BeatJockey: A new tool for enhancing DJ skills

Pablo Molina*, Martín Haro**, Sergi Jordá**
Music Technology Group
Universitat Pompeu Fabra
Roc Boronat, 138, 08018 Barcelona, Spain
*faival@gmail.com, **name.surname@upf.edu

ABSTRACT

We present *BeatJockey*, a prototype interface which makes use of Audio Mosaicing (AM), beat-tracking and machine learning techniques, for supporting Diskjockeys (DJs) by proposing them new ways of interaction with the songs on the DJ's playlist. This prototype introduces a new paradigm to DJing in which the user has the capability to mix songs interacting with beat-units that accompany the DJ's mix. For this type of interaction, the system suggests song slices taken from songs selected from a playlist, which could go well with the beats of whatever master song is being played. In addition the system allows the synchronization of multiple songs, thus permitting flexible, coherent and rapid progressions in the DJ's mix. *BeatJockey* uses the Reactable, a musical tangible user interface (TUI), and it has been designed to be used by all DJs regardless of their level of expertise, as the system helps the novice while bringing new creative opportunities to the expert.

Keywords

DJ, music information retrieval, audio mosaicing, percussion, turntable, beat-mash, interactive music interfaces, real-time, tabletop interaction, reactable.

1. INTRODUCTION

After the term Diskjockey (DJ) was coined in the early 30s, the first DJs used a single device to sequentially playback songs on the radio (Radio-DJ). Afterwards, the appearance of portable turntables on the scene inspired club/rave DJs to use two turntables and a mixer. In Jamaica, Scratching-DJs [9], and Mixing-DJs [25], started to increase complex manipulations on the turntables and the mixer, in order to drive peoples' attitude to the mix into many emotional states. Nowadays, when amateur-DJs can afford digital DJing systems to perform at home informally for their friends, the DJ music industry is focused on pushing these so called Bedroom-DJs [22] to the stage [9].

Mixing-DJs are essentially interested in the problem of beat-matching and cross-fading songs as *smoothly* as possible [25]. In that sense, we find that Mixing-DJs think about four main questions when they aim to introduce new music in their performances. These questions consider *What?*, *When?*, *Which?* and *How?* new music content will be introduced.

First, the performer needs to know *What?* songs will accompany well with the song being played. Second, knowing *When?* to introduce new music material lets the DJ manipulate the flow of the mix in order to drive the expectations of the audience. Third, the practice of beat-matching allows DJs to plan *Which?* song elements would sound more noticeable when playing different songs. Last, the DJs needs to know *How?* to seamlessly synchronize songs, a help that most currently available digital DJing systems[1] already provide.

In this paper we present *BeatJockey*, a tool for DJs that supports and enhances the traditional playback interaction of DJing. Besides from the traditional features already present in most current DJ systems, this system is also capable of suggesting and introducing music material, thus providing answers to the four aforementioned questions (*What?*, *When?*, *Which?* and *How?*). We believe that if the system is capable of playing back song slices preserving the event and rhythm structure of a master song then such a system will have acquired the basic knowledge of an experienced mixing-DJ, and therefore will be ready to help the non-experienced one. In order to do so, the system suggests beat-slices, taken from other songs of the playlist, that present similarities to the master song being played. These beat-slices form sequences that rhythmically match the master song. In addition, the system supports the synchronization of multiple songs, thus allowing coherent and rapid progressions between the songs in the mix.

BeatJockey uses beat-tracking [5] to help decide *When?* to playback beat-slices. A set of content descriptors [20] extracted from the songs, and machine learning techniques [11] indicate *What?* beat-slices could be played. In addition, *BeatJockey* uses a concatenative synthesis technique called Audio Mosaicing (AM) [17], in order to recreate a target sound by using slices from other sources. Our AM approach encodes *Which?* beat-slices should be sequentially played in order to resemble a master song. Lastly, the current *BeatJockey* prototype has been implemented in the Reactable [14] musical tangible user interface (TUI), changing consequently the normal way *How?* DJs manage to synchronize songs.

The remainder of this document is structured as follows: Section 2 overviews related systems that have contributed to DJing. Next, Section 3 presents some new possibilities for DJs to interact with songs, and describes the system's implementation and control. Finally, in Section 4 we evaluate how the users have assessed the music produced by our system.

