Segmentation of electronic dance music

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

September 25, 2014

Abstract

We consider the problem of segmenting DJ-mixed dance music recordings (pod-casts, radio shows, live events) into their original sequence of contiguous tracks. We present an algorithm to reconstruct a fixed number of segment boundaries as close as possible to what a human domain expert would create in respect of the same task. As the number of segments is known in advance we do not have to rely on local points-of-change heuristics prevalent in common segmentation algorithms. We also adapt the method to estimate the number of tracks in a recording and compare our method with others. The goal of DJ-mixing is to render track boundaries invisible from human perception.

Contiguous segmentation[1] is performed on a self-similarity matrix which is derived from normalized cosines of various cost matrices which have themselves been derived from a time-series of Fourier based spectral features. The cost matrices introduced in this paper introduce notions of self-similarity, symmetry, contiguity and evolution in respect of time. The method could be used on other segmentation tasks where the number of segments is known in advance, and the audio based features could be replaced with ones from another domain.

Our work incorporates self-similarity over a soft time horizon with a fixed upper boundary and is quantitatively assessed on a large corpus of radio show recordings which 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 disc jockey (DJ). For this reason EDM music streams are 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 pairs from the same track would be distinguishable by their cosine. Our features are based on a Fourier transformation 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 to create the cost matrices and again to perform the most economical segmentation of the cost matrices to fit a predetermined number of tracks. Dynamic programming means solutions to a problem are described in terms of overlapping sub-solutions to achieve a significant improvement in time complexity and therefore execution time.

The contiguous-segmentation differs from the standard clustering problem in that the clusters arrive sequentially and are homogeneous. Our work considers

[1] blabla

a contiguous-segmentation that is globally-optimised and based upon the self-similarity of segments, avoiding transient heuristics typically used in local search methods. Music and mixes of music have the property that they are made up of recursively repeating self-similar regions. Our method does not require any training or tenuous heuristics to perform well. The distinguishing feature of our problem domain is that the number of segments is known a priori but the segmentation boundaries are not known, or ambiguous and subjective. However, computing the best solution is desirable.

The purpose of the algorithm is to reconstruct optimal boundaries given a fixed number of tracks known a priori (the track listing is usually published by the DJ). 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 were not known in advance, the number of tracks could be estimated as the variable of track lengths is Gaussian (see Figure 1). However the algorithm is optimised for knowing the number of tracks and may not perform as well as other contiguous-segmentation algorithms otherwise. It may be possible to infer from our results model if the correct number of tracks was used as a parameter. This would open up new applications, but is an area for future research.

One of the interesting features of audio is that you *cannot 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 are an active area of research [1]. Time index meta-data would allow click through monetisation, and allow improved use-case scenarios (for example publishing track names to social networks, information discovery and retrieval). Capturing meta-data in audio is a time consuming and error-prone process. Tzanetakis [2] found 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.

DJs always match the speed or beats per minute (BPM) 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).

1.1 Literature Review

Audio segmentation in the literature is colloquially implemented in the context of structural analysis. Music structure denotes the organization of a composition by its melody, harmony, timbre and rhythm. Repetitions, transformations and evolutions of music structure also contribute to its identity and it is this semantic information that structural analysis algorithms aim to extract from music. An example structure for a song might be "Common" → "Verse" → "Chorus" → "Common". Speaker diarization is another example of structural analysis

Segmentation in the context of structural analysis has been thus far been concerned with creating a novelty function to find points-of-change using distance-based metrics, rather than trying to find a fixed number of segments. Heuristics with hard decision boundaries have been used to find the best change points, for example Tzanetakis [2] used first-order derivatives of a time series of audio features.

The use of a similarity matrix to visualize and analyse local time dependencies (then called "Recurrence Plots") was first proposed by Eckmann[3].

J. Foote et al [4, 5, 6, 7, 8] were the first to use self-similarity matrices to visualise and exploit time dependencies in music data. Foote evaluated a Gaussian 'tapered' checker board kernel along the diagonal of a music self similarity matrix to create a 1-dimensional novelty function that had the notion of self-similarity over a fixed time horizon. The kernel was 'tapered' down to zero on the top right and bottom left edges to reduce edge effects. The dynamic program in this paper allows any self-similarity time horizon up to a fixed limit (Foote's work had a fixed kernel size).

Goodwin et al. used a dynamic program for segmentation [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 normalized. Afterwards, Goodwin formulated the problem into one of finding the globally minimum cost path through a state graph ('cluster space trajectory') modelling local and transition costs. Goodwin already demonstrated in [10] that novelty peaks often exist within segments, not only on the boundary of segments and took the approach of modelling all possible sequential transitions between all possible clusters.

A potential drawback of the approach by Goodwin and all other approaches in scene analysis segmentation that we are aware of; is that they are somewhat local methods that focus on points-of-change rather than optimizing for the best possible results for a fixed number of segments.

All contiguous segmentation algorithms that we looked at do not know a priori how many segments to find. In almost all applications, the number of known segments derives from knowledge of where boundaries are. This is the reason why the algorithm presented in this paper is unique.

[**?**]

A distinguishing feature of our approach is that we evaluate how well we are doing compared to humans in respect of the same task. We compare our reconstructed indices to the ones created by a human domain expert and the algorithm itself is optimised for the domain of mixed music.

In the coming sections we describe the corpus (see Section 2), human accuracy (see Section 3), the evaluation criteria (see Section 4), how we pre-process the data (see Section 5.1), how we perform feature extraction (see Section 5.2.1), designing the cost matrices using observed phenomena in the domain (see Section 5.2.2), computing the best segmentation (see Section 6), discussion of confidence intervals (see Section 7), our methodology (see Section 8), materials used (see Section 10), results (see Section 11) and finally the summary (see Section 12).

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.

The music is 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 "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 (see Figure 1)) 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, so it is fair to say that the boundary "complexity" increases during the guest mix and is at least not fixed throughout the shows.

We believe this corpus is one of the largest of its kind used in the literature going on the comparative table of segmentation corpora listed by Peiszer et al in their literature review of audio segmentation [11]. Peiszer used a corpus of 109 songs by the Beatles. To date

foote2003media foote2000automatic

There is already a large community of people interested in getting track/time meta-data 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 meta-data (indices) for a media file.

