A long-range self-similarity approach to segmenting DJ mixed music streams

Tim Scarfe, Wouter M. Koolen and Yuri Kalnishkan

Computer Learning Research Centre and Department of Computer Science,
Royal Holloway, University of London, Egham, Surrey, TW20 0EX, United Kingdom
{tim,wouter,yura}@cs.rhul.ac.uk

Abstract. In this paper we describe an unsupervised, deterministic algorithm for segmenting DJ-mixed Electronic Dance Music (EDM) streams (for example; podcasts, radio shows, live events) into their respective tracks. We attempt to reconstruct boundaries as close as possible to what a human domain expert would engender. The goal of DJ-mixing is to render track boundaries effectively invisible from the standpoint of human perception which makes the problem difficult.
We use Dynamic Programming (DP) to optimally segment a cost matrix derived from a similarity matrix. The similarity matrix is based on the cosines of a time series of kernel-transformed Fourier based features designed with this domain in mind. Our method is applied to EDM streams. Its formulation incorporates long-term self similarity as a first class concept combined with DP and it is qualitatively assessed on a large corpus of long streams that have been hand labelled by a domain expert.

Keywords: music, segmentation, DJ mix, dynamic programming

1 Introduction

Electronic Dance Music tracks are usually mixed by a DJ, which sets EDM streams apart from other genres of music. Mixing is the *modus operandi* in electronic music. We first transform the audio file into a time series of features (grouped into tiles) and transform those tiles into a domain where any pair from the same track would be distinguishable by their cosine. Our features are based on a Fourier transform with kernel filtering to accentuate instruments and intended self-similarity. We create a similarity matrix from these cosines and derive from it a cost matrix showing costs of a fitting a track at a given time with a given width. We use Dynamic Programming (DP) to create the cost matrix and again to perform the most economical segmentation of the cost matrix to fit a predetermined number of tracks.

A distinguishing feature of our algorithm is that it focuses on long term self similarity of segments rather than short term transients. Dance music tracks have the property that they are made up of repeating regions, and the ends are almost always similar to the beginning. For this reason we believe that some techniques from structural analysis fail to perform as well for this segmentation task

because we focus on the concept of self-similarity ranging over a customisable time horizon. Our method does not require any training or tenuous heuristics to perform well.

The purpose of this algorithm is to reconstruct boundaries given a fixed number of tracks known in advance (their names and order are known). This is relevant when one has recorded a show, downloaded a track list and needs to reconstruct the indices given a track list. The order of the indices reconstructed is critical so that we can align the correct track names with the reconstructed indices. If the track list was not known in advance the number of tracks could be estimated in most cases.

To mix tracks DJs always match the speed or *BPM* (beats per minute) of each adjacent track during a transition and align the major percussive elements in the time domain. This is the central concept of removing any dissonance from overlapping tracks. Tracks can overlap by any amount. DJs increase adjacent track compatibility further by selecting adjacent pairs that are harmonically compatible and by applying spectral transformations (EQ).

The main theme of the early literature was attempting to generate a novelty function to find points of change using distance-based metrics or statistical methods. Heuristic methods with hard decision boundaries were used to find the best peaks. A distinguishing feature of our approach is that we evaluate how well we are doing compared to humans for the same task. We compare our reconstructed indices to the ones created by a human domain expert.

J. Foote et al ([1] [2] [3] [4] [5]) have done a significant amount of work in this area and the first to use similarity matrices. Foote evaluated a Gaussian tapered checkerboard kernel along the diagonal of a similarity matrix to create a 1d novelty function. One benefit to our approach is that our DP allows any range of long-term self similarity (which relates to the fixed kernel size in Foote's work).

Goodwin et al. also used DP for segmentation ([6] and [7]). Their intriguing supervised approach was to perform Linear Discriminant Analysis (LDA) on the features to transform them into a domain where segmentation boundaries would be emphasised and the feature weights normalised. They then reformulated the problem into a clustering DP to find an arbitrary number of clusters. We believe the frame of mind for this work was structural analysis, because it focuses on short term transients (mitigated slightly by the LDA) and would find segments between two regions of long term self similarity. Goodwin was the first to discuss the shortcomings of novelty peak finding approaches. Goodwin's approach is not optimized to work for a predetermined number of segments and depends on the parametrization and training of the LDA transform.

