Segmenting electronic dance music streams based on self-similarity

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

March 17, 2014

Abstract

We describe an unsupervised, deterministic algorithm for segmenting DJ-mixed Electronic Dance Music (EDM) streams (for example; pod-casts, radio shows, live events) into their respective tracks. We attempt to reconstruct boundaries as close as possible to what a human domain expert would create in respect of the same task. 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 to optimally segment a cost matrix derived from a self-similarity matrix. The similarity matrix is based on the cosines of a time series of kernel-transformed Fourier based features. Our method is applied to EDM streams. Its formulation incorporates long-term self similarity as a first class concept combined with dynamic programming and it is qualitatively assessed on a large corpus of long streams that have been labelled by a domain expert.

Keywords. music,segmentation,DJ mix,dynamic programming

1 Introduction

Electronic Dance Music tracks are usually mixed by a disc jockey (DJ). For this reason EDM music streams are quite unique compared to other genres of music. Mixing is the *modus operandi* in electronic music. We first transform the audio file into a time series of features discretized into adjacent tiles and transform them into a domain where some pairs from the same track would be distinguishable by their cosine. Our features are based on a Fourier transform with convolution filtering to accentuate prominent instruments and self-similarity within tracks. We create a similarity matrix from these cosines and then derive cost matrices showing the costs of fitting a track at a given time with a given length. 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 transients or a novelty function. 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 configurable time horizon. Our method does not require any training or tenuous heuristics to perform well.

The purpose of this algorithm is to reconstruct optimal 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 quite reliably as the variable of track lengths is Gaussian (see Figure 1).

One of the interesting features of audio is that you *can not scrub through it and get an overview in the same way you can with video.* Audio has a reduced *contextual continuum* when the user skips through it perhaps due to the lack of redundant, persistent scene-setting information or indeed a psychological reason. Even in video applications, discovery, context and scrubbing is an active area of research [1]. Time index metadata would allow click through monetisation, and allow improved use-case scenarios (for example publishing track names to social networks, information discovery and retrieval). Capturing metadata in audio is a time consuming and error-prone process. Tzanetakis found in [2] that it took users on average 2 hours to segment 10 minutes of audio using standard tools. While not directly relevant we might glean from those findings that there is a strong motivation to automate this process. We have also noticed that while the error variable of the human captured time metadata that we were supplied with appears Gaussian, there is a noticeable deteriorate towards the end of the shows or in relation to the complexity of the mixing by the disc jockey.

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 [3, 4, 5, 6, 7] 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 [8, 9]. 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 [10, 11] 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 TODO

2 Corpus

We have been supplied with several broadcasts from three popular radio shows. These are: Magic Island, by Roger Shah (106 shows); A State of Trance with Armin Van Buuren (110 shows); and Trance Around The World with Above and Beyond (88 shows) (Total 304 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 [12] in the literature. The music remains uninterrupted af-

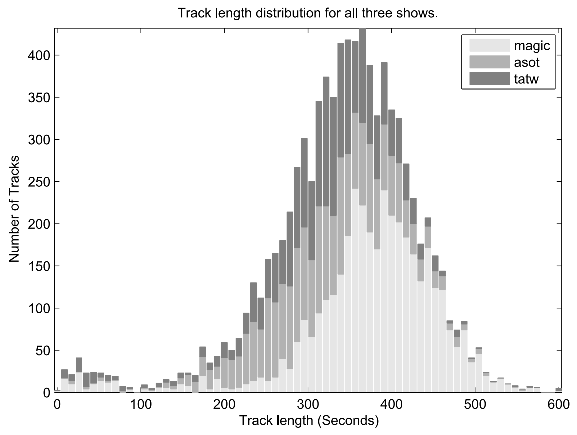


Figure 1: The track length distribution for all three radio shows. The *bump* of short tracks (less than 3 minutes) are often introductions or otherwise extraneous.

ter 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 "Sound eXchange"[1] program to do this. These shows are all 2 hours long. The overall average track length is 5 and a half minutes (slightly less for Magic Island) 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 technically convoluted mixing.