We had our time meta-data (indices) and radio shows provided to us and hand captured by *Denis Goncharov*; a domain expert and one of the principal contributors to *CueNation*. One of the significant

[1]`http://sox.sourceforge.net`
[2]`http://www.cuenation.com`

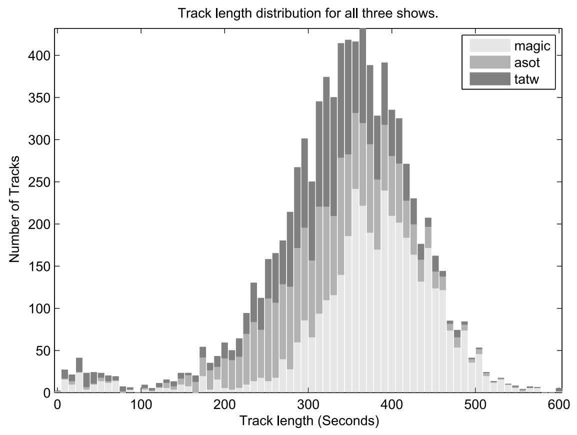


Figure 1: The track length distribution for all three radio shows. The *bump* of short tracks (less than 3 minutes) is often introductions or otherwise extraneous.

problems with this task is that there is a random error variable associated with 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 clear 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 frequently make mistakes or that it is a matter of opinion where to place the track and some consistent method may be preferential. A potential outcome of our method could be an assistance mechanism to help 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.

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

> Trance music is made in slices of 8 bars (1 bar is 4 beats. At 135 beats per minute, 8 bars is $\frac{60}{135}(4 \cdot 8) \approx 14.8$ sec). Trance music tends to be around 130-135 BPM. 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 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 than the actual tracks. The percussive feature extractor transformed the autocorrelation of the audio samples in the time domain tiles, and compared the cosine of their absolute values. It was reasonably clear from that research that track boundaries are revealed with less uncertainty between instruments and harmonic content. However. We do not rule out looking at percussive features again in future research because we are currently ignoring potentially useful features.

Some DJs "mix harmonically" (by matching instruments as opposed to percussion) but this preys on human hearing and perception. An algorithm capturing the harmonic information would most likely be able to distinguish two harmonically compatible tracks.

3 Human Accuracy

We did some analysis on how accurate the humans themselves are at creating indices. In the absence of a perfect data set our analysis instead hinged on the amount to which the humans disagreed with each other aggregated over a large amount of historical data. Mikael Lindgren was kind enough to send us a dump of his cuesheet database to experiment with. As ASOT is such a popular show there were many independently captured cuesheets to compare against for all of the historical shows. We selected all the shows having at least 3 distinct cuesheets (not copies or shifted/misaligned copies of each other) and such that all the cuesheets had the same number of tracks. The first track was ignored (as it was always 0 seconds). We ended up with 115 shows with 3 authors. 65 shows with 4 authors and 30 shows with 5 authors. We generated a histogram of distances from the median time for each track, for each cuesheet and assumed values greater than 100 seconds or less than -100 seconds were outliers. The standard deviation of the 'human disagreement' variable is 9.13 seconds. See Figure 2 for an illustration. So at this stage it does not seem feasible for us to achieve a higher accuracy when we are evaluating against a method which is intrinsically error prone.

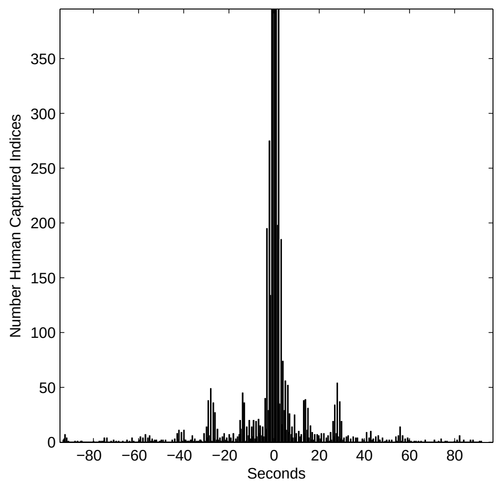


Figure 2: Illustration of the 'human disagreement' random variable (zoomed in at the bottom), standard deviation 9.13 seconds. Peaks are visible at intervals of 8 bars ($\approx$ 14.8 seconds) which corroborates the analysis from Denis Goncharov in Section 2.

4 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 detected but placed out of alignment.

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

The *average* (in seconds) given as

$$\hat{A}(P,A) = \frac{1}{|P|}\sum_{i=1}^{|P|} |P_i - A_i|$$

is 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 in some sense because although it depends on the real accuracy it also depends on the number 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. This metric is the most meaningful for capturing the overall *robustness* of the method.

The *heuristic precision* metric denoted by

$$\hat{H}(P,A) = \text{Median}\left(\text{Sort}\left(|P_i - A_{1,2,\ldots,|P|}|\right)_1\right)$$

for all $i = 1, 2, \ldots, |A|$ takes the median of the absolute differences between each prediction in P and the nearest actual index in A. This is the most meaningful single metric of *accuracy.* It is largely invariant to misplacements. There are typically only a few misplacements, if any. The median gives us a fair idea of what the accuracy would have been without the misplacements.

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 [12].

Note that our results will contain the mean average of these metrics (captured for each show) across the dataset being evaluated.

5 Data Handling

5.1 Preprocessing

The dataset had some outliers that may have slightly distorted the analysis of our method. Many of the "tracks" in our data set (of indices) 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 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 strange elements (for example voice-overs, interviews, show-related 'jingles'). This required some manipulation of the cue-sheets and audio files. The undesired segments of the audio files were chopped out, and the cue-sheets were reflowed so that the time indices point to the correct location in the file.

The algorithm still performs similarly when removing just these indices and leaving the audio intact underneath, so it would not significantly affect any real-world implementation.

For those wishing to use this algorithm in practice with pre-recorded shows; the introductions at the start of the shows are often fixed length or at least predictable so error would be small on average.

5.2 Feature Extraction

5.2.1 Music

We used SoX (see Sect. 2) to downsample the shows to 4000Hz. We are not particularly interested in frequencies above around and above 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 transform the tiles into the frequency domain using the discrete Fourier transform

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

which 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 [14] that we used to perform this computation operates significantly faster when N is a power of 2 so we zero pad the input to the next power of 2. We denote the tile width by M in seconds (an algorithm parameter). Note that

$$N = \frac{L}{M}$$

denotes the tile size in samples (length of show in samples over the tile size). Let

$$T = \left\lfloor \frac{L}{\tilde{M}} \right\rfloor$$

be the total number of tiles, and

$$\tilde{M} = \frac{L}{N}$$

the tile width in samples. Because we are passing real values into the $F(x_k)$, the second half of the result is a rotational copy of the first half.

Show samples are collated into a time series Q_i^y $(T \times N)$ of contiguous, non-overlapping, adjacent *tiles* of equal size where $i = 1, 2, \ldots, T$. Samples at the end of the show that do not fill a complete tile get discarded. The affect of this is increasingly negligible with decreasing tile size. Since we zero-pad N to the next power of two, this also decreases the affect.

