Geolocation Adaptive Music Player

perez carillo,; THALMANN, FS; fazekas,; sandler,; 2nd Annual Web Audio Conference

Geolocation Adaptive Music Player

Alfonso Perez-Carrillo, Florian Thalmann, György Fazekas, Mark Sandler
Center for Digital Music
Queen Mary University
Mile End Road
London E1 4NS
{a.perezcarrillo, f.thalmann, g.fazekas, mark.sandler}@qmul.ac.uk

ABSTRACT

We present a web-based cross-platform adaptive music player that combines music information retrieval (MIR) and audio processing technologies with the interaction capabilities offered by GPS-equipped mobile devices. The application plays back a list of music tracks, which are linked to geographic paths in a map. The music player has two main enhanced features that adjust to the location of the user, namely, adaptable length of the songs and automatic transitions between tracks. Music tracks are represented as data packages containing audio and metadata (descriptive and behavioral) that builds on the concept of Digital Music Object (DMO). This representation, in line with next-generation web technologies, allows for flexible production and consumption of novel musical experiences. A content provider assembles a data pack with music, descriptive analysis and action parameters that users can experience and control within the restrictions and templates defined by the provider.

1. INTRODUCTION

Technological advances in computer networks, storage, broadband communications and emerging web technologies and standards allow a new generation of web-based applications with unique opportunities in areas of user experience, portability and social collaboration. The capabilities of modern web browsers on desktop and mobile devices allow to accomplish tasks previously reserved to stand-alone applications. Furthermore, mobile devices have developed into compact multi-sensory computers (e.g. multi-touch, accelerometer, gyroscope, video camera, microphone, or geolocation tracking), which provide new interaction opportunities. However, the full potential of such advances is yet to be exploited while also raising the need for novel computational models and algorithms for the analysis and representation of multimodal data and related metadata.

This paper presents a geo-location adaptive music player based on Music Information Retrieval (MIR) methods, audio processing algorithms and the concept of Digital Music Object [3] (DMO) along with the sensing capabilities provided by mobile platforms. The presented work is very related to concepts used in computer game music and have also been applied using mobile sensor data, such as *G1*, a native iOS app produced by RjDj [1].

Music Information Retrieval (MIR) methods are used to extract semantic music features from the audio signal and audio processing algorithms are used to apply transformations to the source music material. A specific type of DMO may be defined as data containers merging sound material, analytical information extracted from the audio, and information about how it should be rendered in realtime. In our specific case, we extend the linear concept of musical track, to a representation of the track as an undirected graph, where the nodes represent the track's beats and the connections indicate the similarity between two beats. Furthermore, we incorporate a set of signal processing methods and a mapping function to associate the tracks to specific locations in a geographic map.

The adaptive player plays back a predefined playlist of DMOs, which are linked with paths in a map. It has two main enhanced features that are controlled by the geo-location of the user, namely, adaptable length of the song and automatic transitions [1, 11]. The player is built as a producer-consumer application, where the content and behavior of the musical objects are prepared by a producer and they are controlled and experienced by consumers. The application was built on Web standards (Web Audio API, HTML5 and Javascript) and cross-platform frameworks (Ionic[2] based on Cordova[3] and AngularJS[4]).

Data representation is discussed in Section 2, audio analysis and semantic representation of tracks as beat similarity graphs is explained in Section 3, rendering algorithms (beat concatenation and automatic transitions) are detailed in Section 4, mapping between tracks and maps in Subsection 5.1 and the rendering behaviour algorithm in Subsection 5.2.

2. DATA REPRESENTATION

A piece of digital music today is typically a signal with some metadata extracted from different sources and processes. An emerging approach to represent musical information and processes is the abstract concept of Digital Music Object (DMO) [3]. It is a musical adaptation of the *Computational Research Object*[2], an amalgamation of a research

[1] https://en.wikipedia.org/wiki/RjDj
[2] http://ionicframework.com
[3] http://cordova.apache.org
[4] https://angularjs.org

Web Audio Conference WAC-2016, April 4–6, 2016, Atlanta, USA

publication, its results, and its methods used to arrive at the results, such as computer code, which enables reproducible scholarship. By capturing the source and the process by which music is composed, produced and comsumed, DMOs provide the means to reconstruct, re-purpose, remix and re-engineer the music [3]. DMOs can be shared, edited and executed by people or machines at any point in the production and distribution process.