There is already a large community of people interested in getting track metadata for DJ sets. "CueNation"[2] 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 time metadata (indices) and radio shows provided to us and hand captured by *Dennis Goncharov*; a domain expert and one of the principal contributors to *CueNation*. One of the significant problems with this task is that there is an (apparently Gaussian) error variable attached to the human captured indices themselves. On some tracks, it is unclear where to place the optimal index and when analysing our results, we have noticed some obvious human errors. Regrettably, there is no obvious way of quantifying this. Many of the cue sheet authors themselves reject the idea of automating the task citing the poor precision of any such result (they often place indices on the exact MP3 frame). However this sentiment seems misplaced given that they make blatant mistakes or that it is a matter of opinion where to place the track and some consistent method would be preferential. A potential outcome of our method would be a way to assist them with initial placements. Our results demonstrate that it is indeed possible to automate this task and that while there is some uncertainty attached to the optimal placement, it is still largely predictable.

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 cue sheet 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.

The most obvious and pervasive element in dance music is the percussion (the beats). We believe on balance that ignoring the percussive information is

[1] `http://sox.sourceforge.net`

[2] `http://www.cuenation.com`

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. It is clear from our research that it is the boundaries between instruments and harmonic content that reveals track boundaries, not percussion or rhythm. When we tried implementing a rhythm feature extractor by looking at a transformed autocorrelation of the time domain tiles, it was ineffective because it is the rhythm more than the instruments that is matched between adjacent tracks by the DJs. Some DJs do mix harmonically too but this preys on human hearing and perception. An algorithm capturing the harmonic information would most likely be able to easily distinguish two harmonically compatible tracks. Admittedly more sophisticated rhythm detection than we tried would likely still work to some extent and should be explored further.

3 Evaluation Criteria

It is challenging to quantify the performance of our method because if we misplace any tracks, it may have a cascade effect. For example if we place one track too many early on in a show, many of the subsequent tracks may be correctly placed but out of alignment.

We perform three types of evaluation: '*average*', '*precision*', and '*thresholds*'.

The *average* (in seconds) given as

$$\frac{1}{|P|}\sum_{i=1}^{|P|}|P_i - A_i|$$

is quite simply the mean absolute difference between constructed and actual indices (P is constructed indices, and A is the human indices). Naturally this metric will have a reduced meaning because although it depends on the real accuracy it also depends on the amount of misplacements and where in the track those misplacements were. The uncertainty variable attached to the misplacements is wide (in relation to the size of the shows and the number of tracks) but does regularise over the course of a large dataset.

The *precision* metric denoted by

$$\hat{H}(P, A) \leftarrow \frac{1}{|P|}\sum_{i=1}^{|P|}\text{Sort}\left(|P - A_i|\right)_1$$

for all takes the mean of the absolute difference between each prediction and the nearest index.

The thresholds metric is the percentage of matched tracks within different intervals of time $\{60, 30, 20, 10, 5, 3, 1\}$, in *seconds* as a margin around any of the track indices we have been given. This metric is also invariant to alignment and is provided for comparison with our other paper [13].

4 Preprocessing

The dataset had some outliers that may have slightly distorted the analysis of our method. Many of the tracks in our data set were in fact not tracks at all but rather introductions or voice-overs. Almost all of these outlier tracks were short in length. These are quite clearly visible on the distribution of track lengths on Figure 1. To ameliorate the situation we simply removed any tracks that were shorter than 180 seconds. We also removed any end tracks that were shorter than 240 seconds as very often the end tracks on a radio show contain some strange elements. This required some manipulation of the cue sheets and audio files. The offending segments of the audio files are chopped out, and the cue sheets are re-flowed so that the time indices point to the correct location in the file.

The algorithm still performs similarly in most cases when removing just these indices and leaving the audio intact underneath.

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).

5 Feature Extraction

We used SoX (see Sect. 2) 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 [14] 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

$$F(x_k) = X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}$$

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 [15] that we used to perform this computation 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 [16]).

The Gaussian first derivative filter is defined as

$$-\frac{2G}{B^2} e^{-\frac{G^2}{B^2}}$$

where

$$G = \{-\lfloor 2B \rfloor, \lfloor -2B + 1 \rfloor, \ldots, \lfloor 2B \rfloor\},$$

and

$$B = b\left(\frac{N}{R}\right).$$

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

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 [2]. We construct a disimilarity matrix of cosines S from $D \times D^\top$ (dot products).