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. Let $\hat{h} = \lceil\ h \ \cdot\ \frac{N}{R}\ \rceil + 1$ be the position of h in the spectrum, and $\hat{l} =\ \lceil\ l\ \cdot\ \frac{N}{R}\ \rceil + 1$ be the position of l in the spectrum.

Let D_e^y $(T \times \hat{l} - \hat{h} + 1)$ denote the feature matrix.

For each tile $\bar{i} = 1, 2, \ldots, T$ we assign

$$D_{\bar{i}}^{1,\ldots,\hat{l}-\hat{h}+1} = \left|\ F(Q_{\bar{i}}^{1,\ldots,\tilde{M}})_{\hat{h},\ \hat{h}+1,\ \ldots,\ \hat{l}}\ \right|$$

selecting the part of the spectrum between the high and low pass filters h and f. We take the absolute values of the complex result of $F(x_k)$ (defined as its distance in the complex plane from the origin using the Pythagorean theorem).

To accentuate instrument harmonics we perform convolution filtering on the feature vectors in D, using a Gaussian first derivative filter. This works like an edge detection/transient 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 from having such large inputs into $F(t_i)$. For example with a tile size of 10 seconds and a sample rate of 4000 we have a frequency resolution of $\frac{1}{2} 10 \cdot 4000 = 20$KHz.

Typically a 'short-time discrete Fourier transform' is used which has smaller sized inputs (windows) into $F(t_i)$ which are usually overlapping and are multiplied by a window function, attenuating the tails to reduce spectral leakage. Usually these window functions look similar to a Gaussian, for example;

$$\text{Hann}_i = 0.5 - 0.5 \cos \frac{2\pi i}{n-1} w(i)$$

where n is the window size (see [15] for an example). The short-time Fourier transform is relevant when increased time precision is needed as there is a frequency-time resolution trade-off with respect of the input size to $F(t_i)$. This is not a concern in this particular application as our time resolution is never required to be better than 1 second which would still produce adequate frequency resolution.

The Gaussian first derivative filter is defined as

$$-\frac{2\hat{\lambda}}{\upsilon^2} e^{-\frac{\hat{\lambda}^2}{\upsilon^2}}$$

where

$$\hat{\lambda} = \{\ -\lfloor 2\upsilon \rfloor, \lfloor -2\upsilon + 1 \rfloor, \ldots, \lfloor 2\upsilon \rfloor\ \},$$

and

$$\upsilon = b\frac{N}{R}.$$

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 on the vector lengths.

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 dissimilarity matrix of cosines as is common in the literature for similar applications [4]. The cosines are computable easily because they are the the inner products of the respective features (the features have been normalized to unit length).

Let

$$S_{ij} \ = \ 1 - \langle D_i, D_j \rangle,$$

define the dissimilarity matrix.

Then we apply some normalizing transformations. First we center S around 0.5 by raising each element to the power $2s$, where $s = \frac{1}{T^2} \sum_{i,j=1}^{T} S_{ij}$. Since for $x \in [0, 1]$ and $y > 0$ we have $x^y \leq x$ if $y \geq 1$ and $x^y \geq x$ if $y \leq 1$, the transformation $S_{ij} \to S_{ij}^{2s}$ increases the values S_{ij} whenever the mean value $s < 0.5$ and decreases them whenever $s > 0.5$. Note that the transformation keeps the values S_{ij} in the interval

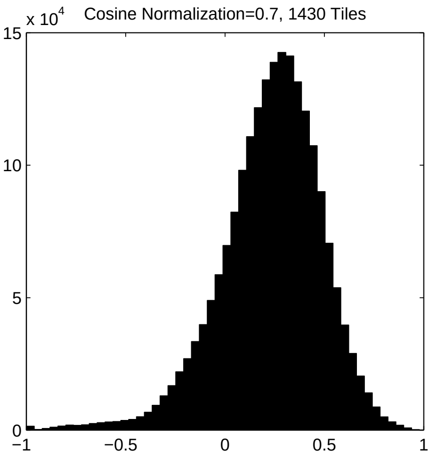


Figure 3: Illustration of the effect of normalization parameter $\hat{c} = 0.7$ on the values in S on radio show Magic Island 110. The small raised section on the left correspond to the tracks.

$[0, 1]$. We find this a convenient and gentle way to rescale S.

Secondly we raise each value S_{ij} to a power $\hat{c} \in [0.5, 1.5]$ and then rescale and translate them to $[0, 1]$ using $S_{i,j} \to 2S_{ij} - 1$. The parameter $\hat{c}$ is tuned so as to achieve the right balance between negative *incentives* and positive *disincentives* for meaningful track placement. The distribution of values in S after the transformations will have a raised tail on left. This will become relevant when we discuss cost matrices as some of them depend on the sign of the value in S.

See Figure 4 for an illustration of S and Figure 3 for an illustration of the normalization.

5.2.2 Self Similarity

We now have a similarity matrix S_{ij} as described in Section 5.2.1.

Let w and W denote the minimum and maximum track length in seconds, these will be parameters of the algorithm that improve the time complexity while not significantly harming the results.

We proceed to constructing a cost matrix $C(f, t)$ that describes the cost of placing a track starting at f and finishing at t (and having length $t - f + 1$). After analysing the data set, we have created 7 cost matrices that exploit observed phenomena in S. We also provide an additional cost matrix which is just a Gaussian random function centred around the mean track length for all times which can be used to regularise the other 7 matrices or used on its own as a comparator to a more naive method of placement.

The cost matrices exploit themes such as contiguity, symmetry, evolution and change as well as simple summation of S as was presented in our last paper [12]. In our previous work S was on the interval $[0, 1]$ and the summation method could only consider disincentives. The new cost matrices have a parameter to shift the consideration of incentive versus disincentive and values on the interval $[-1, 1]$.

On the whole, a significant number of tile pairs within one track are similar to each other. Pairs of tiles that do not belong to the same track are expected to be dissimilar, most of the time. However, tracks have contiguous regions within them that are dissimilar to each other. Transitions between songs may appear as a self-similar region but usually also similar to each adjacent track to varying degrees.

Summation The most obvious strategy of all is to sum up all relevant tiles 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 similarities between all pairs of tiles inside it

$$\tilde{C}(f, t, \Omega) = \sum_{i,j=f}^{t} \hat{S}_{ij}$$

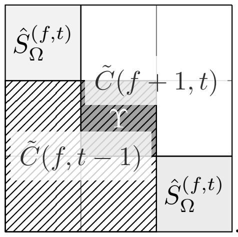
where

$$\hat{S}_{ij} = \begin{cases} \Omega S_{ij}, & \text{if } S_{ij} > 0 \\ (1 - \Omega)\, S_{ij}, & \text{otherwise} \end{cases}$$

for all $i, j \in S$ (the parameter Ω control the balance of positive and negative values). Direct computation using the definition takes $O(TW^3)$ time. We can improve this to $O(TW)$ by using the following recursion (assume that $f+1 \leq t-1$):

$$\tilde{C}(f,t) = \tilde{C}(f+1,t) + \tilde{C}(f,t-1) \\ - \tilde{C}(f+1,t-1) + \hat{S}_{ft} + \hat{S}_{tf}.$$

The recursion implies that the cost of a track of length $L = t - f + 1$ can be calculated from the costs of shorter tracks using a constant number of operations. The following picture (where the middle cell corresponds to $\Upsilon = \tilde{C}(f+1, t-1)$) provides an illustration:



Symmetry A common feature on dance music tracks is partial mirror-symmetry. We build a cost matrix to capture that.