Based on the idea of DMO, we extend the traditional concept of musical track to an amalgamation of (a) semantic audio description (Section 3.1) extracted through MIR techniques; (b) signal processing rendering algorithms (Section 4); (c) mapping from geographic location to musical tracks (Section 5.1); and behavioral algorithms (Section 5.2) that adapt the music playback to the movement of the listener.

At present, the data is delivered as a set of files containing the audio signal, the time onsets of the beats and the beat similarity graph. The signal processing algorithms are coded in the application. In the future, we will adopt Semantic Web technologies with standard formats such as the Resource Description Framework [5] or the Web Ontology Language (OWL) [6] and use ontology-based modelling techniques inspired by the Music Ontology [10].

3. SEMANTIC AUDIO DESCRIPTION

Semantic and musically meaningful features are extracted from analysis of audio through Music Information Retrieval techniques. In this work we are mainly focused on the most basic rhythmic unit, the *beat* (Subsection 3.1.1), and in the computation of similarity measures that allow the comparison of different beats.

Measuring similarities in recorded music is a difficult task that depends on a great variety of criteria. It is usually approached by numerically comparing psychoacoustic based descriptions of the music. Relevant features related to psychoacoustic characteristics of the sound (i.e. timbre, melody, loudness and rhythm) are extracted from the audio signal (Section 3.1), including the beat onset positions (Section 3.1.1) and then, these features are averaged across each beat interval into beat-synchronous features. Similarity is represented as beat-synchronous self-disimilarity matrices (Section 3.2) and graphs (Section 3.3).

3.1 Feature Estimation

The estimated features are related to rhythm (i.e. beat position, beat duration, metrical position), pitch (chroma features), timbre (MFCC coefficients) and loudness (Beat Average Energy and Beat Onset Energy). Features were extracted in the spectral domain (except for the energy) and the used parameters were: analysis window length $w_L = 4096$, hop size $hop = w_L/4$ with no zero padding, so the FFT length is the same as w_L. All analyzed tracks had a sampling rate of $sr = 44100$.

3.1.1 Rhythm: beat tracking

Perception of rhythmic structures and patterns in music is due to the succession of strong and weak events. *Onset* detection is the computational tool for finding those events and dividing the music signal into the smallest segments. The *beat* is a periodic pulse, which is perceptually induced from the succession of onsets and is the primary music metrical unit. It is the pulse with which we tap our feet when listening to music. It determines the *tempo*, a measure that is perceptually restricted to a range of around 40 to 260 beats per minute (BPM). Other periodic structures built on top of the beat pulse are the *rhythm* and the *meter*. In this work we are mainly interested in beat onsets, duration and metrical position:

- *Beat onset position.* Usually called *beat tracking* is performed using the dynamic programming method described in [6]. After running the algorithm, special care is taken to place the beats at zero-crossings in order to avoid clicks when concatenating consecutive beats later.
- *Beat duration.* The duration is directly extracted from the interval between beat onsets.
- *Metrical position.* This feature is still a very complex problem and existing algorithms still provide inaccurate results. For this reason, in this work, metrical position is manually indicated by providing the time signature and the first down-beat of each track.

3.1.2 Timbre

Timbre is represented based on the so-called Mel-scale Frequency Coefficients (MFCC), which are a perceptually grounded description of the instantaneous timbre envelope. MFCCs were computed based on D. Ellis Matlab implementation [5]. The algorithm takes the FFT in a frame-by-frame basis. Then its magnitude is warped to a Mel Frequency scale and logarithmic amplitude and the DCT of this log-mel spectrum is computed. The first $n = 12$ coefficients are used.

3.1.3 Pitch

Since it is difficult to compute the pitch in a mixed polyphonic source that may include non-harmonic sounds and noise, a much more relevant representation is the 12 dimensional chroma [7]. A chromagram is a representation against time of the power spectrum energy accumulated into 12 pitch classes that correspond to the *equal temperament* chromatic scale. Since pitches exactly one octave apart are perceived as particularly similar, the distribution of chroma can reveal perceived musical similarity that is not apparent in the original spectra. It is computed using Ellis' *chromagramIF* Matlab implementation [7]. The algorithm uses instantaneous frequency estimates from the spectrogram to keep only real harmonics, which allows to obtain high-resolution chroma profiles.