See Figure 2 for an illustration of S.

6 Cost Matrices

We now have a dissimilarity matrix $S(i,j)$ as described in Section 5.

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

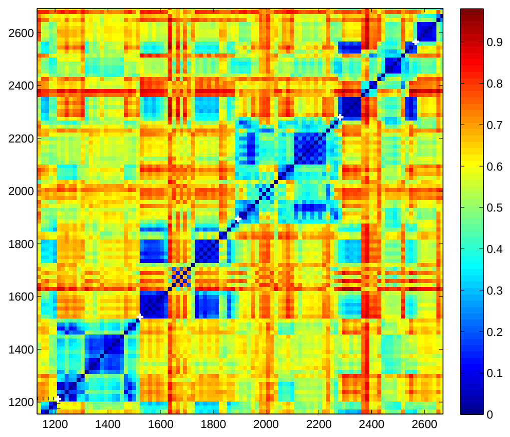


Figure 2: An illustration of what the dissimilarity matrix $S(i,j)$ looks like

Intuitively, most features within the same track are similar, while pairs of tiles that do not belong to the same track are significantly more dissimilar.

We require a cost matrix $C(f,t)$ that describes the cost of placing a track of length $t-f+1$ at f. After analysing the data set, we designed three cost matrices that exploited themes present in the possible track placements in S. These were; summation, contiguity and symmetry.

6.1 Summation

The most obvious strategy of all is to sum up the relevant elements in S for each candidate track from tile f through tile t. 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 with a regularization exponent ω

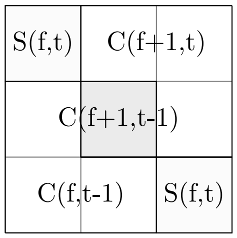
$$C(f,t) \;=\; \frac{\sum_{i=f}^{t}\sum_{j=f}^{t} S(i,j)}{(t-f+1)^{\omega}}.$$

As a first step, we pre-compute C for each $1 \le f \le t \le T$. 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 $f+1 \le t-1$)

$$\begin{aligned}\tilde{C}(f,t) \;&=\; \tilde{C}(f+1,t) + \tilde{C}(f,t-1) \\ &\quad - \tilde{C}(f+1,t-1) + S(f,t) + S(t,f).\end{aligned}$$

Note that the track size normalization step can be done independently of the DP procedure.

Here is a visualisation of the update routine in the domain of S.



One caveat is that this method only looks for disincentives for track placement. This might fall down when you have self-similar regions in tracks that are completely dissimilar to the rest of the track. So we define the *sum cost incentive bias* $\mu \in [0,1]$ to describe the ratio of incentive verses disincentive (with 0.5 meaning equal incentive and disincentive). To enable this work we use $\tilde{S} = S - \mu$ on the computation.

6.2 Contiguity

Let

$$\begin{aligned}\Omega(f,t) \mapsto \{ \\ & S(f,f), S(f+1,f+1), S(f+2,f+2)\ldots \\ & S(f+1,f), S(f+2,f+1), S(f+3,f+2)\ldots \\ & \ldots \\ & S(t,t)\ \}\end{aligned}$$

return a vector corresponding to concatenated diagonals in one half of the symmetric matrix $S(f\ldots t, f\ldots t)$. We are interested in rewarding adjacent pairs that have both have low value (less than *contiguity threshold* ψ) and punishing pairs that both have a high value (more than $1-\psi$). Scores are weighted by the contiguity incentive bias ϱ.

Therefore, let

$$\begin{aligned}&\hat{\Upsilon}(f,t) \\ &\mapsto \sum_{i=2}^{|\Omega(f,t)|} \begin{cases} \theta_{i-1} < \psi \wedge \theta_i < \psi & : \xi\varrho \\ \theta_{i-1} > 1-\psi \wedge \theta_i > 1-\psi & : \xi(1-\varrho) \end{cases}\end{aligned}$$

denote the contiguity cost matrix ($\theta = \Omega(f,t)$) where $\xi = \frac{1}{2}(\theta_{i-1} + \theta_i)$.