Peeters et al ([8] [9]) did some interesting work combining k-means and a transformation of the segmentation problem into Viterbi (a dynamic program).

We compare our error to the relative error of cue sheets created by human domain experts. We focus directly on DJ mixed electronic dance music.

In the coming sections we will describe the Data Set (Section 2), the Evaluation Criteria (Section 3), the Test Set (Section 4), Data Preprocessing (Sec-

tion 5), Feature Extraction (Section 6), Cost Matrix (Section 7), Computing the Best Segmentation (Section 8), Experiment Methodology and Results (Section 9), and finally Conclusions (Section 10).

2 Data Set

We have been supplied with several broadcasts from three popular radio shows. These are: Magic Island, by Roger Shah (108 shows); A State of Trance with Armin Van Buuren (110 shows); and Trance Around The World with Above and Beyond (99 shows) (Total 317 shows). The show genres are a mix of Progressive Trance, Uplifting Trance and Tech-Trance. We believe this corpus is the largest of its kind used in the literature (see [10]). The music remains uninterrupted after the introduction (no silent gaps). The shows come in 44100 samples per second, 16 bit stereo MP3 files sampled at 192Kbs. We resampled these to 4000Hz 16 bit mono (left+right channel) WAV files to allow us to process them faster. We have used the SoX [11] (Sound eXchange) program to do this. These shows are all 2 hours long. The overall average track length is 5 and a half minutes and normally distributed. The average number of tracks is 23 for ASOT and TATW, 19 for Magic Island. There is a guest mix on the second half of each show. The guest mix DJs show off their skills with some of the most technically convoluted mixing imaginable.

3 Evaluation Criteria

We perform two types of evaluation: average track accuracy (in seconds) given as $\frac{1}{|P|}\sum_{i=1}^{|P|}|P_i - A_i|$ (P is constructed indices, and A is the human indices) and a measure of precision. The precision metric is the percentage of matched tracks within different intervals of time (thresholds) $\{60, 30, 20, 10, 5, 3, 1\}$, in *seconds* as a margin around any of the track indices we have been given. The precisions metric is invariant to alignment of the constructed indexes.

4 Test Set

There is already a large community of people interested in getting track metadata for DJ sets. CueNation ([12]) is an example of this. CueNation is a website allowing people to submit *cue sheets* for popular DJ Mixes and radio shows. A cue sheet is a text file containing time metadata (indices) for a media file.

We had our indices and radio shows provided to us and hand captured by *Dennis Goncharov*; a domain expert and one of the principal contributors to CueNation. As a result of this configuration; we can assume the alignment between the cue sheet and the radio show recording is exact.

Dennis Goncharov provided us with this description of how he captures the indices. To quote from a personal email exchange with Dennis:

The transition length is usually in factors of 8 bars (1 bar is 4 beats. At 135 beats per minute, 8 bars is 14.2 sec). It is a matter of personal preference which point of the transition to call the index. My preference is to consider the index to be the point at which the second track becomes the focus of attention and the first track is sent to the background. Most of the time the index is the point at which the bass line (400Hz and lower) of the previous track is cut and the bass line of the second track is introduced. If the DJ decides to exchange the adjacent tracks gradually over the time instead of mixing them abruptly then it is up to the cuesheet maker to listen further into the second track noting the musical qualities of both tracks and then go back and choose at which point the second track actually becomes the focus of attention.

5 Data Preprocessing

We went through the dataset carefully and removed some of the indices given and the corresponding audio when they did not correspond to actual musical tracks. This was for the show introductions (at the beginning) or for the introductions given to the guest mixes. The algorithm still performs similarly in the case of removing just these indices and leaving the audio intact underneath. When we removed audio from the shows because of extraneous introductions the following indices were nudged accordingly so that they still pointed to the equivalent locations in the audio stream. For those wishing to use this algorithm in practice with pre-recorded shows; the introductions at the start of the shows can be thought of as being fixed length (with a different length for each show type).

