INTERACTIVE MUSIC APPLICATIONS BY MPEG-A SUPPORT IN SONIC VISUALISER

JESUS C. GARCIA [1], COSTANTINO TAGLIALATELA [2], PANOS KUDUMAKIS [3], ISABEL BARBANCHO [1] , LORENZO J. TARDON [1], MARK SANDLER [3]
[1] *Universidad de Málaga , Spain*
jcorral@uma.es, {ibp, lorenzo}@ic.uma.es
[2] *Seconda Universita' Degli Studi Di Napoli, Italy*
c.taglialatela@tiscali.it
[3] *Queen Mary University of London*
{panos.kudumakis,mark.sandler}@eecs.qmul.ac.uk

New interactive music services have emerged, despite currently using proprietary file formats. Having a standardized file format could benefit the interoperability between these services. In this regard, the ISO/IEC Moving Picture Experts Group (MPEG) issued a new standard, the so called, MPEG-A: Interactive Music Application Format (IM AF).
The purpose of this paper is to describe the design and implementation of an IM AF codec and its integration into Sonic Visualiser. In this way, the visualization of the chords or the pitch of the main melody aligned in time with the song's lyrics is achieved. Furthermore, this integration provides the semantic audio research community with a test-bed for further development and comparison of new Sonic Visualiser VAMP plug-ins, e.g., for the conversion of singing voice to text and/or automatic highlighting of lyrics for karaoke applications.

1 INTRODUCTION

The fact that the MP3 file format reigns supreme in the music market is not news. Despite having its market position constantly "attacked" by new formats such as AAC and Ogg Vorbis, nothing has changed so far with respect to user experience [1]. In fact, users continue to play the role of "passive listener" and they are still unable to modify the songs and adapt them to their own taste.
In this context, new interactive music services have emerged with the aim of enhancing the listeners' experience. However, each service uses its own proprietary file format and this has made it difficult to boost a global interactive music market. Thus, a standardized file format is inevitably required to provide the so much needed interoperability between various interactive music players and interactive music songs and albums [2].
This issue is addressed in a recently published standard by ISO/IEC Moving Picture Experts Group (MPEG), known as, MPEG-A: Interactive Music Application Format (IM AF) [3][4]. IM AF allows users to modify the mixing style of a song by changing the volume of each music instrument according to their taste. Other multi-track formats have already been available, e.g., MOGG [5], IEEE 1599 [6] and iXMF [7].
However, IM AF is distinguished by features such as: *timed-text* synchronized with audio tracks, which can represent the lyrics or the chords of a song; *images* related to the song, album and/or artist; *presets* which are predefined versions of a song offered by the music producer (e.g., a karaoke or rhythmic version); and, *rules* constraining the user mixing result to preserve the artistic creation of the composer.
The IM AF specifications come together with a reference software (a non-optimized) player and conformance files. In this way MPEG enables companies to build IM AF compliant encoders and offer associated services and products to market. However, no one has released an open source implementation for an IM AF encoder yet, although commercial services exist [8].
The work described in this paper comes to fill in this gap, proposing an open source implementation of a codec compliant to the IM AF file format [9], with the aim to help the interactive music market blossoming. With respect to the latter, on-line music forums and social networks could be handy: personal mixes of songs could be exported and easily shared between users, having lighter files that contain only information about the mixing parameters while the audio tracks can be made available through various on-line music services. Each audio track could even be replaced by the

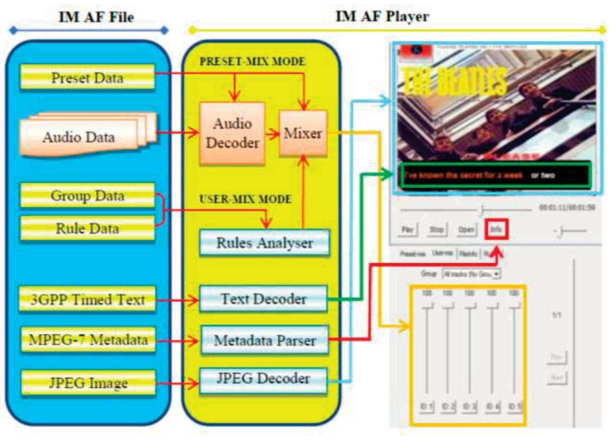

Figure 1. IM AF file format structure and IM AF player architecture with the corresponding media data decoders/parsers.

