Segmenting DJ Mixed Music Streams. A long-range self-similarity approach

Tim Scarfe
Wouter M. Koolen
Yuri Kalnishkan

Microsoft Research Talk, Cambridge UK

tim@cs.rhul.ac.uk

June 16, 2013

In this presentation:

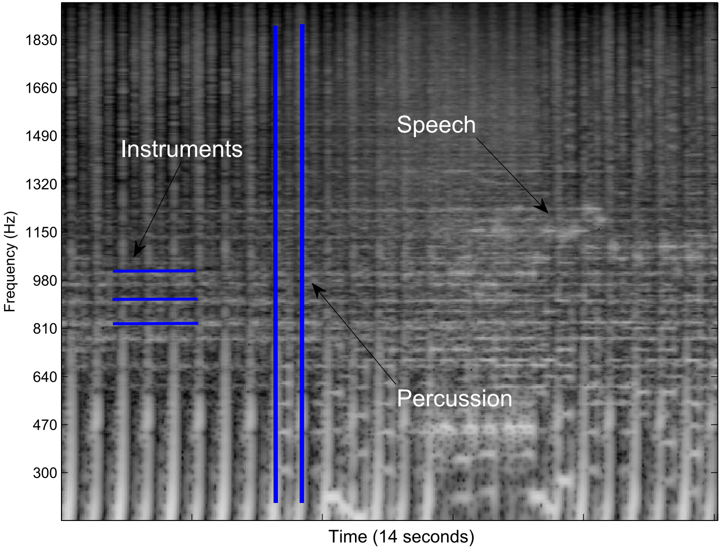
Algorithm proposed for segmenting DJ mixed dance music audio streams into respective tracks

- Domain
- Corpus
- Pre-processing
- Feature Extraction
- Evaluation and Methodology
- Results
- Novelty/Literature
- Conclusions

Electronic Dance Music (EDM)

- EDM is repetitive, self-similar music
- Mixes, Live Events, Podcasts and Radio Shows
- Beat Matching (percussion alignment and harmonic alignment)
- DJs intend to blur the boundaries between tracks

Spectrum for Trance Music



Problem Domain

- Construct optimal time metadata to describe track boundaries *as would be captured by a human* in EDM streams when the number of tracks is known
- We might have the track list from the DJ's web site
- Can be subjective
- Tracks can overlap by *any amount/configuration*

Motivation for extracting this metadata

- Scrubbing works for video, not audio
- Increased awareness, improved listening experience
- Monetisation/click through purchases
- Social networks i.e. Last.FM/Spotify/Facebook
- Table of contents allows skipping ahead
- Generation of CueSheets (there are already community sites where people create and share these)

Our Approach

We have created a specialised algorithm to cater for our circumstances:

- We know how many tracks to find in advance
- EDM Tracks are highly symmetric and comprised of contiguous self-similar regions
- Dynamic Programming approach allows us to evaluate a massive space of possible solutions in reasonable time

What are we using as a corpus

We have been supplied with several broadcasts from three popular radio shows. These are:

- *Magic Island, by Roger Shah* (208 shows);
- *A State of Trance with Armin Van Buuren (*110 *shows)*;
- *Trance Around The World with Above and Beyond (*99 *shows)*.

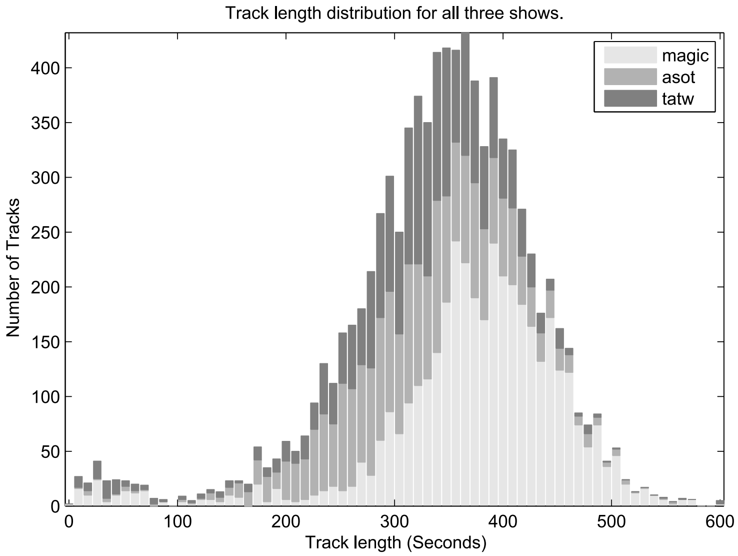
A domain expert from the community (Dennis Goncharov) has supplied us with indices of his evaluations of the optimal placement of track boundaries.