6 Feature Extraction

We used SoX [11] to downsample the shows to 4000Hz. We are not particularly interested in frequencies above around 2000Hz because instrument harmonics become less visible in the spectrum as the frequency increases. The Nyquist theorem ([13]) states that the highest representable frequency is half the sampling rate, so this explains our reason to use 4000Hz. We will refer to the sample rate as R. Let L be the length of the show in samples.

Fourier analysis allows one to represent a time domain process as a set of integer oscillations of trigonometric functions. We used the discrete Fourier transform (DFT) to transform the tiles into the frequency domain. The DFT given as $F(x_k) = X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}$ transforms a sequence of complex numbers $x_0, \ldots, x_N$ into another sequence of complex numbers $X_0, \ldots, X_N$ where $e^{-i2\pi \frac{k}{N} n}$ are points on the complex unit circle. Note that the fftw algorithm that we used to perform this computation (see [14]) operates significantly faster when N is a power of 2 so we zero pad the input to make that the case. Because we are passing real values into the DFT fuction, the second half of the result is a rotational copy of the first half. As we are not always interested in the entire

range of the spectrum, we use l to represent a low pass filter (in Hz) and h the high pass filter (in Hz). So we will capture the range from h to l on the first half of the result of F. We always discard the imaginary components of F.

Show samples are collated into a time series $Q_i, i \in \{1, 2, \ldots, \lfloor \frac{L}{M_s} \rfloor\}$ of contiguous, non-overlapping, adjacent *tiles* of equal size. Samples at the end of the show that do not fill a complete tile get discarded. We denote the tile width by M in seconds (an algorithm parameter) and M_s in samples ($M_s = M \times R$). For each tile $t_i \in Q$ we take the DFT $F(t_i)$ and place a segment of it into feature matrix D_i ($|Q|$ feature vectors in D). For each DFT transform we select vector elements $\lceil h \times \frac{M_s}{R} \rceil + 1$ to $\lceil l \times \frac{M_s}{R} \rceil + 1$ to allow effective spectral filtering.

To focus on the instruments and improve performance we perform convolution filtering on the feature vectors in D, using a Gaussian first derivative filter. This works like an edge detection filter but also expands the width of the transients (instrument harmonics) to ensure that feature vectors from the same song appear similar because their harmonics are aligned on any distance measure (we use the cosines). This is an issue because of the extremely high frequency resolution we have from having such large DFT inputs. Typically a STFT approach is used which has smaller DFTs (for example [15]).

The Gaussian first derivative filter is defined as $-\frac{2G}{B^2} e^{-\frac{G^2}{B^2}}$ where $G = \{-\lfloor 2B \rfloor, \lfloor -2B + 1 \rfloor, \ldots, \lfloor 2B \rfloor\}$, $B = b\left(\frac{N}{R}\right)$. b is the bandwidth of the filter in Hz and this is a parameter of the algorithm. After the convolution filter is applied to each feature vector in D, we take the absolute values and normalize each one

$$D_i = |D_i|, \forall i \in D, \qquad D_i = \frac{D_i}{\|D_i\|}, \forall i \in D.$$

Because the application domain is well defined in this setting, we can design features that look specifically for what we are interested in (musical instruments). Typically in the literature; algorithms use an amalgam of general purpose feature extractors. For example; spectral centroid, spectral moments, pitch, harmonicity ([16]). We construct a disimilarity matrix of cosines S from $D \times D^\top$ (dot products).

7 Cost Matrix

We now have a dissimilarity matrix $S(i, j)$ as described in Section 6.

Let w and W denote the minimum and maximum track length in seconds, these will be parameters.