$\Upsilon(f,t)$ is normalized by the track length regularized by exponent λ.

$$\hat{\Upsilon}(f,t) = \frac{\hat{\Upsilon}(f,t)}{(t-f+1)^{\lambda}}$$

$$\Upsilon(f,t) = \frac{(\Upsilon(f,t) - \min\Upsilon)}{\max\Upsilon - \min\Upsilon}$$

6.3 Symmetry

Let

$$\begin{aligned}\Lambda(f,t,d) \leftarrow \{&S(f+d-1,f), S(f+d,f+1), S(f \\ &+d+1,f+2),\ldots,S(t,t-d-1)\ \}\end{aligned}$$

take the diagonal d from $S(f\ldots t, f\ldots t)$. For each diagonal in $S(f\ldots t, f\ldots t)$ we want to compare each element against its mirror counterpart. 'symmetric' pairs that are both lower than *symmetry threshold* ν confer an advantage (average value of both elements normalized by the track length and weighted by the *symmetry incentive balance* ς), and vice versa.

Let symmetry cost matrix

$$\begin{aligned}&\hat{\Gamma}(f,t) \\ &\mapsto \sum_{i=1}^{|\Lambda(f,t,d)|} \begin{cases} \kappa_i < \nu \wedge \rho_i < \nu & : \alpha\varsigma \\ \kappa_i > 1-\nu \wedge \rho_i > 1-\nu & : \alpha(1-\varsigma) \end{cases}\end{aligned}$$

for all $d = 1,2,\ldots,t-f+1$ where $\alpha = (t-f+1)\frac{1}{2}(\kappa_i + \rho_i)$, $\kappa = \Lambda(f,t,i)$ and $\rho_i = \Lambda(f,t,i)_x$ for $x = |\Lambda(f,t,i)|, |\Lambda(f,t,i)|-1,\ldots,1$.

$\Gamma(f,t)$ is normalized by the track length regularized by exponent ζ and scaled onto the $[0,1]$ interval.

$$\hat{\Gamma}(f,t) = \frac{\hat{\Gamma}(f,t)}{(t-f+1)^{\zeta}}$$

$$\Gamma(f,t) = \frac{(\Gamma(f,t) - \min\Gamma)}{\max\Gamma - \min\Gamma}$$

6.4 Gaussian

Let

$$G(\varpi, N)_{tw} = e^{-\frac{1}{2}\frac{\varpi n}{\frac{1}{2}W}^{2}}$$

for all $n = 1,2,\ldots,W$ denote the Gaussian matrix cost function of $N \times W$. $G(\varpi, N)$ is time-independent and every row is the same. We will use this cost function for regularising the other three and for use on its own for comparison against a 'naive' approach to the problem. Increasing values of ϖ will tighten up the Gaussian although for simplicity we will just assume it has a value of 1.

7 Mixing Cost Functions

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 $W = 60\times 15$.

Let $N(T, W, w, m)$ be the number of segmentations with time T (in tiles),

We can write the recursive relation

$$N(T, W, w, m) = \sum N(t_m - 1, W, w, m-1)$$

, where the sum is taken over t_m such that

$$\begin{aligned} t_m &\leq T - w + 1 & t_m &\geq T - W + 1 \\ t_m &\geq (m-1)w + 1 & t_m &\leq (m-1)W + 1 \end{aligned}$$

The first two inequalities mean that the length of the last track is within an acceptable boundary between w and W. The last two inequalities mean that the lengths of the first $m-1$ tracks are within the same boundaries.

We calculated the value of $N(7000, 60\times 15, 190, 25)$ and got 5.20×10^{56} which is still infeasible to compute with brute force.

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 Posterior Marginal of Song Boundary

Fix a learning rate η, and fix T and M. Let

$$P(j, s) \;=\; \frac{\sum_{\boldsymbol{t}\in\mathbb{S}_m^T : t_j = s} e^{-\eta\,\ell(\boldsymbol{t})}}{\sum_{\boldsymbol{t}\in\mathbb{S}_m^T} e^{-\eta\,\ell(\boldsymbol{t})}}$$

That is, $P(j, s)$ is the "posterior probability" that song j starts at time s.

