, if one defines the end of a note as the end of its reverberated part, the ambiguity occurs when (1) certain partials are boosted by the room modes and extend longer than the others, and when (2) reverberation tails are overlapped by the following notes and the end of reverberation is not observable.

If manual annotation/alignment is reliably done for non-reverberated recording, it is still disputable at which accuracy one can extract multiple F0s as ground truth. Due to all the issues discussed above, evaluation based on unverifiable reference data endangers the trustworthiness of the reported performance. Therefore, we believe that ground truth should be derived by means of an automatic procedure from the isolated clean notes of the polyphonic music.

3 METHODOLOGY FOR CREATING POLYPHONIC MUSIC DATABASE

To be able to improve the validity of the ground truth, we propose the use of synthesized music rendered from MIDI files. The biggest advantage of synthesized music is that one can have access to every single note from which the ground truth can be established. The argument against synthesized music is often that it is "non-realistic", but few raise doubts about the ground truth. It seems that MIDI note event data is considered as ground truth, which is not true. In fact, MIDI `note off` events are messages requesting the sound modules/samplers to start rendering the end of notes, which usually extends the notes to sound longer after `note off`. Thus, creating the reference data for the rendered audio signal from its original MIDI file is not straightforward. The extended note duration depends on the settings of sound modules or samplers, which is controllable and thus predictable. In order to retain each sound source for reliable analysis as automatic annotation, we present a systematic method to synthesize polyphonic music from MIDI files together with verifiable ground truth.

Given a MIDI file, there are several ways to synthesize a musical piece: mixing monophonic samples according to MIDI `note on` events [10] [11], rendering MIDI files using sequencers with either sound modules, software instruments, or samplers. We choose to render MIDI files with samplers for the following reasons. Firstly, sequencers and samplers (or Sound Bank players) allow us to render MIDI files with real instrument sound samples into more realistic music. Many efforts have been made to provide large collections of musical instrument sound samples such as McGill University Master Samples [1], Iowa Musical Instrument Samples [2], IRCAM Studio On Line [3] and RWC Musical Instrument Sound Database [4]. These sample databases contain a variety of instruments with different playing dynamics and styles for every note in playable frequency ranges, and they are widely used for research. Secondly, there exists an enormous amount of MIDI files available for personal use or research. This is a great potential for expanding the database. Currently,

[1] http://www.music.mcgill.ca/resources/mums/html/index.htm
[2] http://theremin.music.uiowa.edu/MIS.html
[3] http://forumnet.ircam.fr/402.html?&L=1
[4] http://staff.aist.go.jp/m.goto/RWC-MDB/rwc-mdb-i.html

we use the RWC Musical Instrument Sound Database together with the Standard MIDI Files (SMF) of RWC Music Database [12] [13]. There are a total of 3544 samples of 50 instruments in RWC-MDB-I-2001 and 315 high quality MIDI Files in RWC-MDB-C-2001-SMF, RWC-MDB-G-2001-SMF, RWC-MDB-J-2001-SMF, RWC-MDB-P-2001-SMF and RWC-MDB-R-2001-SMF. Finally, we are free to edit MIDI files for evaluation purposes and make different versions from the original MIDI files. For example, limiting the maximal concurrent sources by soloing the designated tracks, changing instrument patches, mixing with or without drums and percussion tracks, etc..

3.1 Musical instrument sound samples

While continuous efforts are being undertaken to manually annotate music scene descriptors for RWC musical pieces [14], no attention is paid to labeling RWC Musical Instrument Sound Database RWC-MDB-I-2001. Each sound file in RWC-MDB-I-2001 is a collection of individual notes across the playing range of the instrument and a mute gap was inserted between adjacent notes. The segmentation should not only separate individual notes but also detect onsets for rendering precise timing of MIDI `note on` events because for certain instrument samples, harmonic sounds are preceded by breathy or noisy regions. If the samples are segmented right after the silence gap, they sometimes lead to noticeable delays when triggered by MIDI events to be played by samplers. These noisy parts in musical instrument sounds come from the fact of the sound generation process. The sources of excitation for musical instruments are mechanical or acoustical vibrators which cause the resonator (instrument body) to vibrate with it. The result of this coupled vibrating system is the setting-up of the **regimes of oscillation** [15]. A regime of oscillation is the state that the coupled vibration system maintains a steady oscillation containing several harmonically related frequency components. It is observed that when instruments are played with lower dynamics (*pp*), it takes much more time to establish the regimes of oscillation. In order to achieve precise onset rendering, we use AudioSculpt for segmenting individual notes. We share the labeled onset markers in SDIF format to facilitate the reproduction of the proposed method described below.