Let $\Lambda(f,t,d)$ be the diagonal parallel to the minor diagonal of S and at the 'distance' d from it. We represent it as an ordered set

$$\Lambda(f,t,d) \\ = \langle S_{f+d,f}, S_{f+d+1,f+1}, S_{f+d+2,f+2}, \ldots, S_{t,t-d} \rangle.$$

For each such diagonal in one triangle/half of S we want to compare each element against its mirror counterpart. For an ordered set Λ we define its cost as

$$\bar{C}(f,t,\Omega)(\Lambda) = \sum_{i=1}^{|\Lambda|} \delta(\Lambda_i, \Lambda_{|\Lambda|-i+1}, \Omega)$$

where

$$\delta(p,q,\Omega) \\ = \begin{cases} 0, & \text{if } \operatorname{sign}(p) \neq \operatorname{sign}(q), \\ \Omega pq, & \text{if } \operatorname{sign}(p) \geq 0 \text{ and } \operatorname{sign}(q) \geq 0, \\ (1-\Omega)pq, & \text{if } \operatorname{sign}(p) < 0 \text{ and } \operatorname{sign}(q) < 0, \end{cases}$$

i.e., 'symmetric' pairs that have the same sign make positive contributions and pairs that have a different sign contribute 0 to the cost. We define the cost matrix as

$$\tilde{C}(f,t,\Omega) = \sum_{d=1}^{t-f+1} \bar{C}(\Lambda(f,t,d,\Omega))$$

Clearly, one can reuse the cost for shorter intervals to calculate the cost of longer ones, namely, $\tilde{C}_{f+1,t-1}$ can be used to calculate $\tilde{C}_{ft}$ this saving computation time.

Static Contiguity Horizontal contiguous traces in S indicate that the track is self-similar (negative values) or self-dissimilar (positive values) due to repetition. If a given tile is the same as a set of contiguous tiles following it, then there is some static contiguous region in the show. The word *static* denotes that the music is not evolving in respect of time.

Let $\Gamma(f,t,h)$ be the horizontal segment in the matrix S showing the similarity of tile $f+h-1$ to 'future' tiles $f+h, f+h+1, \ldots, t$. We represent is as an ordered set

$$\Gamma(f,t,h) \\ = \langle S_{f+h-1,f+h-1}, S_{f+h-1,f+h}, S_{f+h-1,f+h+1}, \ldots, \\ S_{f+h-1,t} \rangle.$$

We define the cost of an ordered set $\Gamma = \langle \Gamma_1, \Gamma_2, \ldots, \Gamma_{|\Gamma|} \rangle$ as

$$\bar{C}(\Gamma,\Omega) = \sum_{i=\rho}^{|\Gamma|} \hat{C}(\Gamma^i, \Omega),$$

where ρ is a parameter indicating the minimum number of contiguous tiles required, $\Gamma^i = \langle \Gamma_1, \Gamma_2, \ldots, \Gamma_i \rangle$ and

$$\hat{C}(\Gamma^i,\Omega) \\ = \begin{cases} \frac{1}{i} \sum_{j=1}^{i} \tilde{f}(\Gamma_j, \Omega), & \text{if the numbers } \operatorname{sign} G_1, \ldots, \operatorname{sign} G_i \text{ are the same,} \\ 0, & \text{otherwise,} \end{cases}$$

where

$$\tilde{f}(v,\Omega) = \begin{cases} \Omega v, & \text{if } \operatorname{sign} v = 1 \\ (1-\Omega)v, & \text{otherwise.} \end{cases}$$

The cost of an interval is defined as

$$\tilde{C}(f,t,\Omega) = \sum_{h=1}^{t-f} \bar{C}(f,t,\Omega) \ .$$

Again it is possible to reuse values $\tilde{C}(f,t,\Omega)$ from shorter intervals to save calculation time for longer intervals.

Evolutionary Contiguity Any diagonal traces in S that are parallel to the main diagonal are partial copies of the track in the future which evolve in respect of time. Evolutionary Contiguity is a cost matrix which compares all adjacent pairs on the diagonals in $\hat{t}$, using the comparator $\delta(p,q,\Omega)$ from the standard symmetry cost function and multiplies those values by the time horizon. All the values are summed and normalized by the track width squared.

Let the evolutionary cost matrix

$$C(f,t,\Omega) = \hat{N}_\Omega \left(\frac{\sum_{d=1}^{t-f+1} \sum_{i=2}^{|\Lambda(f,t,d)|} \delta(\kappa_i, \kappa_{i-1}, \Omega) \cdot d}{(t-f+1)^2} \right)$$

where $\kappa_i = \Lambda(f,t,i)$. As before $\tilde{C}$ is normalized by track width and incentive bias.

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 others and for use on its own for comparison against a 'naive' competitor. Increasing values of ϖ will tighten up the Gaussian although after experimentation we observed that 1 was always the best value and stuck with that.

Normalization All cost matrices are normalized in the following manner. First we divide $\tilde{C}(f,t,\Omega)$ by the track length,

$$\tilde{C}(f,t,\Omega) \leftarrow \frac{\tilde{C}(f,t,\Omega)}{t-f+1}.$$

Then we scale and shift C according to the value of Ω. If $\Omega = 0$, we want the resulting values to fill the interval $[0,1]$, if $\Omega = 0.5$ we want the resulting values to fill the interval $[-1,1]$, and if $\Omega = 1$ we want the resulting values to fill the interval $[-1,0]$. For intermediate values of Ω we want a linear combination.

This is achieved by applying the normalization function to each element:

$$\hat{N}(x,\Omega) = \frac{x - \min_{ft} C(f,t,\Omega)}{\max_{ft} C(f,t,\Omega) - \min_{ft} C(f,t,\Omega)} \hat{h}_\Omega - s_\Omega,$$

where $\hat{h}_w = 2 - 2|0.5 - \Omega|$ and

$$s_\Omega = \begin{cases} 1 - 2(0.5 - \Omega) & \text{if } \Omega < 0.5, \\ 1 & \text{otherwise} . \end{cases}$$

We let

$$C(f,t,\Omega) = N(\tilde{C}(f,t,\Omega), \Omega)$$

for all f and t.