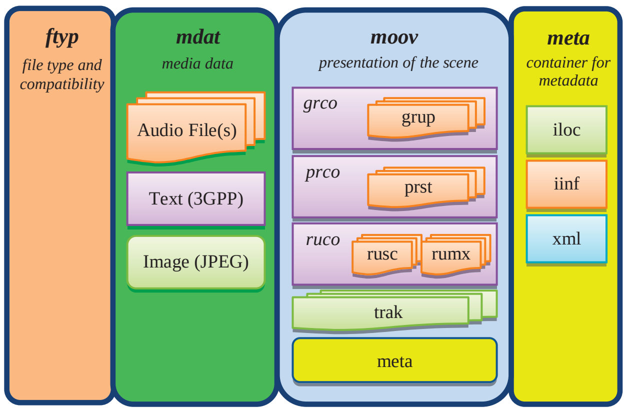

Figure 2. Detailed IM AF file format structure.

users' personal recordings. As a result, it encourages people to develop singing and music instruments playing skills through active learning.

Furthermore, the paper introduces the benefits of Sonic Visualiser with IM AF support. Sonic Visualiser [10] is an open source application designed to assist semantic audio and signal processing researchers to study in a friendly way and take a look at what lies inside an audio file. It has been selected among other music analysis and annotation tools because of its widespread use (>10.000 active users). An essential strength of Sonic Visualiser is its ability to support third-party plug-ins. The native plug-in format is called VAMP, an audio processing plug-in system for plug-ins that extract descriptive information from audio data. In this context, with respect to IM AF, we will make use of VAMP plug-ins specifically designed for audio alignment [11], chords [12] and pitch extraction [13].

In particular, the rest of this paper is structured as follows: the IM AF file format is described in Section 2; in Sections 3 and 4 the implementation aspects of the IM AF player and encoder are discussed, respectively; Section 5 introduces a discussion on how Sonic Visualiser with IM AF support can be used with respect to the aforementioned plug-ins; future developments and conclusions are given in Sections 6 and 7, respectively.

2 FILE STRUCTURE

The framework of the IM AF file is based on the MPEG-4 ISO Based Media File Format (ISO-BMFF) standard [14]; however, IM AF has introduced some improvements to enable interactivity control. The IM AF standard supports compression of the audio tracks in various formats including PCM, MP3, AAC, and SAOC [15]. It also supports the JPEG file format for still pictures [16], the 3GPP timed text for lyrics [17] and the MPEG-7 Multimedia Description Scheme for metadata [17]. Figure 1 shows the architecture of an IM AF interactive music player with the corresponding decoders/parsers for media data.

The IM AF file format consists of a series of boxes that include all relevant data. There are two different types of boxes: general boxes (simply called *Boxes*), which may contain other boxes inside them, and *FullBoxes* which just contain data (no other boxes allowed inside them).

Figure 2 shows how *Boxes* (*ftyp*, *mdat*, *moov*, *meta*) and *FullBoxes* (*grup*, *prst*, *trak*, *etc.*) are located inside an IM AF file [3][4]. All boxes start with a header, which defines their *size* and *type*. The *size* specifies the number of bits that the box will occupy inside the file. The *type* specifies the box identifier (i.e. *ftyp*, *mdat*, *moov and meta)*. Following the same order of boxes as in Figure 2 is not mandatory, however, it is recommended to start with the *File Type Box 'ftyp'*, which identifies the *type* of the file by declaring its *brand compatibility*. The *brand* is related to the maximum number of simultaneously decoded audio tracks and depends on the processing capabilities of the device that will play the file.