[1]Stanton (Final Scratch), Rane (Scratch Live), Native Intruments (Traktor Scratch Pro)

2. RELATED WORK

This section introduces a set of systems under the scope of DJing. In the literature there are different approaches that solve individually the (*What?*, *When? Which?* and *How?*) problems of introducing music material. Accordingly, we classify these works with regard to the question they solve.

- *What?* songs would propperly accompany a given master song is addressed in [2, 15]. The authors present interfaces that take content descriptors into account in order to suggest new songs.
- *When?* to introduce sound events is addressed in [8, 12], in which the authors describe different synchronization strategies based on beat and tempo of the songs.
- The AM approaches by [17] and LoopMash[2] address *Which?* sequence of sound units should be sequentially played in order to resemble a given target sound. In addition to AM, different techniques for synthesizing a target sound out of pre-existing sound exist [21].
- In [9], the *How?* DJs are able to practice DJing with the help of devices is addressed. Some of these devices can replace the traditional vinyl[3], are intended to be used at home[4], or are oriented for gaming[5] [9]. Experimental DJing interfaces also exist that augment traditional equipments [1, 3, 19], while others aim to provide control over other performance parameters [18, 24]. In [6, 16, 23] some systems that offer novel ways of controlling the playback of songs are presented. In addition, a variety of systems supporting multi-touch interaction for DJs, either commercial[6] and research[7] oriented are found. In [10] support for DJing interaction is implemented under the Reactable[8], the same TUI used by*BeatJockey*.

We find however that none of these interfaces contemplates the *What?*, *When?*, *Which?* and *How?* problems at the same time.

3. A NEW PARADIGM OF SONG INTERACTION FOR THE DJ

The system we propose implements the basic set of standard DJ functionalities such as, playback of multiple songs, tempo adjustment, song's gain, filters, song positioning, etc.. Moreover, it also introduces new functionalities for enhancing DJs' creativity at their performances.

3.1 New functionalities

For every beat-slice of the master song the system tries to find a matching beat-slice from another song in the DJ's playlist. The suggested beat sounds will build beat sequences that resemble the master song. These beat sequences are sorted and disposed using AM. This results in a beat-mashed sequence that rhythmically accompanies the master song, and which the performer has the possibility to put in the foreground or leave in the background. Moreover, at any beat, the DJ has the possibility to drive the beat sequences synchronously towards any selected song.

With such functionalities, we speculate that the system might decrease the performance and cognitive load of experienced mixing-DJs, thus inciting and enhancing their creativity.

3.2 System's corpus

BeatJockey uses two layers of information extracted from the music in order to introduce a new paradigm of song interaction for DJing.

- The first layer of information provides a solution to know *When?* to trigger beat-slices. Since the *beat* is the event that the majority of people would follow in order to respond to the rhythm of the music [7], we have assumed that the *beat* is the basic cue that a DJ uses to synchronize songs. *BeatJockey* uses the algorithm BeatRoot developed by Dixon [5] to extract the beat times of the songs.
- The second layer of information solves *What?* sounds are more likely to sound coherent when played back together with a master song. In order to compare beat-slices of sound our system extracts two kinds of information from them. In [11], a collection of content descriptors useful to classify drum sounds is described. From this collection we have selected the following set: spectral energy, spectral spread and flatness, Mel-frequency cepstral coefficients and Bark-bands. These descriptors are computed with the help of an in-house library[9]. Second, we use a statistical model for labeling each of the beat-slices [11]. The model (support vector machine) classifies beat-slices into four different Percussive Class Labels (PCL), *Bass drum, Snare drum, Hi Hat* and *Cymbal* (BD, SD, HH, CY), that reflect which elements of a drum kit are more likely to be present in the beat.

The system's information corpus is illustrated in Figure 1. In this figure, the BeatRoot algorithm extracts the beat-times for each song of the DJ's playlist. Then, each beat is cut at the quarter-note level. For each beat-slice we then take the set of content descriptors previously mentioned, and their PCLs. The system packs the beat-slice information into their informational points. As these points' dimensions are formed by both high-level (PCLs) and low-level (content descriptors) information, they reflect information about the beat-slice sounds. This allows the system to find similarities between the beat sounds contained in the database, thus providing a solution for the questions of *What?* and *When?*.

3.3 System's performance modes

AM helps to suggest *Which?* sequences of beat-slices will be sequentially arranged and played, in order to maintain event-wise sychronization and *Percussive Structure* (PS), with respect to a master song. We define the PS of both units, a master song's fragment and the suggested beat-sequences, as the succession of their beat-slices's PCLs (e.g. (BD, HH); (HH,SD); (BD,CY)). Our AM approach always tries to preserve the PS's between the master song and the suggested beat-sequences (see figure 2).