3.2 Creating instrument patches

When receiving MIDI event messages, samplers can render musical instrument samples according to the **keymap**s defined in an **instrument patch**. A sample can be assigned to a group of MIDI notes called a **keyzone**. A set of keyzones is called a keymap, which defines the mapping of individual samples to the MIDI notes at specified velocities. For each MIDI note of a keymap, we assign three samples of the same MIDI note number but different dynamics (often labeled as *ff*, *mf* and *pp*). The mapping of the three dynamics to 128 velocity steps is listed in Table 1. In this way, an instrument patch includes all the samples of a specific playing style, which results in more

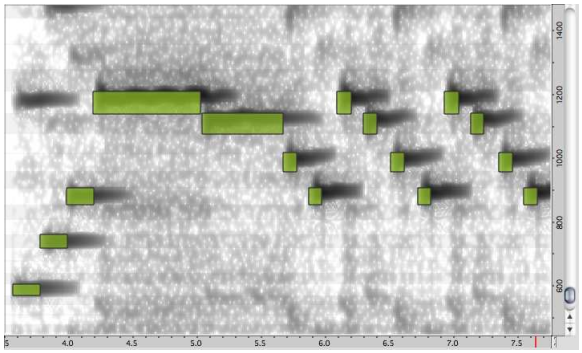

Figure 2. Comparison of MIDI notes with the spectrogram of the rendered audio signal

dynamics in the rendered audio signals. Currently, we focus on the two playing styles: normal and pizzicato.

dynamics	MIDI velocity range
ff	100-127
mf	44-99
pp	0-43

Table 1. Mapping the playing dynamics to the MIDI velocity range

3.3 Rendering MIDI files into multiple monophonic audio tracks

Once instrument patches are created, MIDI files can be rendered into polyphonic music by a sequencer+sampler system. Direct rendering of all the tracks into one audio file would prevent the possibility of estimating the ground truth using single-F0 estimation algorithm. One might then suggest to render each MIDI track separately. However, this is not a proper solution, not only for polyphonic instrument tracks (piano, guitar, etc.) but also for monodic instrument tracks.

To discuss the issues, one example is illustrated in Figure 2. The MIDI notes are extracted from the flute track of *Le Nozze di Figaro* in RWC-MDB-C-2001-SMF. After rendering them by a sequencer+sampler system using RWC flute samples, the spectrogram of the rendered audio signal is shown along with the MIDI notes. Each rectangle represents one MIDI note, with time boundaries defined by `note on` and `note off`, and with frequency boundaries defined by a quarter tone from its center frequency. It is observed that even if the MIDI note events don't overlap, the rendered signals may overlap in time and frequency, depending on the **delta time** between the note events and the **release time** parameter of the instrument patch.

In order to access individual sound sources for verifiable analysis, it is necessary to prevent the overlaps of concurrent notes as well as those of consecutive notes.

Therefore, we propose to split each MIDI track into tracks of separate notes such that the rendered signals don't overlap. Given the release time setting of an instrument patch, concurrent and consecutive notes in a MIDI track can be split into several tracks with the following condition:

$$T_{\text{note on}}(n) \geq T_{\text{note off}}(n-1) + T_{\text{release}} \qquad (1)$$

where $T_{\text{note on}}(n)$ is the `note on` time of the current note, $T_{\text{note off}}(n-1)$ is the `note off` time of the previous note and T_{release} is the release time setting of the instrument patch. In this way, the rendered notes are guaranteed to not overlap one another and we can always refer to individual sound sources whenever necessary. When splitting a MIDI file into several ones, the control messages [5] are retained in the split tracks such that individual notes are exactly the same as those rendered in the polyphonic result (see Figure 3).