Evaluation

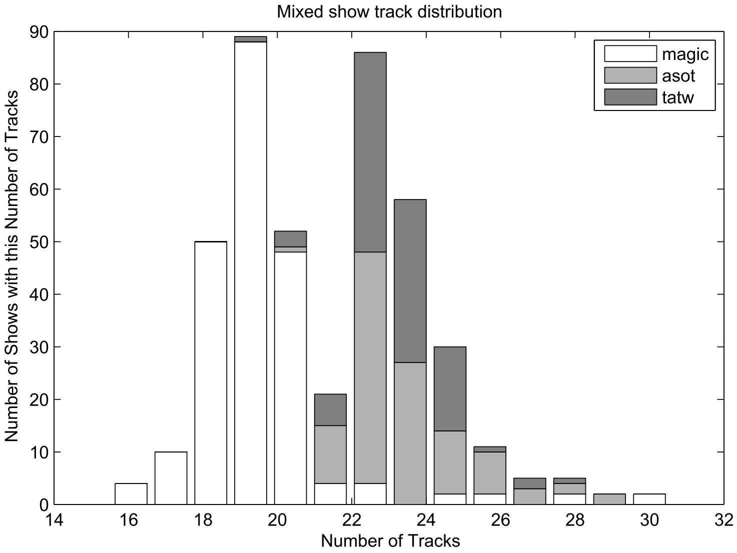
We will evaluate our performance against these indices in two ways (per show):

- Average absolute difference $\frac{1}{|P|}\sum_{i=1}^{|P|}|P_i - A_i|$ (P is predicted indices, A is human indices)
- % of indices that land within time thresholds (s) $\{60, 30, 20, 10, 5, 3, 1\}$ (precisions). Minimum show length we can find will be > 60s.

Dataset information (1)



Dataset information (2)



Preprocessing

- *Minimum track removal* (C) is a parameter; it means that we discard tracks smaller than from the given indices without modifying the media stream at all. We assume they are voice over tracks.
- The spoken introduction track (first track) was removed from all streams along with the corresponding first index.

Transforming the shows into something we can work with

- Shows come in 44100Hz Stereo 192K MP3 format
- Downsampled to 4000Hz
- Audio grouped into a time series of non-overlapping adjacent contiguous tiles of equal size (representing between 1 and 20 seconds).
- Tiles are transformed into the frequency domain using Fourier Analysis
- We want to capture the instruments, the horizontal lines in the spectrogram shown earlier

Fourier Analysis (instrument features)

Fourier analysis allows the representation of time domain process as a set of integer oscillations of trigonometric functions.
The DFT given as

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

transforms a sequence of complex numbers $x_0, \ldots, x_N$ into another sequence of complex numbers $X_0, \ldots, X_N$. We discard the complex complex part

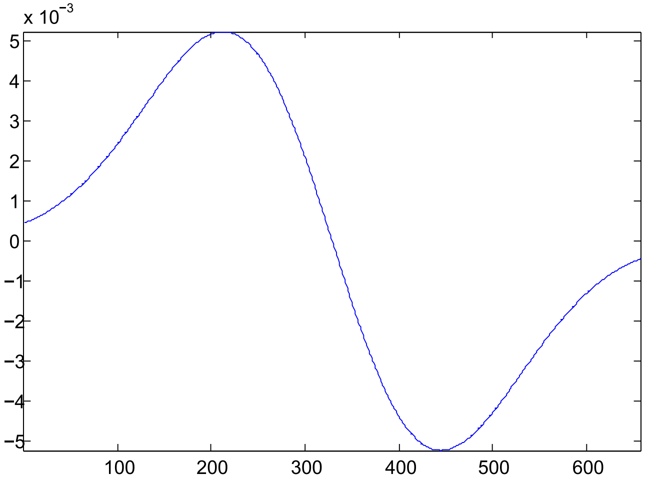
Filtering

- We want pairs of tiles from one track to have a cosine close to 1 and tiles from different tracks to have a cosine close to 0.
- We noticed that this wasn't quite working on the DFT features
- The instrument peaks in the tiles were often slightly misaligned due to the high frequency resolution
- We needed a filter to emphasize extreme transients (like edge detection in vision) and then widen them where they occur.

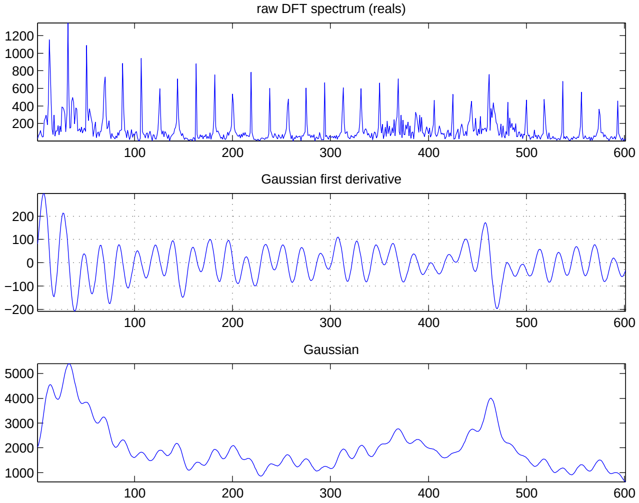
First Derivative Gaussian Filter

- First derivative Gaussian kernel $-\frac{2G}{B^2}e^{-\frac{G^2}{B^2}}$ where $G = -2B, -2B+1, \ldots, 2B$ and $B = b\lfloor\frac{N}{R}\rfloor$ applied to the tile with convolution
- b is the bandwidth of the filter in Hz
- N is tile size
- R is the sampling rate
- Absolute value of the result is taken