To compute $P(j, s)$, we need an extended notion of segmentation. We call $\boldsymbol{t}$ a $m/F : T$ segmentation if

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

Let $\mathbb{S}_m^{F:T}$ be the set of all $m/F - T$-segmentations.

We have

$$\sum_{\boldsymbol{t}\in\mathbb{S}_m^T:t_j=s} e^{-\eta\,\ell(\boldsymbol{t})} \;=\; \sum_{\substack{\boldsymbol{t}\in\mathbb{S}_{j-1}^{s-1},\\ \boldsymbol{t}'\in\mathbb{S}_{m-j+1}^{s:T}}} e^{-\eta(\ell(\boldsymbol{t})+\ell(\boldsymbol{t}'))} \;=\; \left(\sum_{\boldsymbol{t}\in\mathbb{S}_{j-1}^{s-1}} e^{-\eta\,\ell(\boldsymbol{t})}\right)\left(\sum_{\boldsymbol{t}\in\mathbb{S}_{m-j+1}^{s:T}} e^{-\eta\,\ell(\boldsymbol{t})}\right)$$

which upon abbreviating

$$\mathcal{H}_m^t \;=\; \sum_{\boldsymbol{t}\in\mathbb{S}_m^t} e^{-\eta\,\ell(\boldsymbol{t})} \qquad \mathcal{T}_m^f \;=\; \sum_{\boldsymbol{t}\in\mathbb{S}_m^{f:T}} e^{-\eta\,\ell(\boldsymbol{t})}$$

means that we can write

$$P(j,s) \;=\; \frac{\mathcal{H}_{j-1}^{s-1}\cdot\mathcal{T}_{m-j+1}^{s}}{\mathcal{H}_m^T}.$$

So it suffices to compute $\mathcal{H}_m^t$ and $\mathcal{T}_m^t$ for all relevant t and m. We use

$$\mathcal{H}_1^t \;=\; e^{-\eta C(1,t)} \qquad \mathcal{T}_1^f \;=\; e^{-\eta C(f,T-f+1)}$$

and for $m \geq 2$

$$\begin{aligned}
\mathcal{H}_m^t &= \sum_{t_m}\sum_{\boldsymbol{t}\in\mathbb{S}_{m-1}^{t_m-1}} e^{-\eta(\ell(\boldsymbol{t})+C(t_m,t-t_m+1))}\\
&= \sum_{t_m} e^{-\eta C(t_m,t-t_m+1)} \sum_{\boldsymbol{t}\in\mathbb{S}_{m-1}^{t_m-1}} e^{-\eta\,\ell(\boldsymbol{t})}\\
&= \sum_{t_m} e^{-\eta C(t_m,t-t_m+1)}\mathcal{H}_{m-1}^{t_m-1}\\
\mathcal{T}_m^f &= \sum_{t_2}\sum_{\boldsymbol{t}\in\mathbb{S}_{m-1}^{t_2:T}} e^{-\eta(C(f,t_2-f)+\ell(\boldsymbol{t}))}\\
&= \sum_{t_2} e^{-\eta C(f,t_2-f)} \sum_{\boldsymbol{t}\in\mathbb{S}_{m-1}^{t_2:T}} e^{-\eta\,\ell(\boldsymbol{t})}\\
&= \sum_{t_2} e^{-\eta C(f,t_2-f)}\mathcal{T}_{m-1}^{t_2}
\end{aligned}$$

10 Posterior Marginal of Song Position

Fix a learning rate η, and fix T and M. Let

$$P(j,s,f) \;=\; \frac{\sum_{\boldsymbol{t}\in\mathbb{S}_m^T:t_j=s\wedge t_{j+1}-1=f} e^{-\eta\,\ell(\boldsymbol{t})}}{\sum_{\boldsymbol{t}\in\mathbb{S}_m^T} e^{-\eta\,\ell(\boldsymbol{t})}}$$

That is, $P(j,s,f)$ is the "posterior probability" that song j starts at time s and finishes at time f. In the same vein as the last section, we now get

$$P(j,s,f) \;=\; \frac{\mathcal{H}_{j-1}^{s-1}\cdot e^{-\eta C(s,f-s+1)}\cdot\mathcal{T}_{m-j}^{f+1}}{\mathcal{H}_m^T}.$$

11 Materials

All of the code presented in this paper with a small working test set is available on GitHub [3]. The large data set ($\approx$ 100GB) we received from Dennis can be made available on request.