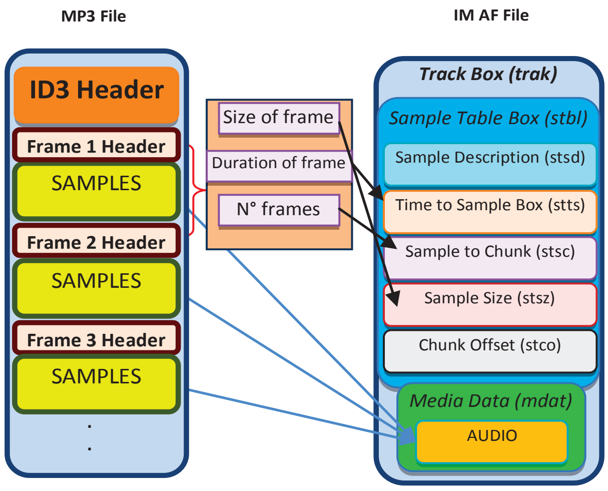

Figure 3. Audio samples and associated data transfer from MP3 to IM AF.

The *Movie Box 'moov'*, describes the scene presentation. This may include one or more *Track Boxes* '*trak*'. Each '*trak*' box contains the description for one type of media (audio or timed-text), while the real media content is stored in the *Media Data Container Box 'mdat'*. This is illustrated in Figure 2. Alternatively, the '*trak*' box may define a URL addressing where the media data are stored. Thus, IM AF files can be very light in terms of storage requirements mainly consisted of metadata.
The *Metadata Box 'meta'* includes metadata, such as, simple background information for a song (title, singer, album, etc.).

3 THE PLAYER

In this section the reference software IM AF player (decoder) and its features are briefly described.

On the player side, users have two different modes of listening a song: *preset-mix mode* and *user-mix mode*. In the *preset-mix mode*, the user could select one of the pre-defined producer mixes. In the *user-mix mode*, the user could select/de-select the tracks and control their volume. All the user's actions are refereed by the *rules analyser* (Figure 1), which decides if the chosen mix is compatible with the rules imposed by the artist (i.e., to avoid complete muting of the guitar track in a rock song).
Independently from the chosen mode, a timed-text track is always available for displaying lyrics, chords or other text information, and so is for pictures in the background.
For legacy players or devices that do not provide multi-track support, IM AF files can still be played. An audio track containing the classic producer's mix of all the instruments in a song can be included in the IM AM file and set as the only "track enabled" that will be played by the legacy player; however, an IM AF player is able to switch it off and play the multi-track version.

4 IMPLEMENTATION OF THE ENCODER

The encoder that creates the IM AF file has been implemented in C programming language. It does not use any external libraries, ensuring that it is cross-platform and will be easily compiled in any computer.
The implementation of a compliant IM AF encoder is mainly based around "filling in" the ISO-BMFF boxes in the right way. This is further explained in the next sections.

4.1 Multi-track Audio

The first step in the encoding process is to extract the audio samples of every single input MP3 audio files and store them into an ISO-BMFF *Media Data Container Box 'mdat'*. This may involve skipping, temporary in this stage, possible MP3 ID3 TAGs containing metadata information related to song, album and artist.
The second step in the encoding process is to extract the following values from each MP3 frame: size (in bytes), duration (in milliseconds) and its number of frames. The IM AF encoder writes this information in the corresponding containers inside a *Track Box 'trak'* as it is illustrated in Figure 3.