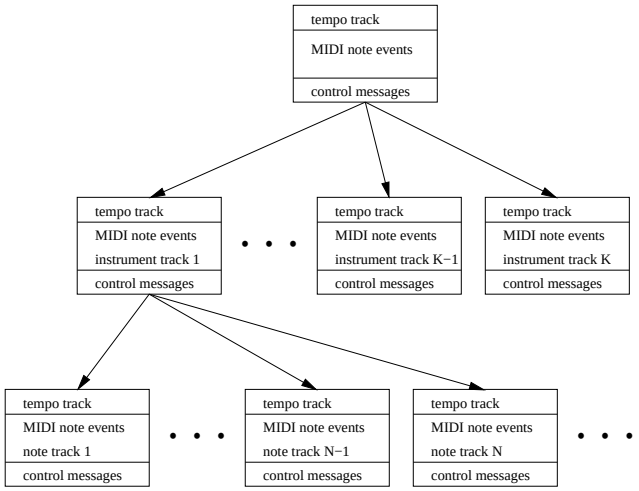

Figure 3. Splitting MIDI files into several containing tracks of separate notes

3.4 Ground truth

Once notes are rendered into non-overlapping samples, we are able to establish the ground truth from the analysis of each rendered sample. The ground truth of fundamental frequencies should be frame dependant. Given the MIDI note number, the reference F0 can be calculated as follows:

$$F_{\text{note}} = \frac{F_{\text{A4}}}{32} \cdot 2^{(\text{MIDI note number}-9)/12} \qquad (2)$$

It is not always correct to calculate F_{note} with a fixed F_{A4} (for example, $440Hz$) because the tuning frequency F_{A4} may differ and moreover, recorded samples may not be played in tune. In Figure 2, MIDI notes are placed at center frequencies calculated with $F_{\text{A4}} = 440$. The D6 note around 1200 Hz either (1) has a higher tuning frequency, or (2) is not played in tune.

[5] Here include channel messages such as pitch bend.

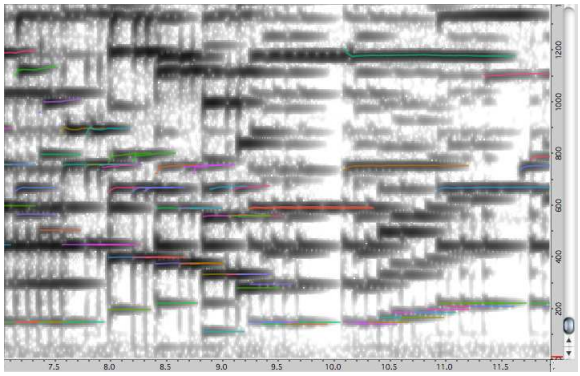

Figure 4. Ground truth of multiple F0 tracks

In order to obtain precise F0s as ground truth, F0 estimation is carried out twice for each sample: a coarse search followed by a fine search. The coarse search uses F_{note} with $F_{\text{A4}} = 440$ for a frequency range $F_{\text{note}} \cdot [0.6\ 1.4]$. Then, limiting the search frequency range to a semi-tone, centered at the energy-weighted average of coarsely estimated F0s. We use the YIN algorithm for F0 estimation because (1) it has been evaluated to be robust for monophonic signals [16] (2) it is available for research use. A window size of $46ms$ is used for analyzing reference F0s. For each F0 track of a sample, only the parts of good periodicity serve as ground truth. The aperiodic parts at the transients and near the end of notes are discarded by thresholding the aperiodicity measure in YIN. Figure 4 shows an example of estimated F0 tracks.

4 STORING MULTIPLE F0S IN SOUND DESCRIPTION INTERCHANGE FORMAT

For verifiable research, analysis data should be stored, well structured, made accessible and be of the highest attainable quality. At the experiment stages, results are often simply dumped into text or binary files using proprietary ad-hoc formats to structure the data. However, this approach has serious drawbacks in the difficulty to extend or to exchange. Ad-hoc formats need some kind of annotation to ensure the contents be interpreted correctly in the future or outside of the specific context in which the file was created. When researchers need to access the data using various tools, as is the case with an evaluation database, a more flexible format is required.

4.1 SDIF vs. XML

A popular format for the storage of MIR data is XML. XML is simple and easily extensible, and well suited to organizing data in a hierarchical way. However, XML also has some serious drawbacks, especially in the case of low-level sound analysis. While XML is a recognized standard format, it can be seen as mainly a syntactic standard, leaving the definition of semantics up to the developers for a specific application. This means that in XML the same

data set can be described in an infinite number of ways, which will all be legal, but not necessarily compatible.

We propose the use of the Sound Description Interchange Format (SDIF) [17], co-developed by IRCAM, CNMAT and MTG-UPF. SDIF addresses a number of specific issues that arise when using XML for the description of sound. One major difference between XML and SDIF is that SDIF stores its data in a binary format, which is much more efficient, both in size and speed, and in a much higher precision than a text oriented format. This high degree of efficiency and accuracy is especially important for low level descriptors.