Mixing Cost Functions We mix cost matrices together by adding them. In our experiments we will have a parameter for each cost matrix $\in [0,1]$ to show its contribution to the mixture. The cost matrices will be multiplied by this number before being mixed.

See Figure 5 for an illustration of a single cost matrix and a mixture.

6 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.

We want to reconstruct m track boundaries ($m+1$ tracks).

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

$$\left(\frac{60^2 \times 2}{25}\right)^{24} \approx 1.06 \cdot 10^{59}$$

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 $W = 60 \cdot 15$.

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

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

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 $N(7000, 60 \cdot 15, 190, 25)$ and got $5.20 \cdot 10^{56}$ 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 $i \geq 2$

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

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 $O(TWm)$.

7 Confidence Intervals

It may be useful for some applications to build a framework to allow confidence intervals for our predicted indices. This may also be useful for meaningful comparison of cost matrices.

7.1 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 $m \geq 2$

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

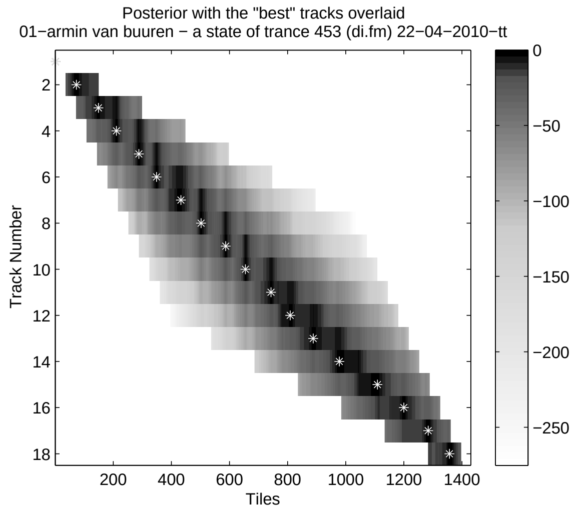


Figure 7: A visualization of $log(P(j,s))$ ($\eta = 10$) for one of the shows in the test set using the cost matrix parameters from experiment 10. The actual tracks are overlaid as white crosses.

See Figure 7 for an example of the posterior for a radio show.

7.2 Posterior Marginal of Song Position

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

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

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

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

7.3 Track Index Confidence

We can use the posterior marginal of song boundary to give estimates of confidence on track index placement and time accuracy.

To estimate the uncertainty of correct track alignment, we select the next highest probability of other track placements at the same time of the optimal placement from $P(j, s)$ and normalize them by the probability of the optimal placement.

Let track index confidence

$$I(j) = 1 - \frac{P(\{1, \ldots, M\} \setminus j, \text{SortInd}(P(j, 1, \ldots, T)_1)^2}{\max(P(j, 1, \ldots, T)_2)}$$

where SortInd(l) will return the original indices corresponding to the sorted list of l.

7.4 Track Time Confidence

Track time uncertainty is estimated by normalizing the value of the next most significant peak in $P(j, \forall s)$ by the probability of actual track placement (which is the most significant peak). Let

$$\tilde{I}(j) = 1 - \frac{\text{Peaks}(P(j, 1, \ldots, T))_2}{\text{Peaks}(P(j, 1, \ldots, T))_1}$$

where $Peaks(s_i)$ is a peak finding algorithm that returns the peaks in descending order of magnitude (we used the `findpeaks` function in MatLab).

See Figure 10 for an illustration of $\tilde{I}(j)$ and $I(j)$.

8 Experimental Methodology

We selected 6 shows at random (two of each show type) to create a training set, which we will refer to as the *GitHub training set*. We have made the set available on-line (see Section 6).

We used the GitHub training set to develop the cost matrices from first principles and to find robust parameters using a genetic algorithm search. Please see Table 1 to see the shows we selected.

In our experiments we decided to fix the tile size at 5 seconds for the sake of speed. A lower tile size does increase accuracy but only marginally. Higher tile sizes can perform more robustly (fewer catastrophic misalignments) but progressively lose out on accuracy.

9 Estimating Segment Count

The original goal of our work was providing the best possible segmentation given a fixed number of tracks m. Little consideration was given to the problem of estimating the number of segments. The problem domain described in this work appears reasonably unique, given that the number of segments is known a priori but the configuration of the segmentation is unknown or subjective.

However. To enable comparison of our work with other methods in the literature, we propose the following way of adapting our framework to estimate the number of homogeneous segments in a data stream.

10 Materials

All the code presented in this paper with the small working test set is available on GitHub [3]. The large data set ($\approx$ 100GB) we received from Denis Goncharov can easily be made available on request (it is in a Google Drive account and easily shareable).

11 Results

Please see Table 2 for the main table of results, Figure 9 for a histogram of predicted versus actual differences for experiment 10. Figure 10 shows a visualization of the confidences and relative performance in relation to show progression (which is a normalized quantity because shows are of variable length).

Our parameter search was slightly limited (we searched 2000 random parameters), and should be done again more extensively. However it was clear from the search that some asymptotic limit was reached.

Adding the Gaussian cost matrix in most cases improves performance because it regularises the track index estimates (see experiment 6 and the improvement in 11). Using the Gaussian cost matrix on its own makes a useful comparison to how our method in general improves on a potential naive approach to the problem.

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

Table 1: The shows randomly selected for inclusion in the *GitHub training set.*

#	**Show Name**	**Artist**	**Date Broadcast**
1	A State Of Trance 453	Armin Van Buuren	April 2010
2	A State Of Trance 462	Armin Van Buuren	June 2010
3	Magic Island 098	Roger Shah	March 2010
4	Magic Island 112	Roger Shah	July 2010
5	Trance Around The World 364	Above & Beyond	March 2011
6	Trance Around The World 372	Above & Beyond	May 2011

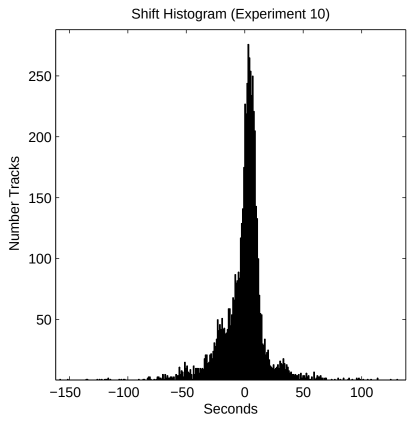


Figure 9: Histogram of the differences between reconstructed and human captured time indices on experiment 10. The slight bumps are uncertainty about track placement perhaps related to the ≈ 14.8 second bar intervals in Trance Music. The humans are much more likely to place an index before a predicted index.