The information needed by the encoder in order to reproduce the media data will be contained in tables. Those tables are held in boxes inside the *Sample Table Box* '*stbl*'. The *Sample Description Box* '*stsd*' gives information of the coding type used, the sampling rate and decoder specific information. The *Time to Sample Box* '*stts*' contains a table that stores for each entry two values: the number of consecutive MP3 frames with the same duration and their duration (in milliseconds). Frames are grouped into chunks. The *Sample to Chunk Box* '*stsc*' defines a table that includes the number of frames of each chunk and the number of chunks that have the same number of frames. For constant bitrate MP3s, this table includes only one entry, because all the frames are equally sized. In the *Sample Size Box* '*stsz*' are stored the number of MP3 frames once more and a table giving the size (in bytes) of each of these audio frames. Finally, the *Chunk Offset Box* '*stco*' indicates the position of the first frame of the audio track in the *Media Data Container Box 'mdat'* of the IM AF file.

4.2 3GPP Timed-Text

IM AF supports text visualization in a very customizable way, with possibility to change style and colour for the font and to add visual effects to it. Timed-text data are composed of text samples and sample description. Every sample is a text string (list of characters), which is stored in an external *".txt"* file and can be synchronized with other timed media. Text samples can be followed by sample modifier boxes containing information about how the text string should be rendered such as highlighted text for karaoke. Sample description specifies the way text is rendered, its horizontal and vertical justification, its background and foreground colour, font type and size, etc. The text file contains the strings and their display start/end time. This information is used for dynamic highlighting,(i.e., for lyrics in karaoke style). The syntax is very simple:

[start time] *Text String* **[end time]**

The encoder seeks every text string inside the text file and it copies them in the '*mdat*' box of IM AF file. After a text string is stored, the encoder writes the sample modifier boxes. Values like the highlight colour of the text or the highlight start/end time of words in a phrase are saved at this point. Storage of timing information is similar to audio tracks, as the same boxes are used (*stts*, *stsc*, *stsz* and *stco*, as showed in Figure 3). Duration for every string is saved in '*stts'* box (*sample delta = duration of interval*timescale)*. In '*stsz'* is stored the size of every string, including the modifiers.
Another important box is '*tx3g'*, contained inside *Sample Description Box* ('*stsd'*), which defines sample descriptions for the text track (i.e., font type and size, horizontal and vertical justification, background colour) [17].

4.3 MPEG-7 Metadata

Different 'meta' boxes can be contained at the same time in different levels, as shown in Table 1. Every level hosts textual XML metadata information.

Metadata level	Location
track level	*trak*/*meta* box
song level	*moov*/*meta* box
album level	*meta* box of file

Table 1. Hierarchical levels of metadata in an IM AF.

The encoder receives these data either from the producer or extracts them from the ID3 tags of the input MP3 files, then stores them all in the *XML Container Box* '*xml'*. MPEG-7 is the multimedia content description standard supported by IM AF [17].

4.4 JPEG Still Pictures

Another feature of the IM AF interactive player is that it can display images while a song is played. This is taking place by supporting the JPEG file format for still pictures [16]. Images can be associated, e.g., to music album's cover and/or artist's photos. A picture is included as metadata in the standalone *Metadata Box* '*meta*' in the IM AF file format. Similar to MPEG-7 creation metadata can also be a hierarchy of still pictures at album, song and track level, as shown in Table 1.
From the encoder point of view, the size of the desired JPEG picture is calculated, and then the whole *'.jpeg'* image file is copied in the *Media Data Container Box 'mdat'* box. The image's size and its offset in the *'mdat'* are saved in the *Item Location Box* '*iloc*'; in particular into the *'extent_lenght'* and *'extent_offset'* values. Other information, such as picture's name and encoding type, are stored in *Item Information Box* '*iinf*'.