3.1.4 Signal Eneregy

The subjective judgment of the intensity of a sound, is here expressed numerically as the energy of the signal. It is computed as the quadratic mean or root mean squared (RMS) of the signal amplitude in time domain. Conversely to the intra-beat stability of the previous features (i.e. timbre and pitch), the energy usually shows a strong peak at the beginning of the beat and rapidly decays. For this reason

[5] http://www.w3.org/RDF/

[6] http://www.w3.org/TR/owl2-overview/

[7] http://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/

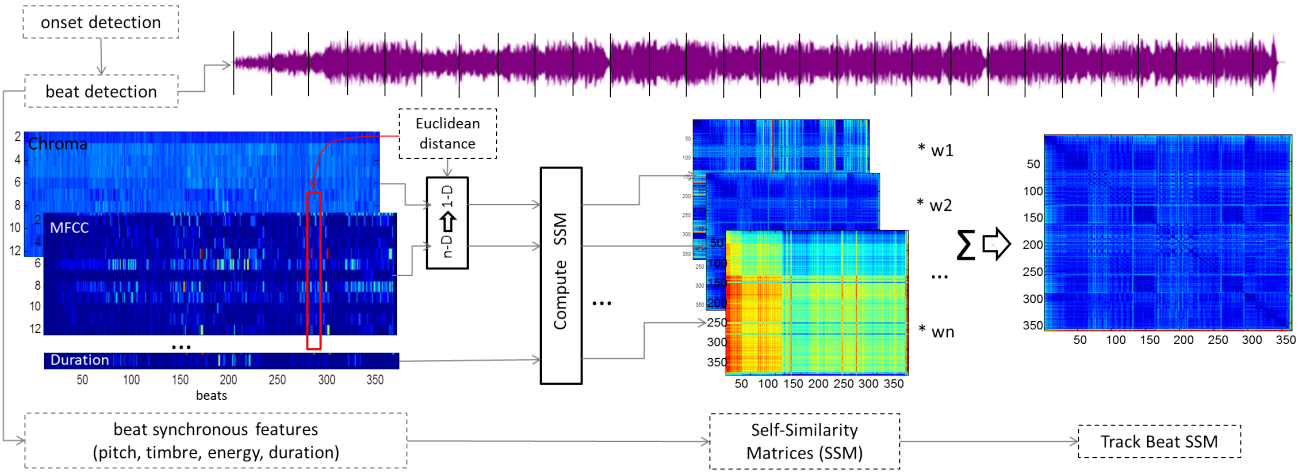

Figure 1: Beat Similarity is represented as a Self-Disimilarity Matrix (SSM) of beat-synchronous features. The process starts by tracking the positions of the beats, based on onset detection algorithms. Then beat-synchronous features are computed, namely Chroma, MFCC, Average Energy, Maximum Energy and Beat Duration. Chroma and MFCC features are reduced from 12-Dimensional to 1-D by computing the Euclidean distance. Finally, SSMs are computed for each of the features and combined in a single matrix by weighted summation.

two loudness features are computed, the average energy (of the whole duration of the beat) and the maximum energy at the beginning of the beat, computed as the peak energy in a short window at the beginning of the beat.

3.2 Beat Disimilarity Matrix

The distance between beats in a track within this feature-space is estimated by computing a self-disimilarity matrix (SSM) [8] of the beat-synchronous features. A SSM is a square matrix where time runs from left to right, as well as from the top to the bottom. The matrix is symmetric, and the diagonal is null.

For each of the audio features (Section 3.1) a SSM is computed and all resulting matrices are combined into a single one by applying a weighted sum. The SSM of a feature vector X is the matrix multiplication between transposed X (X^T) and X, after normalization of the columns of X. Normalization in case of one-dimensional features, such as the beat duration, is straight forward: vectors are subtracted from their minimum value and divided by the maximum. However, for n-dimensional features as it is the case of MFCC coefficients and Chroma features (12 dimensions each) a measure for dimension reduction is needed. In this work the Euclidean distance is used since these features as based on the auditory spectrogram, which is perceptually normalized, i.e. the geometric distance between segments is proportional to the perceptual distance. Finally, every feature SSM is combined in a weighted sum in order to obtain the general Beat Disimilarity Matrix (BSM). The whole process is represented in Figure 1.