1. In *Beat-mash* mode, the suggested beat-sequences are built from beat-slices of different selected songs. A

[2]`http://loopmash.com/`
[3]Technics (SL-DZ1200), Denon (DN-2500F), Vestax (Spin), M-Audio (Torq), Tonium (Pacemaker)
[4]Hercules DJ Control MP3
[5]Activision (DJ Hero), Arcade Games (Beatmania)
[6]Stanton (SCS.3D), `http://www.smithsonmartin.com/products/emulator/`, `http://www.algoriddim.com/djay-ipad`, `http://ipadmixr.com/`
[7]`http://www.soundwidgets.com/stribe/`
[8]`http://www.reactable.com`.
[9]`http://mtg.upf.edu/technologies/essentia`

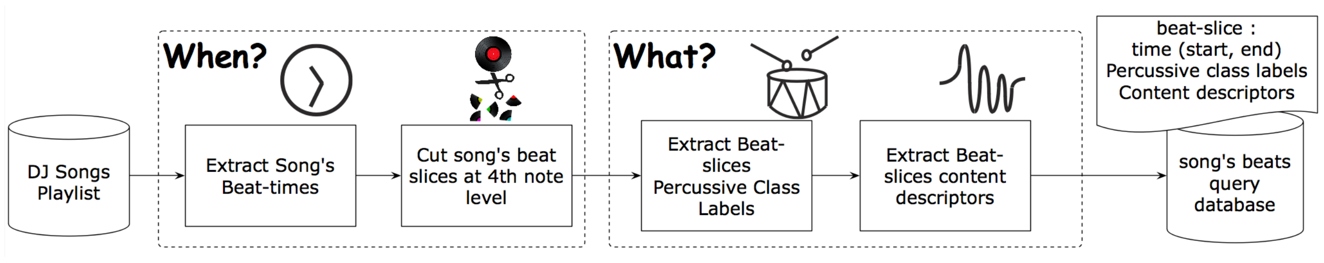
When?
What?
DJ Songs Playlist
Extract Song's Beat-times
Cut song's beat slices at 4th note level
Extract Beat-slices Percussive Class Labels
Extract Beat-slices content descriptors
beat-slice : time (start, end) Percussive class labels Content descriptors
song's beats query database

Figure 1: Two different infomation layers (*What?* and *When?*) are analyzed by the system to characterize music events in order to suggest music material.

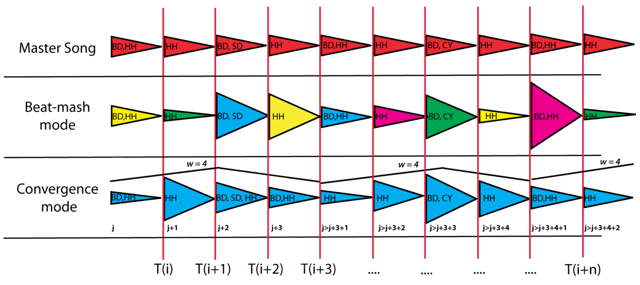

Figure 2: An overview of the new song interaction functionalities. Vertical lines depict the master-song's beat-times, where (i) denotes the beat number of a song with the unit being the quarter-note. Color denotes the different songs in the DJ's playlist. The area of the triangle denotes the similarity of the suggested beat-slices with respect to the master-song's beats. The first row illustrates a master song decomposed in beat-slices of sound that are played sequentially. The second row corresponds to the *Beat-mash* mode and the third row to the *Convergence* mode. Here *j* denotes the target song's beats indexes and *w* a PS query-window's size. See text for further details.

Nearest Neighbor (NN) algorithm [4], with euclidean distance is used to search for similar beat-slices. Given a target beat-slice, its most similar slice will be the one containing strictly the same PCLs and being the closest, in euclidean distance, according to its content descriptors.

2. In *Convergence* mode the intention is to converge the rhythms between the master and one selected target song. In that case, beat-sequences of w beats with the same PS of the master song, will be searched. The beat-sequences are compounds of beat-slices with increasing beat indexes in the target song. This assures a progression towards the target song, while keeping an automatic synchronization between the master and a target song. The user may change the target song at any beat time. Consequently, this mode allows for rapid and coherent progressions towards different songs of a playlist.

In both modes, when the system does not find a similar beat-slice or a target beat-sequence, it does not suggest any suggestion at this particular moment.