4.5 Additional boxes

IM AF allows music tracks to be grouped (e.g., all the guitar tracks of a song can consist the *guitar group*). It also provides support for *presets* (e.g., 'karaoke' or 'acoustic' versions of the same song) and *rules* (e.g., not muting completely the guitar solo track in a song). Presets are involved in the player's *preset-mix mode*, while rules are defined to referee user's interaction in *user-mix mode*. Some additional boxes had been introduced to ISO-BMFF structure to define these IM AF's features: *Group Container Box* '*grco*', *Presets Container Box* '*prco*' and *Rules Container Box* '*ruco*'. Each container can host, respectively, one or more *Group Boxes* '*grup*', *Preset Boxes* '*prst*' and *Selection Rule Boxes* '*rusc*' or *Mixing Rule Boxes* '*rumx'*.

Groups

Several audio tracks can be gathered in a group (e.g., all guitars of a song). The groups' container is *Group Container Box 'grco'* box, it hosts one or more *Group Boxes* '*grup*', that is, as many as the groups desired by the producer. This feature allows hierarchical structure of audio tracks in the IM AF file. The elements in a group can be a set of tracks or other groups, allowing several hierarchical levels. Number of tracks per group, name and number of groups are defined by the producer.

Rules

Rules are restrictions imposed by the producer/artist on the remixing level allowed of a song by a user (e.g., may vocals cannot be muted completely). The main goal of these rules is to preserve the artistic creation. The rule's container may host two types of rules: *selection rules* and *mixing rules*.
The four kinds of selection rules are: *exclusion* rule, *not mute* rule, *implication* rule and *min/max* rule. Each one of these essentially defines if a track is in *active* state or not. When an audio track is played its state is *active*, otherwise is *inactive*. The *exclusion rule* specifies which tracks will never be in the active state at the same time. The *not mute rule* defines a track always in the active state. This means that the track will always be played. The *implication rule* specifies that the activation of a track implies the activation of another track. The *min/max* rule is used to set the min/max number of active tracks in groups.
The four kinds of mixing rules are: *limits* rule, *equivalence* rule, *upper* rule and *lower* rule applied on elements' volumes. *Mixing rules* shall be considered only for the elements which are in *active* state at rendering time. The *limits rule* sets the minimum and maximum limits of the volume of each track. *Equivalence* and *upper/lower* rules are binary rules applied for coupling of audio tracks. The *equivalence rule* applied to two different tracks means that these tracks are always played at the same volume. Therefore, if the volume of one of these tracks is changed, the volume of the other track will also be changed by the same level. Volumes affected by *upper/lower rules* shall respect a non-strict superiority/inferiority relationship.

Presets

The IM AF standard defines some default presets. Every preset contains information related to the volume of each track in a song (e.g., a *default preset* could be the original mix by the music producer or a *karaoke preset* with the vocal track muted). The *Preset Container Box* '*prco*' stores one or more *Preset Boxes* '*prst*', that is, as many as the available default preset versions of the mixed-down song.
Presets form two main categories: *static* and *dynamic*. Using *static presets*, the encoder sets a fixed volume to each element involved for the whole duration of the song. In *dynamic presets* the volume of a track (or various tracks, simultaneously) can vary over time [4]. This functionality is useful to create effects like fade-in or fade out.
Every available preset can be equalized using standard filters implemented into the IM AF player, such as: Low and High Pass Filter, Low and High Shelf Filter and/or Peaking Filter. More than one filter can be applied on each track at the same time.

5 RESULTS AND DISCUSSION

Compliance of the developed encoder to the IM AF standard is achieved by creating IM AF files -- similar to conformance files provided by the standard -- which in turn are parsed and played successfully using the reference software player.

As an example, an IM AF created file could have the generic features as those shown in Table 2.

Example1.ima	
Format of tracks	MP3
Number of tracks	6
Groups	1
Presets	2
Rules	2
Still pictures	Yes (album image)
Timed-text track	Yes (lyrics)

Table 2. Generic features of an IM AF created file for testing the encoder compliance to the standard.