3.3 Beat Similarity Graph

The Beat Similarity Matrix of a track is transformed into an indirected graph as it allows a much more compact representation, it is easily serializable in text human-readable formats such as *JSON* and it allows to find paths (shortest, longest, etc.) based on standard graph search algorithms. In order to reduce the amount of information, a non-directed graph with filtered weight-less edges is used. Graph edges among beats are pruned based on the following parameters: $MinSimilaritythreshold$, the minimum similarity value for two beats to be connected in the graph; $maxCandidates$, the maximum number of connections that a beat can have; and $minimumSep$, the minimum temporal separation between connected beats. This includes self-connections that are always pruned.

4. RENDERING ALGORITHMS

This work supports the use of Web standards versus native implementations as the efforts being made at present on Web technologies forecast an stable and fast cross-platform environment in the future, specially since the developments of the Web Audio API (WAAPI) [8]. At present, however, sound processing in real-time is still in a very preliminary phase. There exist alternatives to using the WAAPI such as javascript libraries, but they are still far to achieve some of the real-time transformations needed in this work, e.g. time-stretching. Our algorithms are therefore adapted to these peculiarities. The main developed algorithms are a beat concatenation module that schedules and concatenates the beats to be played; a beat-matcher that aligns the beats of active tracks with different tempo; an energy cross-fader that smoothly cross fades energy profiles of active tracks; a time-stretching module; and an algorithm to find the best transition instant between two songs.

4.1 Beat Scheduling and Concatenation

Representation of musical tracks as beat similarity graphs allows for advanced playback capabilities, in contrast to the traditional sequential playback, such as jumping to different parts of the song through similar beats. Track duration can be, therefore, adapted from the smallest loop in the graph to infinite. The music player implements a beat scheduler,

[8]https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

which queues the next beats to render based on the position of the listener by means of standard graph algorithms (i.e. Dijkstra [4]). During beat concatenation, cross-fading may be applied, although it is not necessary as beats are already cut at psycho-acoustically strategic positions (zero crossings at onsets) and transitions are generally artifact-free and smooth. However, jumping to other parts of the song may be noticeable and higher level semantic information should be incorporated, for instance, phrasing or detection of voiced fragments, which can only be split at very specific instants.

4.2 Music Track Transitions

The second enhanced characteristic of the presented system is its ability to automatically perform transitions between tow (or more) songs when the listener walks into the geographic path of a new track. The problem consists basically of aligning the beats of both tracks or beat-matching (Section 4.2.2) and smoothly cross-fading the volumes (Section 4.2.4). In typical DJ performances with analog turntables, the process of beat-matching is achieved by adjusting the speed of the turntable, which alters the pitch of the song. In order to avoid this artifact we perform time-stretch to modify the length of the beats while preserving the pitch.

Awaiting for technical advances in WAAPI in the near future, we propose two different approaches to overcome the current limitations. The first approach (ap_1) consists of pre-computing and storing the transitions and the second (ap_2) just cuts off the beats to fit the length of the tempo curve. Approach ap_1 is less exposed to sound artifacts as signal processing is carried out off-line, but it is less flexible as transitions can not be adapted in real time, they are just triggered when the user enters a transition part. Method ap_2 is more flexible and allows adaptation in real-time but artifacts may be present if the tempi of the tracks being mixed is too distant, as no time-stretching is being applied.

In both cases, an algorithm first looks for the best position to perform the transition among the active tracks. Then, it estimates a tempo envelope for the transition based on the tempi of the active tracks that is used to beat-match the beats of the active tracks. Finally, cross-fading is applied to the energy of the tracks. If the tempo difference between both tracks in the transition is higher than 50%, then the lowest song tempo is artificially doubled by adding virtual beats at the middle of each beat.

4.2.1 Finding best transition position

The first step to perform the automatic transitions is to determine the position (in beats) of the transition. In approach ap_1, it is estimated by computing a weighted cross-correlation of the beat-synchronous features (see section 3.1) of the active tracks. The operation is performed within a transition window length determined by parameters *maximum* and *minimum transition duration* (tr_{maxL}, tr_{minL}) expressed in beats. Figure 2 shows an example of the weighted correlation functions between the computed set features for a lag of 100 beats (the last 100 beats of the outgoing track and the 100 first ones of the incoming track). It can be observed how *metrical position* correlation envelopes are composed by successive peaks and valleys, which allows to accurately find the position of the tracks with similar rhythmic patterns, while the rest of the features determine the areas with higher correlation. Regarding approach ap_2, the position is estimated in real time by forcing the tracks to match downbeats.