In addition, SDIF provides standard types for a large number of common aspects of sound, such as fundamental frequency, partials and markers, while treating sound's ever present time property in a special way. As it is flexible to extend SDIF by adding custom types or augmenting existing ones, this will not compromise the compatibility of the files with other programs supporting the standard.

Continuous development since its inception in 1997 has established SDIF as a mature and robust standard, supporting a large range of platforms and programming environments. A recent move to Sourceforge [6] further highlights SDIF as an open source project (LGPL license), ready for integration into open applications as well as commercial ones. Binary and source distributions are available for Windows, Linux, OSX , and bindings exists for C++, Matlab and for Java and scripting languages using SWIG.

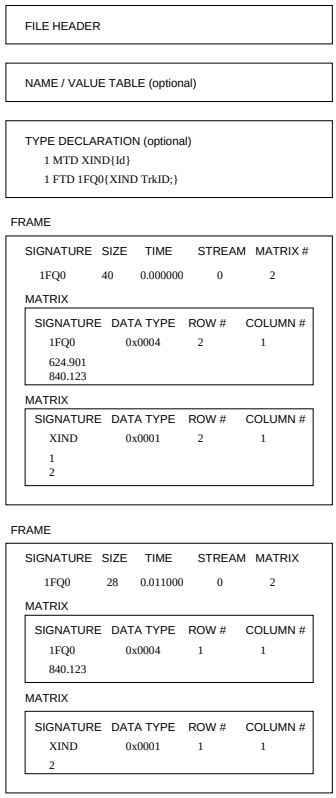

Figure 5. SDIF structure with the text rendition of an example of multiple F0s

4.2 Storing multiple F0s in SDIF

An example of multiple F0s stored as SDIF is shown in Figure 5. Multiple F0s ($624.901Hz$, $840.123Hz$, etc.) are to be stored together with the tracjectory IDs (1, 2, etc.). The SDIF structure is shown along side with the text rendition of the stored data. The optional name/value table is useful for storing metadata, for example the identity of an F0 estimation algorithm, analysis parameters, etc. In the optional type declaration, new types can be declared and standard types extended. In this example, the type "XIND" is declared for the trajectory IDs. After the above header part, data matrices can be stored frame by frame.

5 CONCLUSIONS AND DISCUSSIONS

We have presented a systematic method for the creation of a polyphonic music database. The synthesized music database is reproducible, extensible and interchangeable. Most importantly, the ground truth is verifiable. We propose to use SDIF for storing low-level signal descriptions such as the case for multiple F0s. Information about this database can be found at the author's page [7] .

To evaluate an F0 estimation algorithm, the database should be generalized for all polyphonies and for as many instruments as possible. With the diverse instrument sound samples available for research, we are able to render high quality MIDI files such as RWC SMFs. It is flexible to pre-edit the MIDI files for the evaluation purpose. By means of selecting and editing designated tracks, we can render various polyphonies and instrument combinations. The distribution of instruments programmed in RWC SMFs are shown in Figure 6. Attention should be paid to the non-uniform distribution of the instruments in RWC SMFs. The database should be generalized in a way that not a single instrument is preferred.

Since the extracted F0 tracks may overlap in time and in frequency, evaluation rules must be specified while reporting the evaluation results. One may specify an allowable frequency range within which multiple F0s can be considered as fewer, or even one. Flexible rules for transient frames or weaker energy sources may also be specified.

The proposed method can be extended to other MIR evaluation tasks, such as beat tracking, tempo extraction, drum detection, chord detection, onset detection, score alignment, source separation, etc. Concerning the ground truth, beats and tempos can be easily programmed in MIDI files and chords can be extracted from MIDI files. For evaluation tasks requiring timing precision, the proposed method can provide verifiable analysis as ground truth.