The symmetry sum is the best performing single cost matrix with an even incentive bias with summation and evolution close behind. The gains from decreasing window size past $T = 5$ asymptotically decay which leads us to believe we are approaching some kind of limit. More interesting is that when we perform a random parameter search we also hit a barrier when we would have expected further improved results. We estimate that we are now limited by the error variable associated with the human indices. The 'human disagreement' variable (see Section 3) has a standard deviation of 9.13 seconds which we believe is limiting our progress.

On our previous work we were using a disincentive only summation matrix, and found that normalizing it on the square root of the width produced the best result. This would have been necessary to encourage placement of longer tracks as no incentive was present. So experiment 11 is roughly comparable and indeed produces the same overall mean average to that previous experiment (≈ 20S). Note that we no longer discard any shows from evaluation which makes the result stronger.

Figure 10 seems to suggest that with the summation matrix alone (and we suspect all of the matrices when used in isolation), we suffered significant loss in confidence and performance in the middle of the shows. This would imply that our method was failing to some extent, and to make matters worse; the magnitude of failure depended on the number of tracks. Apparently the mixture of cost functions removes this effect.

12 Summary

We believe our algorithm would be useful for segmenting DJ-mixed audio streams in batch mode. It would be excellent if Sound Cloud[4] for example started to do something similar. Sound Cloud is an on-line music service with many electronic dance music radio shows with the track listing in text. This method would allow them to reliably segment the shows, and they could display an interactive segmentation in the music player.

The new cost matrices in combination improve robustness significantly over single matrices or regularised single matrices (as in our last paper). We are seeing about a 50% improvement in overall mean accuracy over single cost matrices that are correctly normalized. The new cost matrices improve on many drawbacks of our previous work (mainly that it was vulnerable to dissimilar regions within tracks).

Our method still has one key drawback that we are aware of. This is the rare instance where there are head or tail segments to a track that seem independent from the rest of the track. When these are small they usually get absorbed without any problems but they can cause misplacements. In spite of this issue we suspect that our predictions are more accurate and more consistent than the human equivalents while not being as precise in situations when our index agrees with theirs.

A more precise corpus where the DJ was also the cuesheet author would allow us to tune the parameters more succinctly and also the generation of an artificial dataset to test against. This work is forthcoming.

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 an 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 [16] 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.

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

13 Acknowledgements

We would like to thank Mikael Lindgren and Denis Goncharov from cuenation[5] for their help explaining how they make cue-sheets and for providing the data set to test the algorithm on.

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

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.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, “Recurrence plots of dynamical systems,” *EPL (Europhysics Letters)*, vol. 4, no. 9, p. 973, 1987.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[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] 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.

[11] E. Peiszer, T. Lidy, and A. Rauber, "Automatic audio segmentation: Segment boundary and structure detection in popular music," *Proc. of LSAS*, 2008.

[12] 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.

[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] 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.

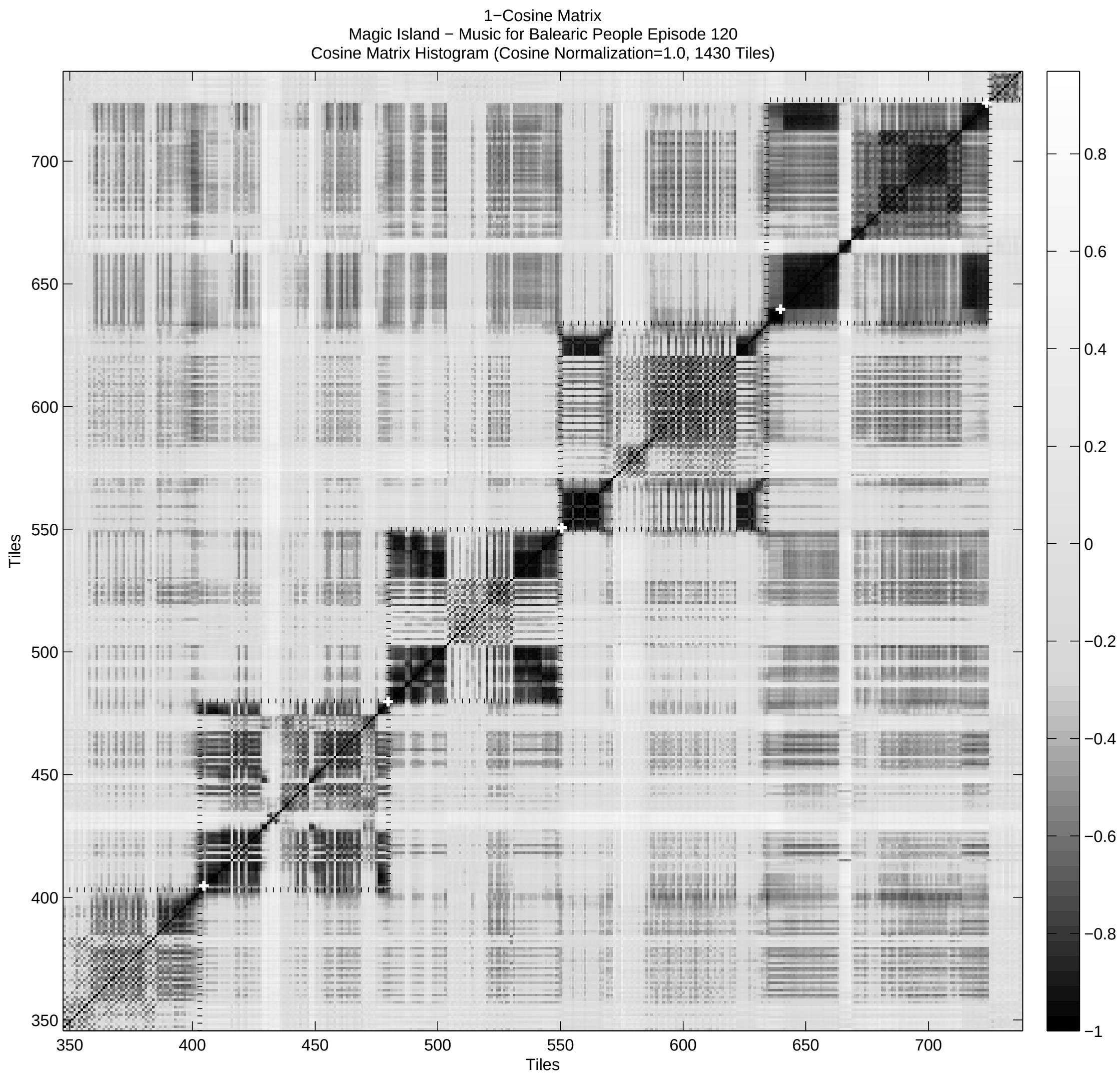
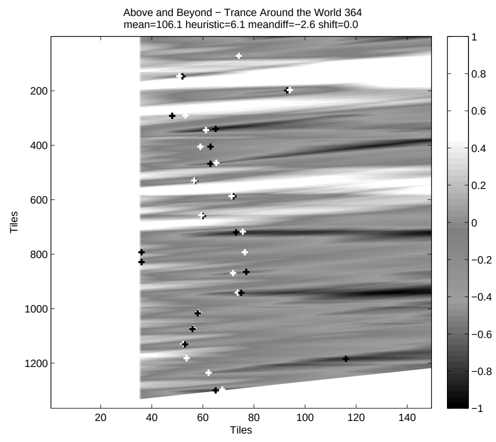