12 Methodology

TODO: parameters,

13 Results

14 Summary

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.

TODO: percussive features We captured the percussive features by taking the cross correlation of the samples in the time domain, and using the same convolution filter.

[3] `github.com/ecsplendid/DanceMusicSegmentation`

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.

15 Acknowledgements

We would like to thank Mikael Lindgren and Dennis Goncharov from cuenation[4] for their help explaining how they make cues and for providing the data set to test the algorithm on.

References

[1] J. Matejka, T. Grossman, and G. Fitzmaurice, "Swifter: Improved online video scrubbing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, (New York, NY, USA), pp. 1159–1168, ACM, 2013.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] E. Peiszer, T. Lidy, and A. Rauber, "Automatic audio segmentation: Segment boundary

[4] `http://www.cuenation.com`

and structure detection in popular music," *Proc. of LSAS*, 2008.

[13] T. Scarfe, W. M. Koolen, and Y. Kalnishkan, "A long-range self-similarity approach to segmenting dj mixed music streams," in *Artificial Intelligence Applications and Innovations*, pp. 235–244, Springer, 2013.

[14] H. Nyquist, "Certain topics in telegraph transmission theory," *American Institute of Electrical Engineers, Transactions of the*, vol. 47, no. 2, pp. 617–644, 1928.

[15] M. Frigo and S. G. Johnson, "The fftw web page," 2004.

[16] 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.

[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.