[6] http://sdif.sourceforge.net

[7] http://recherche.ircam.fr/equipes/analyse-synthese/cyeh/database.html

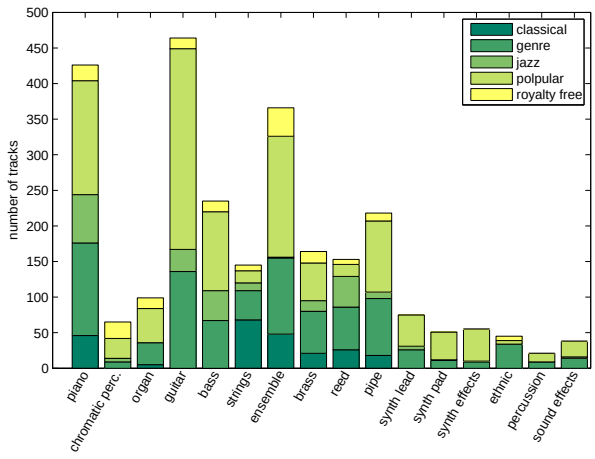

Figure 6. Instrument families used in RWC Standard MIDI Files

6 ACKNOWLEDGEMENT

This work is a part of the project **MusicDiscover**, supported by MRNT (le Ministère délégué à la Recherche et aux Nouvelles Technologies) of France. The author C. Yeh would also like to thank Alain de Cheveigné and Arshia Cont for discussing the issues of creating a polyphonic music database for the evaluation of F0 estimation.

7 REFERENCES

[1] Klapuri, A. "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness", *IEEE Trans. on Speech and Audio Processing*, Vol. 11, No. 6, pp. 804-816, November, 2003.

[2] *Complete MIDI 1.0 Detailed Specifications (Japanese version 98.1)*, Association of Musical Electronics Industry, 1998.

[3] Bogaards, N., Roebel, A. and Rodet, X. "Sound analysis and processing with AudioSculpt 2", *Proc. of Int. Computer Music Conference (ICMC'04)* Miami, Florida, USA, 2004.

[4] Ryynänen, M. and Klapuri, A. "Polyphonic music transcription using note event modeling", *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA'05)*, Mohonk, NY, USA, 2005.

[5] Kameoka, H., Nishimoto, T. and Sagayama, S. "A multipitch analyzer based on harmonic temporal structured clustering", *IEEE Trans. on Audio, Speech and Language Processing*, pp. 982-994, Vol. 15, No. 3, March, 2007.

[6] Rodet, X., Escribe, J. and Durigon, S. "Improving score to audio alignment: percussion alignment and precise onset estimation", *Proc. of Int. Computer Music Conference (ICMC'04)*, pp. 446-449, Miami, Florida, USA, 2004.

[7] Roebel, A. "Onset detection in polyphonic signals by means of transient peak classification", *International Symposium for Music Information Retrieval-MIREX (ISMIR/MIREX'06)*, Victoria, Canada, 2006.

[8] Baskind, A. *Modèles et Méthodes de Description Spatiale de Scènes Sonores*, Université Paris 6, December, 2003.

[9] Yeh, C., Roebel, A. and Rodet, X. "Multiple F0 Tracking in Solo Recordings of Monodic Instruments", *120th AES Convention*, Paris, France, May 20-23, 2006.

[10] Li, Y. and Wang D. L. "Pitch detection in polyphonic music using instrument tone models", *Proc. IEEE, International Conference on Acoustics, Speech and Signal Processing (ICASSP'07)*, pp.II.481-484, Honolulu, HI, USA, April 15-20, 2007.

[11] Kitahara, T., Goto, M., Komatani, K., Ogata, T. and Okuno, H. G. "Instrument Identification in Polyphonic Music: Feature Weighting to Minimize Influence of Sound Overlaps", *EURASIP Journal on Advances in Signal Processing*, Article ID 51979, 2007.

[12] Goto, M., Hashiguchi, H., Nishimura, T. and Oka, R. "RWC Music Database: Popular, Classical, and Jazz Music Databases", *Proc. of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pp. 287-288, Paris, France, 2002.

[13] Goto, M. "RWC Music Database: Music Genre Database and Musical Instrument Sound Database", *Proc. of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pp. 229-230, Baltimore, Maryland, USA, 2003.

[14] Goto, M. "AIST Annotation for the RWC Music Database", *Proc. of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, Victoria, Canada, 2006.

[15] Benade, A. *Fundamentals of Musical Acoustics*. Dover Publications, Inc., New York, 1976.

[16] de Cheveigné, A. and Kawahara, H. "YIN, a fundamental frequency estimator for speech and music", *Journal of the Acoustical Society of America*, Vol. 111, No. 4, pp. 1917-1930, April, 2002.

[17] Schwarz, D. and Wright, M. "Extensions and Applications of the SDIF Sound Description Interchange Format", *Proc. of Int. Computer Music Conference (ICMC'00)*, Berlin, Germany, 2000.