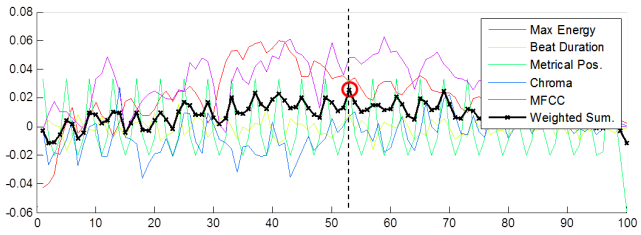

Figure 2: Cross-correlation of beat-synchronous features for 100 beats of two consecutive tracks. The red dot (at beat 52) indicates the beat with maximum correlation, which means that the transition will have a duration of 52 beats. The *metrical position* envelope is composed by successive peaks and valleys, which helps to fine-tune the exact transition position.

4.2.2 Transition tempo envelope

Tracks are played back at the original speed, but during transitions, if active tracks have different tempi, there must be a smooth tempo change. In approach ap_1, this tempo evolution is estimated as a linear interpolation envelope between the *starting* and *ending* tempi of the transition. The tempo values at the boundaries are determined by averaging the beat durations before (in-tempo) and after (out-tempo) the transition in a window of a fixed number of beats, typically set to ten. In Figure 3, it can be observed the linear tempo envelope during the transition, expressed in duration of the beats and beats-per-minutes (BPM). Regarding ap_2, the tempo evolution is computed in real-time as the average of the active tracks, weighted by the distance (in cm) of the user to each of the active tracks. Once the tempo curve is determined, beat duration from both tracks are time-stretched to meet their new duration.

4.2.3 Time stretching

Time-stretching algorithms are used to speed up or slow down the music without affecting the pitch. The stretch is performed in a beat-by-beat basis. In the off-line approach, ap_1, a phase-Locked-Vocoder [9] is used. In the case of the real-time approach, ap_2, after trying several computationally in-expensive time domain techniques such as resampling, Overlap-add (OLA), we decided to just cut the beats or fill them with silence at the end in order to match the transition tempo envelope. Both resampling and OLA affect the pitch, resulting in undesired artifacts, whereas cutting or filling the beats is almost unnoticeable assuming that tempi are not too far and given that possible occurring artifacts are masked by the mix and the energy fades. In fact, the most perceptually important features are to keep all active tracks on beat and to preserve their original pitch.

4.2.4 Energy Band Cross-fading

The last step of the automatic transitioning is to smoothly cross-fade the energy of the tracks. Typical fade shapes are linear, logarithmic, exponential or S-Curve and the slopes can be designed to be constant in gain or power. We use a logarithmic shape with constant power, that is, the sum of the squares of the envelopes is constant and equal to one.

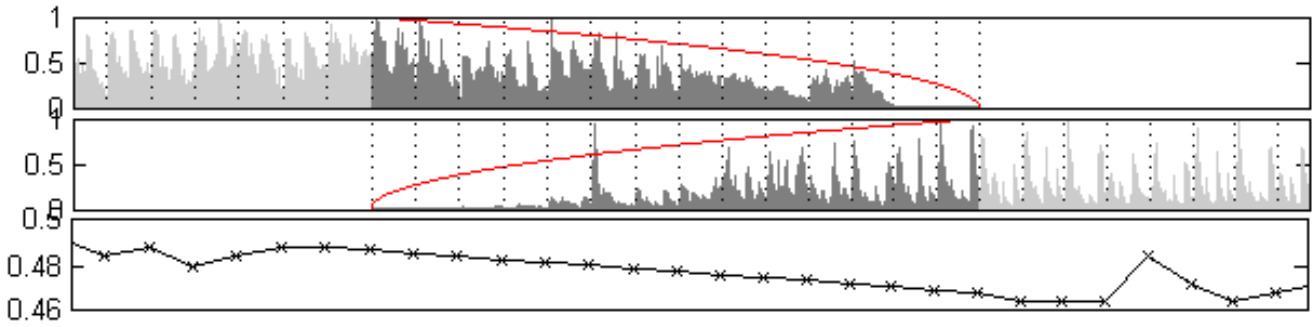

Figure 3: Example of transition between two tracks at different BPM. At the top there is the out-going track, at the middle the in-coming track and at the bottom the tempo envelope (expressed as beat duration and BPM), which is estimated as the linear regression between the *starting* and *ending* tempi of the transition. Beats during the transition are time-stretched to meet the new duration. We can also observe the constant-power cross-fading envelopes applied to each of the tracks (in red).