3.4 System's control

In order to reinforce the How? problem, we have implemented *BeatJockey* within the Reactable application [13], as we believe that both sides can win from this symbiosis. *BeatJockey* has been designed taking into account both the specific affordances of this device as well as the prevalent turntable metaphor, nevertheless its main functioning principles could be easily ported to other interfaces. On one hand, *BeatJockey* extends the limited DJ interaction that Reactable implements. On the other hand, the Reactable's multitouch tangible interface provides affordances comparable to analog-digital devices. It allows different users to perform in the same interface by sharing the controls on the surface, and it also offers a modular approach that eases the inclusion of new features, as long as they adhere to the interface's main metaphors.

Reactable offers four different types of objects with varied typologies: sound generation objects; sound processing objects; control objects; and global objects, that modify parameters affecting all the objects in the table. The functioning principle of these objects are the same for all four types. Objects are activated when they are put on the table's surface, and the object's control parameters may be modified by rotating them, and by moving virtual sliders or selectors around the objects, with the fingers.

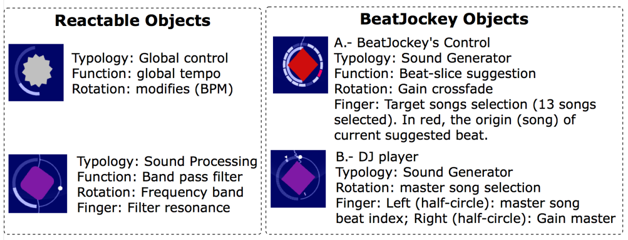
Reactable Objects
Typology: Global control
Function: global tempo
Rotation: modifies (BPM)
Typology: Sound Processing
Function: Band pass filter
Rotation: Frequency band
Finger: Filter resonance
BeatJockey Objects
A.- BeatJockey's Control
Typology: Sound Generator
Function: Beat-slice suggestion
Rotation: Gain crossfade
Finger: Target songs selection (13 songs selected). In red, the origin (song) of current suggested beat.
B.- DJ player
Typology: Sound Generator
Rotation: master song selection
Finger: Left (half-circle): master song beat index; Right (half-circle): Gain master

Figure 3: Overview of *BeatJokey's* control objects.

With these considerations, we have designed the control objects shown in Figure 3. Their sliders and the selectors provide affordances comparable to the classical DJ setup, although the positioning of a track is centered at the beat time. The system does not support yet time-stretching of songs and beat-sequences.

The performance modes[10], *Convergence* (one song), and *Beat-mash* (more than one song), are activated by selecting songs with the controller A. If no target songs are selected the system does not suggest any beat. If object A is taken out of the table and put back, the selection of target songs will be reset (no songs selected). There may be different users using combinations of A and B objects, thus allowing for multi-DJ collaborative performances.

4. EVALUATION

We have not yet evaluated *BeatJockey* as a live tool, but as a proof of concept we have done a preliminary evalua-

[10]please refer to video, http://dl.dropbox.com/u/13952105/BeatJockey.mov

tion of *BeatJockey*'s performance, asking ten listeners to listen to and evaluate the results of previously recorded *BeatJockey* sessions. Evaluation results for the *Beat-mash* mode reflected that the suggested beat-slices were preferred over randomly generated ones (t-test p-value<0.05). For the *Convergence* mode, evaluation results were not statistically significant (t-test p-value $= 0.782$) for determining the most appropriate window size ($w = 2$, 4 and 6 were tested). Nevertheless, we find that when w is too small ($w<=2$), the target song does not progress and stays in a beat-sequence that matches perfectly the master's PS. Conversely, when too large values of w are used ($w>=6$), the target beat-sequences are not found, and the system does not preserve the continuity of the suggested beat-sequence. Therefore, our final implementation uses a $w = 4$.

5. CONCLUSIONS

We have overviewed current trends in the development of DJ supporting systems, and we have introduced *BeatJockey*, a system which takes into account the basic DJs' playing rules. With these rules, *BeatJockey* is able to support non-experienced mixing-DJs while it also provides to more experienced DJs, new ways to interact with songs. *BeatJockey* also extends the Reactable functionalities taking benefits from the Reactable's main functioning principles.

BeatJockey needs further refinement. The mapping between Reactable objects and system functions needs to be further studied and improved in order to achieve a better 'turntable' metaphor. Moreover, in order to avoid silent beat-slice suggestions, we need to allow the system to provide more flexible matchings (e.g. not taking into account PCLs) and also let the user control the w parameter.