Intuitively, features within the same track are reasonably similar on the whole, while pairs of tiles that do not belong to the same track are significantly more dissimilar. We define $C(f, t)$, the cost of a candidate track from tile f through tile t, to be the sum of the dissimilarities between all pairs of tiles inside it, normalized on track length:

$$C(f,t) \;=\; \frac{\sum_{i=f}^{t} \sum_{j=f}^{t} S(i,j)}{\sqrt{t-f+1}}$$

As a first step, we pre-compute C for each $1 \leq f \leq t \leq T$. Direct calculation using the definition takes $O(TW^3)$ time. However, we can compute the full cost matrix in $O(WT)$ time using the following recursion for the unnormalized quantity $\tilde{C}(f,t) = C(f,t)(t-f)$ (for $f+1 \leq t-1$)

$$\tilde{C}(f,t) \;=\; \tilde{C}(f+1,t) + \tilde{C}(f,t-1) - \tilde{C}(f+1,t-1) + S(f,t) + S(t,f).$$

Note that the normalization step can be done independently of the DP procedure. We discovered experimentally that normalizing using the square root of the track length was advantageous. Doing so slightly discourages tracks of a larger length.

8 Computing Best Segmentation

We obtain the cost of a full segmentation by summing the costs of its tracks. The goal is now to efficiently compute the segmentation of least cost.

A sequence $\boldsymbol{t} = (t_1, \ldots, t_{m+1})$ is called an m/T-segmentation if

$$1 = t_1 < \ldots < t_m < t_{m+1} = T + 1.$$

m is the number of tracks we are trying to find and is a parameter of the algorithm. We use the interpretation that track $i \in \{1, \ldots, m\}$ comprises times $\{t_i, \ldots, t_{i+1} - 1\}$. Let $\mathbb{S}_m^T$ be the set of all m/T-segmentations. Note that there is a very large number of possible segmentations

$$|\mathbb{S}_m^T| \;=\; \binom{T-1}{m-1} = \frac{(T-1)!}{(m-1)!(T-m)!} = \frac{(T-1)(T-2)\cdots(T-m+1)}{(m-1)!} \geq \left(\frac{T}{m}\right)^{m-1}.$$

For large values of T, considering all possible segmentations using brute force is infeasible. For example, a two hour long show with 25 tracks would have more than $\left(\frac{60^2 \times 2}{25}\right)^{24} \approx 1.06 \times 10^{59}$ possible segmentations!

We can reduce this number slightly by imposing upper and lower bounds on the song length. Recall that W is the upper bound (in seconds) of the song length, w the lower bound (in seconds) and m the number of tracks. With the track length restriction in place, the number of possible segmentations is still massive. A number now on the order of 10^{56} for a two hour show with 25 tracks, $w = 190$ and $W = 60 \times 15$.

Our solution to this problem is to find a dynamic programming recursion.

The loss of an m/T-segmentation $\boldsymbol{t}$ is

$$\ell(\boldsymbol{t}) \;=\; \sum_{i=1}^{m} C(t_i, t_{i+1} - 1)$$

We want to compute

$$\mathcal{V}_m^T \;=\; \min_{\boldsymbol{t} \in \mathbb{S}_m^T} \ell(\boldsymbol{t})$$

To this end, we write the recurrence

$$\mathcal{V}_1^t \;=\; C(1,t)$$

and for $i \geq 2$

$$\mathcal{V}_i^t \;=\; \min_{\boldsymbol{t}\in\mathbb{S}_i^t} \ell(\boldsymbol{t}) \;=\; \min_{t_i} \min_{\boldsymbol{t}\in\mathbb{S}_{i-1}^{t_i-1}} \ell(\boldsymbol{t}) + C(t_i,t) \;=$$

$$\min_{t_i} C(t_i,t) + \min_{\boldsymbol{t}\in\mathbb{S}_{i-1}^{t_i-1}} \ell(\boldsymbol{t}) \;=\; \min_{t_i} C(t_i,t) + \mathcal{V}_{i-1}^{t_i-1}$$

In this formula t_i ranges from $t-W$ to $t-w$. We have $T \times m$ values of $\mathcal{V}_m^T$ and calculating each takes at most $O(W)$ steps. The total time complexity is $O(TWm)$.