Figure 4: An illustration of the similarity matrix S with the actual indices drawn on with white crosses, and our reconstructed indices indicated with the black dotted lines. There are examples here of evolutionary repetition ($t = 500, \dots, 550$), static contiguity everywhere where there is solid black, and symmetry on the middle two tracks.



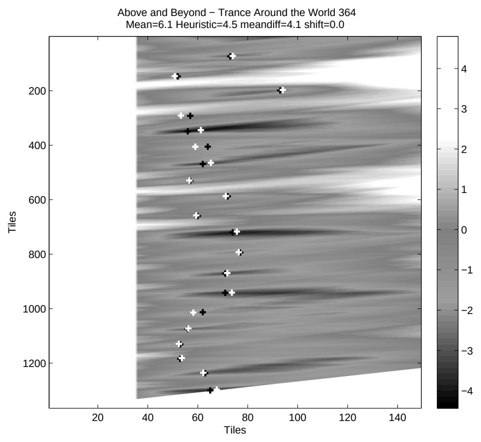
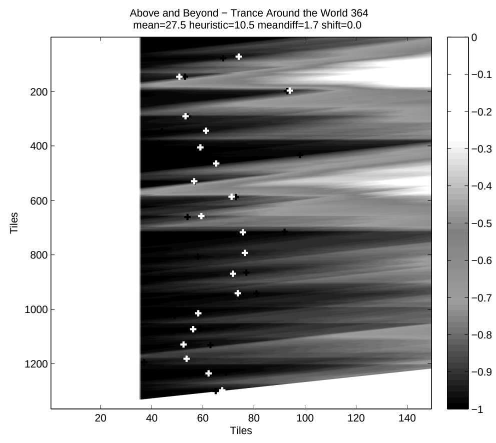


Figure 5: Cost matrices for Magic Island episode 110. On the left is the cost matrix parameters from experiment 5 (summation only with incentive bias 0.5) and the right the mixed cost matrix using parameters from experiment 10. On the right, the predicted (white) and actual tracks (black) are shown in place. Note that the white space is infinite corresponding to the minimum and maximum track parameters w and W.



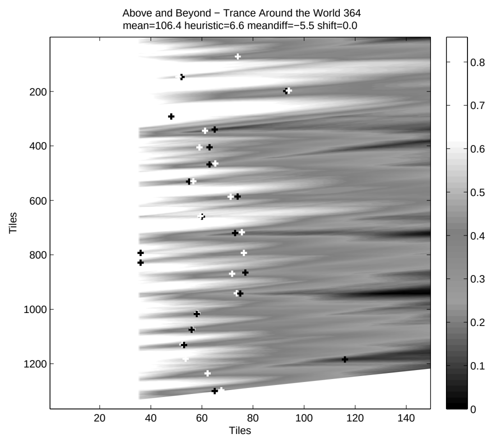


Figure 6: Summation cost matrices for Magic Island episode 110. The matrix on the left has an incentive bias $\Omega = 1$ and therefore only contains disincentives. However the matrix on the right is the other extreme, $\Omega = 1$ containing only incentives.