The interface is yet to be evaluated with both expert DJs and novice users. This will be done in the near future. An online implementation of the analysis stage could help to synchronize DJ sessions between different performances at different places. We think, this online implementation would provide a rich space of interaction between multiple performers and audiences.

6. REFERENCES

[1] T. Andersen. Mixxx: Towards novel DJ interfaces. In *Proc. of NIME*, pages 30–35. National University of Singapore Singapore, Singapore, 2003.

[2] D. Baur, T. Langer, and A. Butz. Shades of music: Letting users discover sub-song similarities. *ISMIR*, 2009.

[3] T. Beamish, K. Van Den Doel, K. MacLean, and S. Fels. D'groove: A haptic turntable for digital audio control. *Proc. of ICAD, Boston, MA*, 2003.

[4] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *Database Theory—ICDT'99*, pages 217–235, 1999.

[5] S. Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

[6] K. Fukuchi. Multi-Track Scratch Player on a Multi-Touch Sensing Device. *Entertainment Computing–ICEC 2007*, pages 211–218, 2007.

[7] F. Gouyon and S. Dixon. A review of automatic rhythm description systems. *Computer Music Journal*, 29(1):34–54, 2005.

[8] G. Griffin, Y. Kim, D. Turnbull, and P. Swarthmore. Beat-sync-mash-coder: A web application for real-time creation of beat-synchronous music mashups. *ICASSP*, 2010.

[9] K. Hansen. The acoustics and performance of DJ scratching. *Doctoral Thesis Stockholm, Sweden*, 2010.

[10] K. Hansen and M. Alonso. More DJ techniques on the reactable. In *Proc. of NIME*, 2008.

[11] M. Haro and P. Herrera. From low-level to song-level percussion descriptors of polyphonic music. In *International Conference on Music Information Retrieval, Kobe, Japan*, 2009.

[12] H. Ishizaki, K. Hoashi, and Y. Takishima. Full-automatic DJ mixing system with optimal tempo adjustment based on measurement function of user discomfort. *ISMIR*, 2009.

[13] S. Jorda. On stage: the reactable and other musical tangibles go real. *International Journal of Arts and Technology*, 1(3):268–287, 2008.

[14] S. Jorda, M. Kaltenbrunner, G. Geiger, and R. Bencina. The reactable*. In *Proc. of ICMC, Barcelona, Spain*, pages 579–582, 2005.

[15] A. Kapur, R. McWalter, and G. Tzanetakis. New Music Interfaces for Rhythm-Based Retrieval. In *Proceedings of the 6th International Conference on Music Information Retrieval*. Citeseer, 2005.

[16] S. Kiser. spinCycle: a color-tracking turntable sequencer. In *Proc. of NIME*, pages 75–76. IRCAM—Centre Pompidou, 2006.

[17] A. Lazier and P. Cook. MOSIEVIUS: Feature driven interactive audio mosaicing. In *Digital Audio Effects (DAFx)*, 2003.

[18] T. Lippit. Turntable music in the digital era: designing alternative tools for new turntable expression. In *Proc. of NIME*, pages 71–74. IRCAM—Centre Pompidou, 2006.

[19] A. Pabst and R. Walk. Augmenting a rugged standard DJ turntable with a tangible interface for music browsing and playback manipulation. In *3rd International Conference on Intelligent Environments, 2007. IE 07.*, pages 533–535. IET, 2008.

[20] G. Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. *CUIDADO IST Project Report*, pages 1–25, 2004.

[21] D. Schwarz, G. Beller, B. Verbrugghe, S. Britton, et al. Real-time corpus-based concatenative synthesis with catart. In *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-06),(Montreal, Quebec, Canada)*, pages 279–282. Citeseer, 2006.

[22] P. Terrett. *Bedroom DJ: a beginner's guide*. Omnibus Pr & Schirmer Trade Books, 2003.

[23] Y. Tomibayashi, Y. Takegawa, T. Terada, and M. Tsukamoto. Wearable DJ system: a new motion-controlled DJ system. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology*, pages 132–139. ACM, 2009.

[24] N. Villar, H. Gellersen, M. Jervis, and A. Lang. The ColorDex DJ system: a new interface for live music mixing. In *Proc. of NIME*, page 269. ACM, 2007.

[25] S. Webber. *DJ Skills: The essential guide to Mixing and Scratching*. Focal Press, 2007.