9 Methodology and Results

We created a validation set of the first 10 episodes from each radio show (30 total) and found the best parameters with a continuous random search optimizing the absolute average accuracy evaluation criterion. We explored 2000 permutations in the search. We searched across the following parameter space:

$$M \in \{1,2,\ldots,25\} \qquad w \in \{120,180,240\} \qquad b \in \{1,2,\ldots,20\}$$

This search did not take long as running time is linear in the parameters. The parameters we found are shown in Table 1. Running the algorithm once on a 2 hour long show takes a couple of seconds on a fast PC and almost all of that time is loading the WAVE file for the show into memory. We had already batch converted the shows into the WAVE files from MP3 and this process took significantly longer, perhaps 30 seconds per show. The parameters we have presented here could be used immediately by an end user so no heavy computation is required. The high and low pass filters h and l were fixed at 0Hz and 2000Hz respectively (effectively were not used but would be useful parameters for specialized implementations). W was fixed at 630 Seconds which we selected by taking the largest track present in the validation set with a 30 second margin added on top.

There are no directly comparable methods in the literature ready to be used for this task. We will construct a simple algorithm to test our algorithm against; the *naive algorithm*. This algorithm constructs indices that are evenly spaced apart across the show.

See Table 2 for the main results. We also provide results for the dataset pruned of any shows with tracks smaller than 180 seconds on Table 3. Ostensibly we would fail to find these tracks as we use 180 as the minimum track length parameter w for the DP algorithm. Having a high value for w allows us to perform robustly most of the time but suffer on the minority of shows that have smaller tracks included. For these pruned results we did not remove the shows used in the validation set.

Table 1. These are the parameter values that were obtained from the parameter search described in Section 9.

w	Minimum Track Length (DP)	180	Seconds
W	Maximum Track Length (DP)	617	Seconds
M	Tile Size	9	Seconds
b	Bandwidth Filter	5	Hz
l	Low Pass Filter	2000	Hz
h	High Pass Filter	0	Hz

Table 2. Main results. The accuracy rows show the mean of the absolute differences between the reconstructed tracks and the human indices (our test set). The thresholds indicate the percentage of reconstructed indices that fall within given time horizons centred around the actual indices. This is described in Section 3.

	Dynamic By Show			**Overall**	
	ASOT	TATW	MAGIC	Dynamic	Naive
Number Shows	101	89	98	288	288
60 Seconds (%)	92.3	96.9	97.9	95.7	42.1
30 Seconds (%)	74.9	90.4	89.8	85.1	22.0
20 Seconds (%)	63.7	82.0	74.4	73.4	14.9
10 Seconds (%)	55.7	70.9	53.2	59.9	7.6
5 Seconds (%)	43.9	51.5	32.9	42.8	4.0
3 Seconds (%)	29.7	35.3	21.2	28.7	2.6
1 Second (%)	11.6	13.9	8.3	11.3	0.9
Accuracy (Seconds)	49.4	40.1	26.5	38.6	112.2

Table 3. Results for the pruned set of shows (that do not contain tracks smaller than 180 seconds). The percentage figure given on the number of shows indicates how many were discarded from the prune. Performance on TATW and Magic Island are robustly improved. Magic Island achieved the improvement with a comparatively small prune of 7.4%.

	Dynamic By Show			**Overall**	
	ASOT	TATW	MAGIC	Dynamic	Naive
Number Shows	64 (42.3%)	61 (38.4%)	100 (7.4%)	225	288
60 Seconds (%)	93.4	98.2	98.9	96.8	42.1
30 Seconds (%)	75.7	92.3	90.8	86.2	22.0
20 Seconds (%)	63.5	84.0	74.9	74.1	14.9
10 Seconds (%)	56.3	72.8	53.2	60.8	7.6
5 Seconds (%)	44.0	52.8	32.6	43.2	4.0
3 Seconds (%)	30.0	36.5	20.9	29.1	2.6
1 Second (%)	12.1	15.2	8.6	12.0	0.9
Accuracy (Seconds)	32.3	14.9	13.3	20.2	112.2

10 Conclusion and Further Work

We believe our algorithm would be useful for segmenting DJ-mixed audio streams in batch mode. Our overall average is encouraging, taking into account the difficulty of the task at hand. The dissimilarity matrix we use is based solely on instrument features. The most pervasive elements in EDM are the percussion (the beats). We believe on balance that ignoring the percussive information was advantageous, because DJs use percussion primarily to blur boundaries between tracks. We tried to capture percussive based features and found that the transitions between tracks and indeed groups of tracks appeared as stronger self-similar regions in S than the actual tracks.