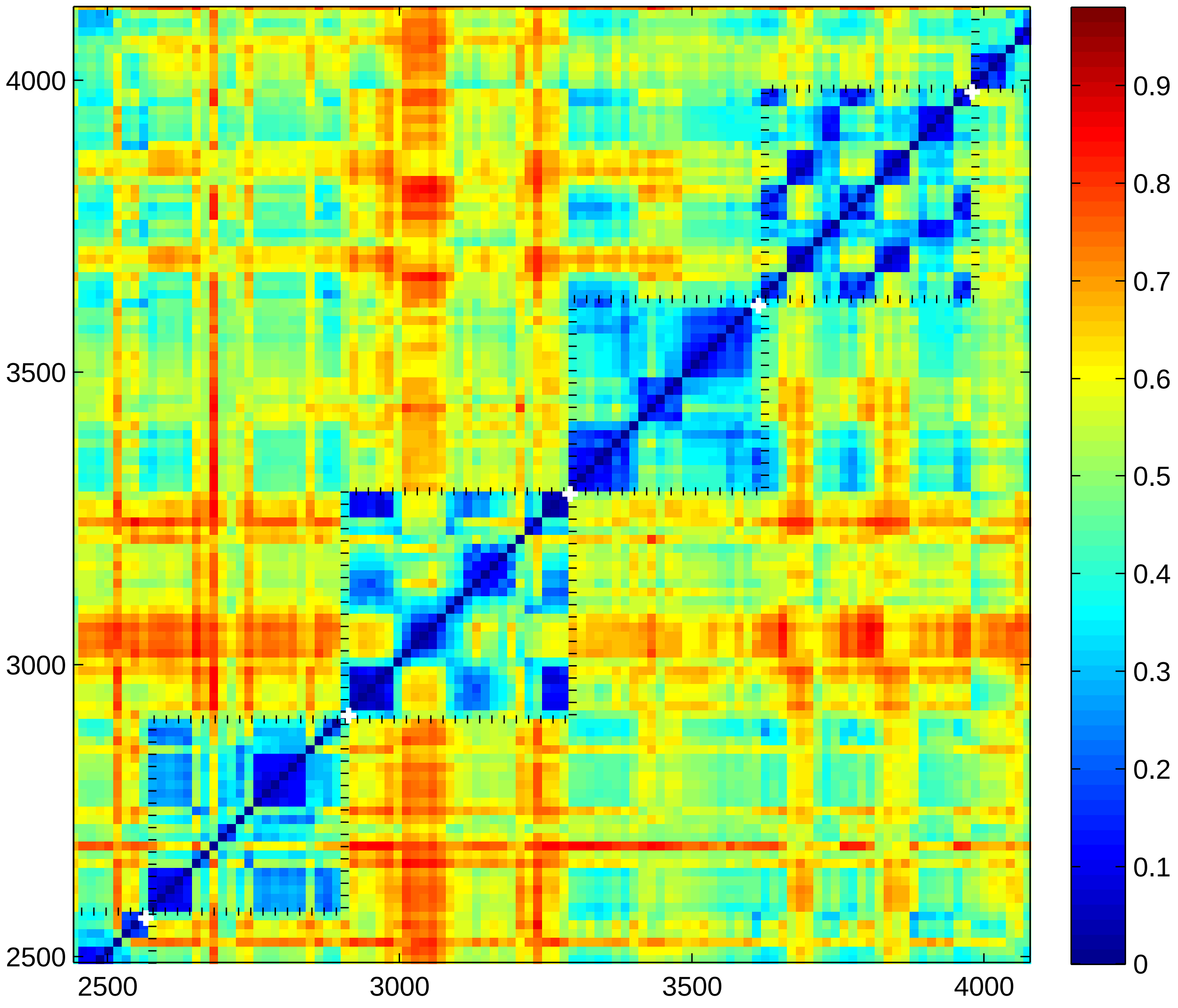


Figure 3: An illustration of what the dissimilarity matrix $S(i, j)$ looks like with the reconstructed indexes superimposed. White crosses indicate the human captured indices, and black dotted lines are the reconstructed indices. Note the error where one track has an end segment which is unlike the rest.

Sym	Cont	Sum	Gauss	Mean	Heuristic	Shift	60s(%)	30s(%)	20s(%)	10s(%)	5s(%)	1s(%)
						Tile Size 20s, $\mu = 0.5$						
1	1	1	$\frac{1}{2}$	21.51	15.12	0.32	97.70	89.22	77.41	47.48	25.71	15.74
1	0	0	$\frac{1}{2}$	22.25	15.91	1.07	97.78	88.46	73.58	43.72	23.54	14.54
0	1	0	$\frac{1}{2}$	21.82	16.58	-0.73	96.71	85.59	74.29	46.94	25.72	15.47
0	0	1	$\frac{1}{2}$	22.21	15.43	0.75	97.35	87.73	77.19	48.84	26.33	16.14
0	0	0	1	104.8	76.18	-38.4	43.57	22.58	15.39	8.19	4.55	2.75
1	1	1	**0**	40.94	17.90	-1.20	96.45	87.62	76.12	46.75	25.19	15.44
1	0	0	**0**	40.16	18.39	-2.32	96.69	86.96	72.19	42.72	23.30	14.34
0	1	0	**0**	44.62	19.72	0.91	95.31	83.79	72.70	45.69	24.76	14.99
0	0	1	**0**	42.66	18.78	-0.08	95.84	86.04	75.22	47.16	25.43	15.47
						Tile Size 10s, $\mu = 0.5$						
1	1	1	$\frac{1}{2}$	22.03	14.79	2.18	97.28	88.19	78.10	54.24	29.99	18.72
1	0	0	$\frac{1}{2}$	21.32	15.16	1.67	97.45	88.12	77.21	51.10	27.33	16.86
0	1	0	$\frac{1}{2}$	22.29	15.81	1.38	96.74	86.06	75.00	52.73	29.20	18.06
0	0	1	$\frac{1}{2}$	21.97	14.94	0.32	97.09	87.55	77.31	55.08	30.02	18.57
0	0	0	1	99.35	75.54	-20.4	43.65	22.90	15.81	7.90	4.12	2.55
						Tile Size 5, $\mu = 0.4$						
0	0	1	$\frac{1}{2}$	20.12	14.59	0.34	97.22	87.48	77.43	57.52	30.57	18.84
						Tile Size 2, $\mu = 0.4$						
0	0	1	$\frac{1}{2}$	19.27	14.53	-1.01	97.18	87.67	76.75	58.04	30.63	18.87

Table 1: Main results.