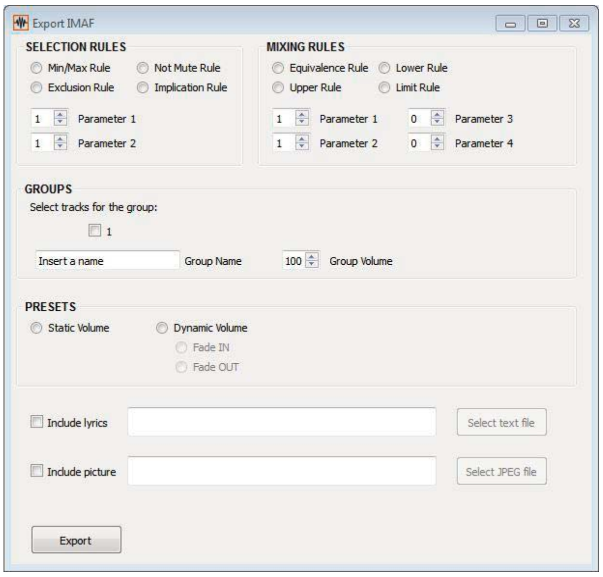

Figure 4. The user interface of the IM AF encoder in Sonic Visualiser.

Combinations of features can be made in creating IM AF files. This example is just a representation of the possibilities of the developed IM AF encoder.

The developed IM AF codec has been fully integrated into Sonic Visualiser. The source code of Sonic Visualiser with IM AF support is available through the SoundSoftware.ac.uk repository [9]. In the latter URL [9], an IM AF (.ima) song is also available for testing and experimenting with the IM AF codec. This song includes several audio tracks, timed text lyrics and the cover of the album. The lyrics have been annotated manually using the Sonic Visualiser itself.

Figure 4 shows the user interface of the IM AF encoder in Sonic Visualiser. This interface has been implemented using the Qt library [19]. Beyond enhancing the user's experience, using IM AF multi-track file format integrated into a tool for the analysis and annotation of audio data, such as Sonic Visualiser, opens up new opportunities for semantic audio researchers.

A possible use case scenario is presented in the following using a Sonic Visualiser plug-in for chords extraction, known as Chordino [12]. Figure 5 shows the difference on using the same Chordino algorithm, with the same set-up, between the single instrument tracks and the classic mix-down of a song. Processing of the mixed track cannot be precise enough for having a reliable chords extraction, due to the overlap of several instruments. A more accurate extraction is achieved by applying the Chordino plug-in to each single instrument track available in an IM AF file (e.g., on the guitar track only, for extracting the guitar chords). Furthermore, the chords extracted are now presented aligned in time with the song's lyrics, as shown in Figure 6. The latter consists the novelty of this paper and becomes feasible thanks to IM AF support in Sonic Visualiser.

The same discussion as the one about chords extraction could be done for the extraction of the melody pitch of a song. Using the VAMP plug-in called Melodia, an algorithm for melody pitch estimation [13], it is possible to have a more efficient pitch extraction from only the vocal track, rather than the mixed song. Furthermore, the melody pitch is now presented aligned in time with the song's lyrics. Extracting the lyrics from the vocal track (singing to text) could be another possible scenario but there is not yet such a VAMP plug-in implemented; however, IM AF support in Sonic Visualiser could encourage such developments since it provides a test-bed for such developments and/or comparisons among different algorithms (e.g., singing voice to text [20], source separation for music instruments extraction from an MP3 [21]).

Another use case scenario of IM AF in Sonic Visualiser involves a VAMP plug-in called MATCH [11], an algorithm for audio alignment between two renditions of the same piece of music. IM AF allows users to add their own musical instrument/vocal track in a song by replacing the existing ones; alignment between the tracks, in such a scenario, is needed for having a proper resulting mix, ensuring that all the instruments start at the same point and follow the same beat. A tool like MATCH can carry out this task easily. However, Sonic Visualiser can run the MATCH plug-in appropriately for calculating automatic alignments between two tracks, it does not do so for multiple audio tracks. The latter is a further demonstration on the opportunities opened up for the semantic audio research community using Sonic Visualiser with IM AF support, as a test-bed not only for the development of new but also for further improvement of existing VAMP plug-ins.