We would like to improve our cost function with one that has some domain knowledge, perhaps using a machine learning algorithm. Currently our cost function has a weakness that the relative similarity of regions within a song matters slightly, it should be independent. Let us consider the song structure {A,B,A}. The problem is that our cost (summing/normalizing the S square) would somewhat take into consideration the similarity of A and B. Anyone interested in optimizing the algorithm for a one specific radio show could consider modifying the cost function to introduce a parameter $\alpha \in [0, 1]$ for fine tuned control over the normalization bias placed on the length of songs; $C(f,t) = \frac{\sum_{i=f}^{t}\sum_{j=f}^{t} S(i,j)}{(t-f+1)^{\alpha}}$.

We would also like to implement some of the methods in the literature (which were mostly designed for scene analysis) to see if we outperform them. It would be tricky to get an exact comparison because we could not find a unsupervised deterministic algorithm which finds a fixed number of strictly contiguous clusters. We could however adapt existing algorithms to get a like for like comparison. We would like to evaluate the performance of J Theiler's contiguous K-means algorithm in particular [17] and also similar algorithms. We have the property of being deterministic but probabilistic methods should be explored. Theiler's algorithm would require some modification to work in this scenario because we require strictly contiguous clusters, not just a contiguity bias.

References

1. J. Foote, "Visualizing music and audio using self-similarity," in *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pp. 77–80, ACM, 1999.
2. J. Foote, "A similarity measure for automatic audio classification," in *Proc. AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, 1997.
3. J. Foote, "Automatic audio segmentation using a measure of audio novelty," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 1, pp. 452–455, IEEE, 2000.
4. J. T. Foote and M. L. Cooper, "Media segmentation using self-similarity decomposition," in *Electronic Imaging 2003*, pp. 167–175, International Society for Optics and Photonics, 2003.

5. J. Foote and M. Cooper, "Visualizing musical structure and rhythm via self-similarity," in *Proceedings of the 2001 International Computer Music Conference*, pp. 419–422, 2001.
6. M. M. Goodwin and J. Laroche, "Audio segmentation by feature-space clustering using linear discriminant analysis and dynamic programming," in *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pp. 131–134, IEEE, 2003.
7. M. M. Goodwin and J. Laroche, "A dynamic programming approach to audio segmentation and speech/music discrimination," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 4, pp. iv–309, IEEE, 2004.
8. G. Peeters, A. La Burthe, and X. Rodet, "Toward automatic music audio summary generation from signal analysis," in *Proc. of ISMIR*, pp. 94–100, 2002.
9. G. Peeters, "Deriving musical structures from signal analysis for music audio summary generation:"sequence" and "state" approach," *Computer Music Modeling and Retrieval*, pp. 169–185, 2004.
10. E. Peiszer, T. Lidy, and A. Rauber, "Automatic audio segmentation: Segment boundary and structure detection in popular music," *Proc. of LSAS*, 2008.
11. "Sox, the swiss army knife of sound processing programs.." http://sox.sourceforge.net/.
12. M. Lindgren, "Cuenation, website for edm community to share track time metadata http://cuenation.com/."
13. H. Nyquist, "Certain topics in telegraph transmission theory," *American Institute of Electrical Engineers, Transactions of the*, vol. 47, no. 2, pp. 617–644, 1928.
14. M. Frigo and S. G. Johnson, "The fftw web page," 2004.
15. G. Tzanetakis and P. Cook, "Multifeature audio segmentation for browsing and annotation," in *Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop on*, pp. 103–106, IEEE, 1999.
16. G. Tzanetakis and F. Cook, "A framework for audio analysis based on classification and temporal segmentation," in *EUROMICRO Conference, 1999. Proceedings. 25th*, vol. 2, pp. 61–67, IEEE, 1999.
17. J. P. Theiler and G. Gisler, "Contiguity-enhanced k-means clustering algorithm for unsupervised multispectral image segmentation," in *Optical Science, Engineering and Instrumentation'97*, pp. 108–118, International Society for Optics and Photonics, 1997.