The resulting shapes are shown in Figure 3 and were computed as,

$$fade_{In} = \sqrt{\frac{1}{2}(1+t)}, \quad (1)$$

$$fade_{Out} = \sqrt{\frac{1}{2}(1-t)}, t = \{-1..1\}. \quad (2)$$

In approach ap_2, cross-fade envelopes can be travelled back and forth depending on the distance of the user to the active tracks. Cross-fades can be applied to the entire energy spectra (as explained above) or just to selected frequency bands (e.g. high-pass, low-pass). In this case, the process would be carried out in steps, until all energy bands are cross-faded.

5. THE APPLICATION

The application was built on Web standards (Web Audio API, HTML5 and Javascript) and cross-platform frameworks (Ionic [9] based on Cordova [10] and AngularJS [11]). The use of DMOs as containers for data and metadata allows for flexible production and consumption of the musical experience. On one side, a provider prepares and packs the music material, the mappings and the behavior and on the other side, consumers can download the package, consume and interact with it within the restrictions defined by the provider. Details regarding Provider and Consumer interfaces are given below.

5.1 Provider Interface

The producer is responsible for the creation of the music playlist, setting the travel path in a map and linking the tracks to specific segments of the path and can also tune the parameters of the audio processing algorithms and the mappings from location to music track. The producer is provided with three tools, a playlist editor, a path designer and a simulator, which were developed as desktop web applications.

[9]http://ionicframework.com
[10]http://cordova.apache.org
[11]https://angularjs.org

At present, the playlist is prepared manually off-line and the audio analysis algorithms are run in MATLAB, which generates the metadata files. In the future, the interface will have an area to drag and drop the music files that will be sent to a web-server with a Python back-end in order to be analyzed. The server will in turn send back the metadata related to the audio analysis (i.e beat position and similarity graph).

The second tool, the path-to-track mapper (PTM), allows to match path segments to tracks. Paths are designed using the Google maps editor, by indicating the path's corners with markers (google.maps.marker), and then, they are exported to a geolocation XML based format (*.kml* or *.gpx*) as shown in Figure 4. The PTM loads such a pre-defined path and a playlist description file and it allows to assign a set of consecutive path-markers to each track.

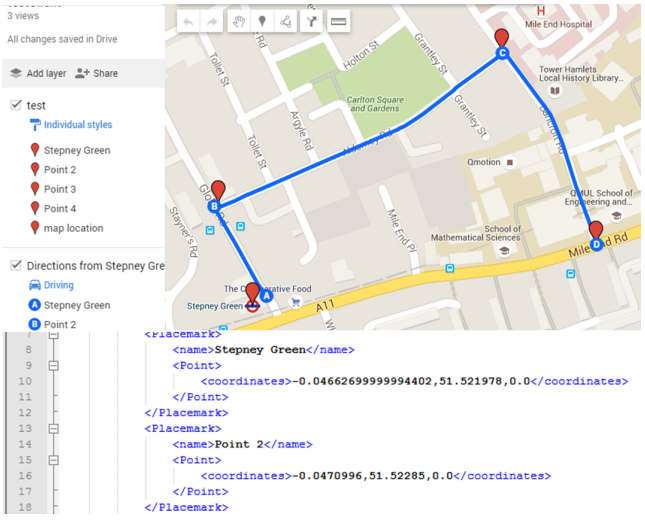

Figure 4: Map editing in google maps and example of .kml formated position of the markers in the path.