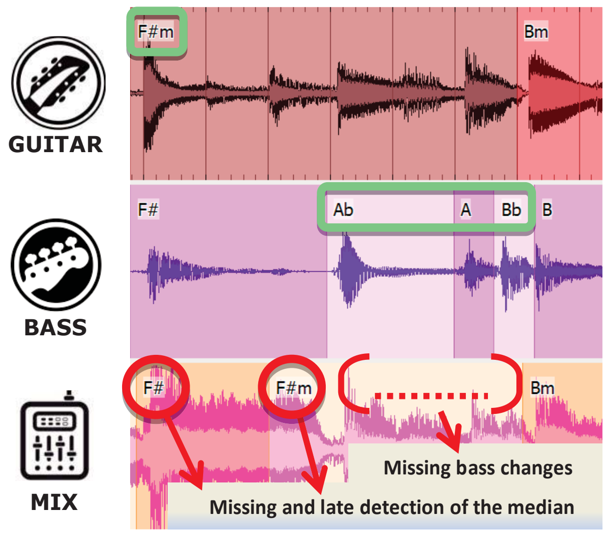

Figure 5. Usage of Chordino in Sonic Visualiser for chords extraction: a) from individual music instruments (guitar and bass) and, b) from the mixed-down track (producer mix/preset).

6 FUTURE DEVELOPMENTS

This innovative file format is expected to enable interoperable commercial interactive music services and collaborative music creation communities to blossom on Internet, due to the flexible IM AF's features inherited from ISO-BMFF. That is, different contents can be stored in different places (e.g., groups, presets and rules are stored into the file; audio tracks, text and pictures are stored on a server linked by URLs), enabling consequently efficient exchange and sharing of IM AF files in social networks. Actually, we are working in this direction by enabling users to record each tracks' volume while disc-jockeying (DJ'ing). In this way users could save their own mixes and share them with their friends using social networks (e.g., Facebook). Furthermore, an alternative cross-platform and cross-browser HTML5 IM AF player [22] has also been developed for users that would only like to listen to their friends' mixes with no need to download Sonic Visualiser.

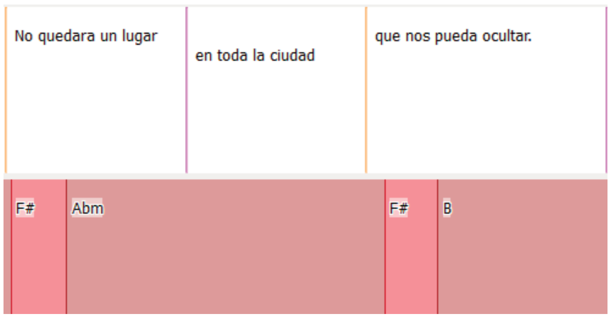

Figure 6. Chords aligned in time with the song's [9] lyrics in Sonic Visualiser.

7 CONCLUSIONS

This paper describes the design and implementation of an IM AF codec and its integration into Sonic Visualiser. This integration provides the semantic audio research community with a test-bed for further development, testing and performance comparisons of newly developed VAMP plug-ins. Some use cases have been described using existing VAMP plug-ins for chord and melody extraction, as well as, music instrument alignment. Based on its features, IM AF has all the potential to be the new music industry file format.

8 ACKNOWLEDGMENTS

Costantino Taglialatela acknowledges that this work has been done during his visit at Queen Mary University of London. Panos Kudumakis acknowledges that this work has been partially done during his visit at the University of Malaga in the context of the program Andalucía TECH: Campus of International Excellence and in conjunction to UK EPSRC project EP/H043101/1 SoundSoftware.ac.uk

9 REFERENCES

[1] P. Kudumakis, "MP3: something's gotta change!", Audio!, pp. 6, Vol. 1, Issue 3, April 2011. Editors: P. Curzon, M. Barthet and S. Dixon.