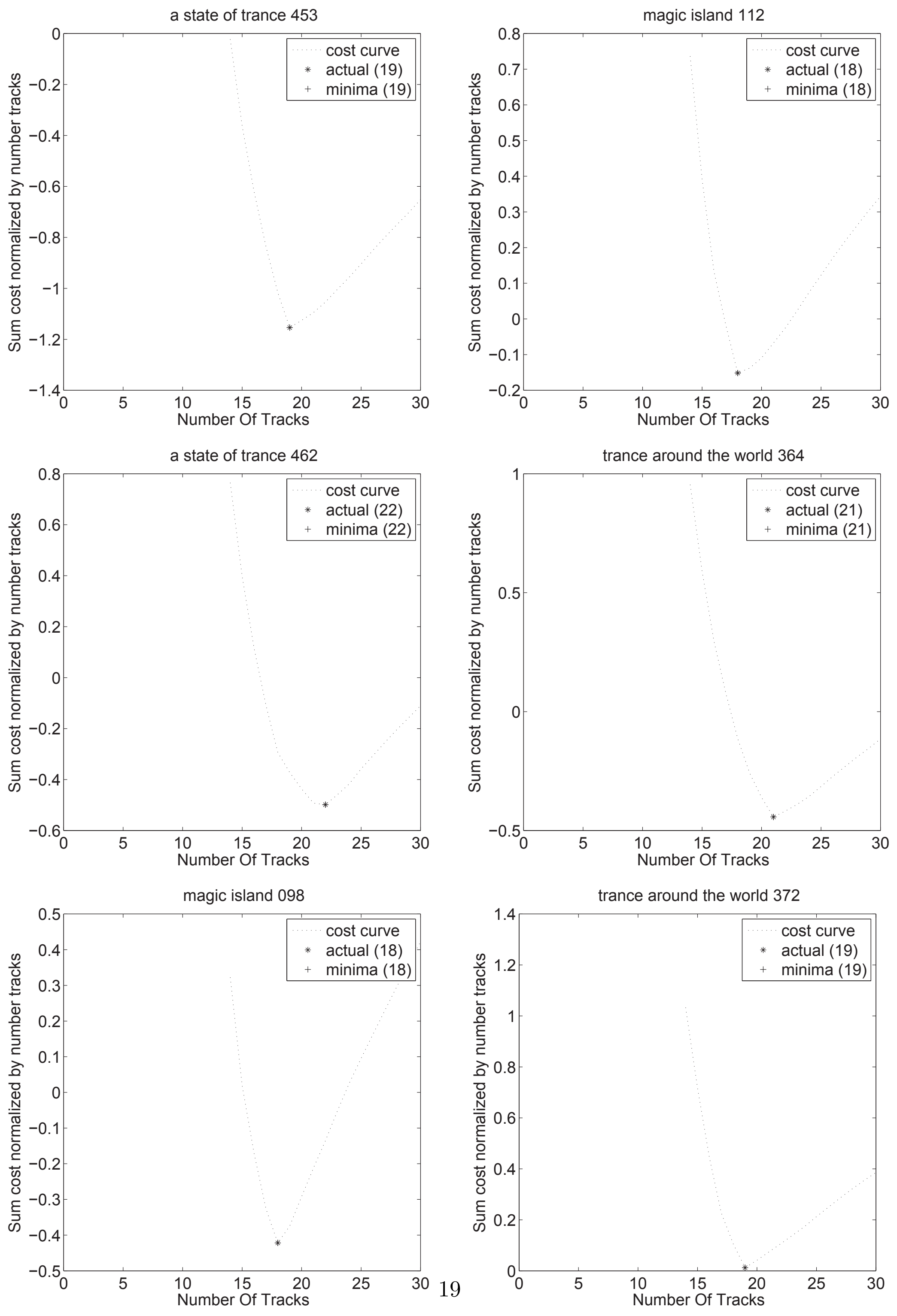


Figure 8: Number of tracks estimated correctly for all six shows in the GitHub training set after a genetics algorithm was run to select a new set of configuration parameters.

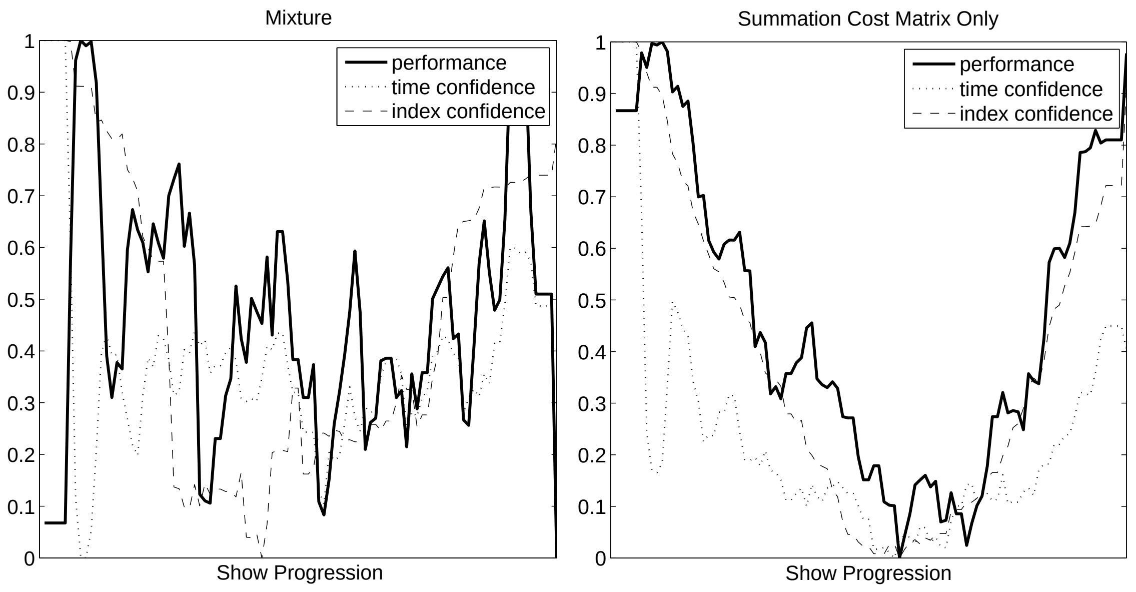


Figure 10: The relative performance and confidence ($\eta = 10$) of two cost matrices. The left is experiment 10 which has a mixture of cost matrices (see the main Results table) and the right is experiment 5 which is only the summation matrix. Ostensibly the summation cost matrix struggles towards the middle of the shows.

Table 2: Main results. Tile size T was 5s for all results. Shift refers to the mean average of the differences in predictions versus human captured indices. $\star$; $h = 1662$Hz, $b = 5$Hz, $\hat{c} = 1$, $\rho = 20$ $\dagger(randomsearch)$; $h = 1662$Hz, $b = 5$Hz, $\hat{c} = 1.2$, $\rho = 26$. Blank cells indicate no contribution from that cost matrix.

Experiment	$h, l, b, \hat{c}, \rho$	Sym Sum	Sym Diff	Symmetry	Contig Past	Contig Fut.	Summation	Gaussian	Evolution	Mean (S)	Heuristic (S)	Shift (S)	60s(%)	30s(%)	20s(%)	10s(%)	5s(%)	1s(%)
1	$*$	$1.0_{\Omega^{\frac{1}{2}}}$								32.13	8.59	1.37	96.06	87.28	78.15	60.48	33.82	20.89
2	$*$		$1.0_{\Omega^{\frac{1}{2}}}$							35.20	16.56	2.52	91.83	73.00	58.54	36.40	19.75	12.41
3	$*$			$1.0_{\Omega^{\frac{1}{2}}}$						68.33	11.50	1.39	92.70	80.35	69.51	50.59	27.55	17.62
4	$*$				$1.0_{\Omega^{\frac{1}{2}}}$					152.58	21.75	5.36	79.64	63.05	49.20	33.12	17.93	10.90
5	$*$					$1.0_{\Omega^{\frac{1}{2}}}$				160.09	27.60	12.25	76.97	54.36	42.27	27.43	14.67	9.04
6	$*$						$1.0_{\Omega^{\frac{1}{2}}}$			34.30	8.65	1.52	95.85	87.17	77.76	60.48	33.66	20.81
7	$*$							$1.0_{\Omega^{\frac{1}{2}}}$		97.83	73.16	-6.08	43.37	22.89	15.53	7.54	3.86	2.38
8	$*$								$1.0_{\Omega^{\frac{1}{2}}}$	33.99	8.49	3.95	96.01	87.76	79.61	60.70	33.82	21.40
9	$*$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	24.88	8.43	3.01	97.20	89.00	80.24	60.55	33.57	20.98
10	$*$	$1.0_{\Omega^{\frac{1}{2}}}$		$1.0_{\Omega^{\frac{1}{2}}}$			$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{\frac{1}{2}}}$	16.31	8.09	2.40	97.61	89.23	80.93	61.87	34.85	22.07
11	$*$						$1.0_{\Omega^{\frac{1}{2}}}$	$1.0_{\Omega^{0.8}}$		21.57	8.92	1.52	97.19	87.57	78.19	59.34	32.65	20.64
12	$\dagger$	$0.4_{\Omega^{0.4}}$	$0.1_{\Omega^{0.1}}$	$0.1_{\Omega^{0.7}}$	$0.2_{\Omega^{0.2}}$	$0.0_{\Omega^{0.7}}$	$0.6_{\Omega^{0.6}}$	$0.7_{\Omega^{0.8}}$	$0.4_{\Omega^{0.1}}$	16.91	8.27	2.33	97.93	89.76	80.74	61.22	34.65	22.16