The third tool is used to simulate a user walking and playing back the DMOs. The interface allows to place the listener at a specific location and can also automatically move

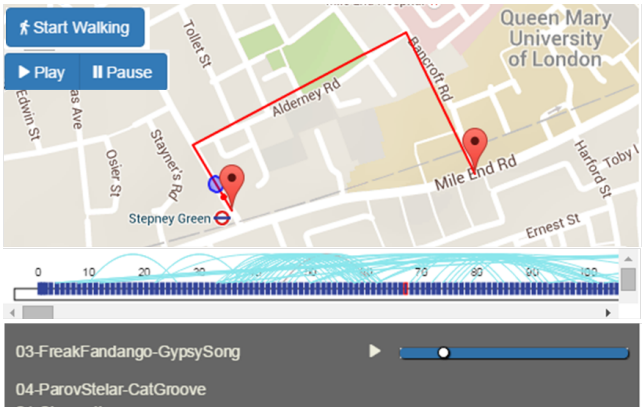

Figure 5: The Simulator loads the defined map (markers) and simulates a listener's motion, listen to the DMOs playing back and monitoring the jumps in the DMOs beats.

the listener along the path with random velocity. Additionally, the interface monitors the connections and the jumps among similar beats and includes a player showing the current track being played and the position of the current beat being rendered (Figure 5). A demo of the simulator, optimized for Google Chrome Navigator, can be found online [12]

5.2 Consumer Interface

The consumer application is run via a Web-Browser in a mobile device. It loads a DMO and shows a path in a map, the names of the tracks, the track being played, the actual position provided by the GPS and optionally the position of the beat being played, in a similar way to the Simulator (Figure 5). The client renders the music using the rendering algorithms in Section 4. It also implements a beat scheduler, selecting the following beats to render and can adapt to the position of the listener based on standard graph algorithms (Dijkstra [4]).

6. CONCLUSIONS

The presented Web Audio Framework based on the concept of Digital Music Object and audio signal processing algorithms, allows for the design of novel music consumption experiences. As an example application we present a DMO player for mobile devices, which reacts to the user geolocation in a map. The application plays back a predefined playlist of music tracks, which are linked with paths in a map. The music player has two main enhanced features which are controlled depending on the position of the listener, namely, adaptable length of the song and automatic transitions.

The length of the song can be adapted to the position and motion of the listener. This is possible because the enhanced player, instead of playing back the beats in order, it can jump between similar beats and similarity grants that a jump to a similar beat will not be noticeable. Furthermore, the representation of similarity as a graph allows to search for specific paths that get close to the position of the listener. Whenever the listener enters the path region of a new song, the transition between songs is triggered. Automatic beat-matching, cross-fading and time-stretching is carried out until the transition is finished, when the adaptive player goes back to the previous state of adapting the length of the song to the listener's position.

In the future, we will work towards an implementation of time-stretching in real-time based on Javascript and the Web Audio API, and adopt Semantic Web technologies for the representation of the Music Objects.

[12]http://www.eecs.qmul.ac.uk/~alfonso/geolocation-player/

7. ACKNOWLEDGMENTS

This paper has been supported by EPSRC Grant EP/L019981/1, Fusing Audio and Semantic Technologies for Intelligent Music Production and Consumption.

8. REFERENCES

[1] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto. Automashupper: Automatic creation of multi-song music mashups. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 22(12):1726–1737, Dec. 2014.

[2] D. De Roure. Towards computational research objects. In *Proceedings of the 1st International Workshop on Digital Preservation of Research Methods and Artefacts*, DPRMA '13, pages 16–19, New York, NY, USA, 2013. ACM.

[3] D. De Roure, G. Klyne, K. R. Page, J. P. N. Pybus, and D. M. Weigl. Music and science: Parallels in production. In *Proceedings of the 2Nd International Workshop on Digital Libraries for Musicology*, DLfM '15, pages 17–20, New York, NY, USA, 2015. ACM.

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, (1):269–271, 1959.

[5] D. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005. http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/.

[6] D. Ellis. Beat tracking by dynamic programming. *J. New Music Research, Special Issue on Beat and Tempo Extraction*, 36(1):51–60, March 2007.

[7] D. Ellis and G. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Proc. Int. Conf. on Acous, Speech, & Sig. Proc. ICASSP*, volume IV, pages 1429–1432., Hawaii, 2007.

[8] J. Foote and M. Cooper. Automatic audio segmentation using a measure of audio novelty. pages 452–455, 2000.

[9] J. Laroche and M. Dolson. New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects. In *Proc. the IEEE Workshop on Applications of Signal Processing to Audio and Acoust.*, New Paltz, New York, Oct. 17-20 1999.

[10] Y. Raimond, S. Abdallah, M. Sandler, and G. Frederick. The music ontology. In *in Porc. 7th International Symposium on Music Information Retrieval*, 2007.

[11] J. Tristan. *Creating Music by Listening*. Phd in media arts and sciences., Massachusetts Institute of Technology, September 2005.