[2] I. Jang, P. Kudumakis, M. Sandler, K. Kang: "The MPEG Interactive Music Application Format Standard", *IEEE Signal Processing Magazine*, pp. 150-154, Vol. 28, Issue 1, Jan. 2011.

[3] ISO/IEC 23000-12:2010 - Information Technology - Multimedia Application Format (MPEG-A) -- Part 12: *Interactive Music Application Format*.

[4] ISO/IEC 23000-12:2010/Amd.2:2012 - Information technology - Multimedia application format (MPEG-A) - Part 12: *Interactive music application format*, AMENDMENT 2: *Compact representation of dynamic volume change and audio equalization*.

[5] MOGG files, Multitrack Digital Audio Format, (Online), Available: http://moggfiles.wordpress.com (last accessed 01.12.13).

[6] L. A. Ludovico: "Key concepts of the IEEE 1599 Standard", *Proceedings of the IEEE CS Conference The Use of Symbols To Represent Music And Multimedia Objects*, pp. 15-26, 2008.

[7] C. Grigg: "Preview: Interactive XMF - A Standardized Interchange File Format for Advanced Interactive Audio Content", *115th Audio Engineering Society Convention*, October 2003

[8] iKlax Media, (Online), Available: http://www.iklaxmusic.com (last accessed 01.12.13).

[9] Panos Kudumakis, "MPEG developments" (Online), Available: https://code.soundsoftware.ac.uk/projects/mpegdevelopments (last accessed 01.12.13).

[10] C. Cannam, C. Landone and M. Sandler: "Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files", *Proceedings of the ACM Multimedia 2010 International Conference*.

[11] S. Dixon and G. Widmer: "MATCH: a music alignment tool chest", *Proceedings of the International Symposium on Music Information Retrieval*, pp. 492-497, 2005.

[12] M. Mauch and S. Dixon: "Approximate Note Transcription for the Improved Identification of Difficult Chords", *Proceedings of the International Symposium on Music Information Retrieval,* pp. 135-140, 2010.

[13] J. Salamon and E. Gómez: "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics", *IEEE Transactions on Audio, Speech and Language Processing*, 20(6):1759-1770, Aug. 2012.

[14] ISO/IEC 14496-12:2008 - Information technology - Coding of Audio-Visual Objects – Part 12: *ISO base media file format*.

[15] ISO/IEC 23003-2:2010 - Information technology - MPEG audio technologies - Part 2: *Spatial Audio Object Coding (SAOC)*.

[16] ISO/IEC 10918-1:1994 - Information technology - *Digital compression and coding of continuous-tone still images (JPEG).*

[17] ETS 3GPP TS 26.245-2004 - Transparent end-to-end Packet switched Streaming Service (PSS); *Timed text format*.

[18] ISO/IEC 15938-5:2003 - Information technology - Multimedia content description interface – Part 5: *Multimedia description schemes*.

[19] Qt Library, (Online), Available: http://qt.digia.com/ (last accessed 01.12.13).

[20] T. Hosoya, M. Suzuki, A. Ito and S. Makino: "Lyrics recognition from a singing voice based on finite state automation for music information retrieval", *Proceedings of the International Symposium on Music Information Retrieval,* pp. 532-535, 2005.

[21] J. Han, Z. Rafii and B. Pardo, "Audio Source Separation" and "REPEAT", *Research projects of Northwestern University*, Dep. of Elec. Eng. and Comp. Sc., (Online), Available: http://music.cs.northwestern.edu (last accessed 01.12.13).

[22] G. Herrero, P. Kudumakis, L. J. Tardon, I. Barbancho and M. Sandler, "An HTML5 Interactive (MPEG-A IM AF) Music Player", *10th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Marseille, France, 15-18 Oct. 2013.