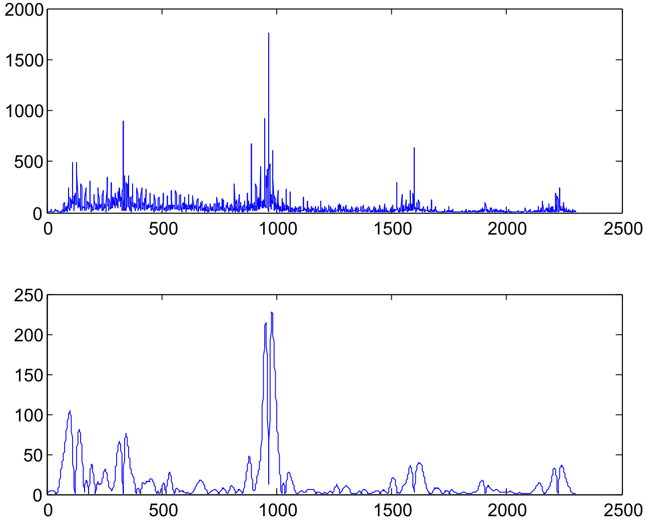
First Derivative Gaussian Filter (2)



First Derivative Gaussian Filter (4)



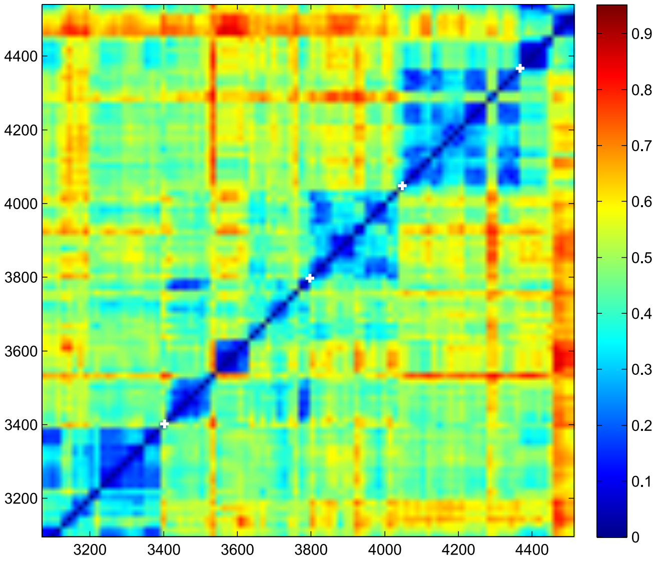
First Derivative Gaussian Filter (3)



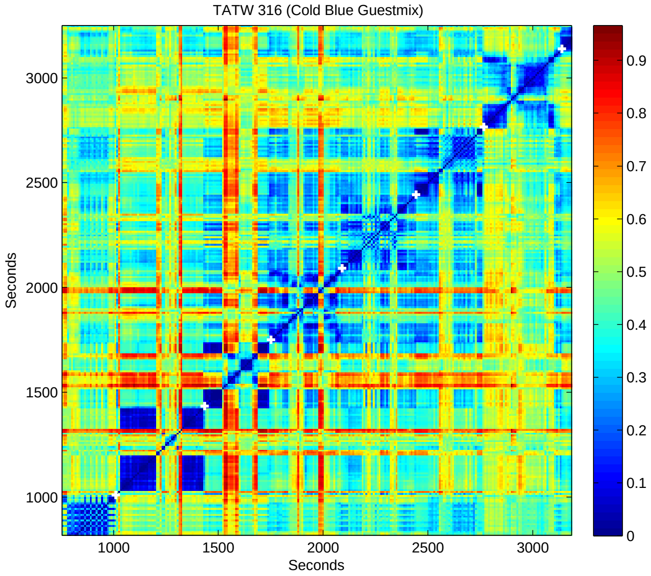
Dissimilarity Matrix

We create an $N \times N$ matrix where N is the number of tiles. The tile vectors have been normalised to unit length. Each cell in the matrix has the $1-$ dot product ($1-$cosine) of a pair of tiles.

Dissimilarity Matrix (2)



Dissimilarity Matrix – Difficult Example



Cost Matrix

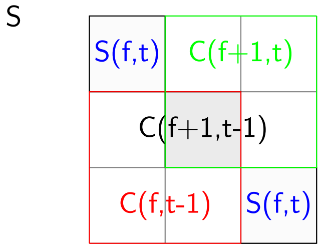
- We need a cost function to score a candidate track starting at time f (*to*) and ending at time t (*from*).
- For this we simply sum the corresponding square of the dissimilarity matrix S
$$C(f,t) \;=\; \frac{\sum_{i=f}^{t}\sum_{j=f}^{t} S(i,j)}{t-f}$$
- Direct calculation using the definition takes $O(TW^3)$ time where W is the maximum track width allowed in seconds (assuming we limit $t-f$)
- T is the length of the stream in seconds.

Dynamic Programming for Cost Matrix

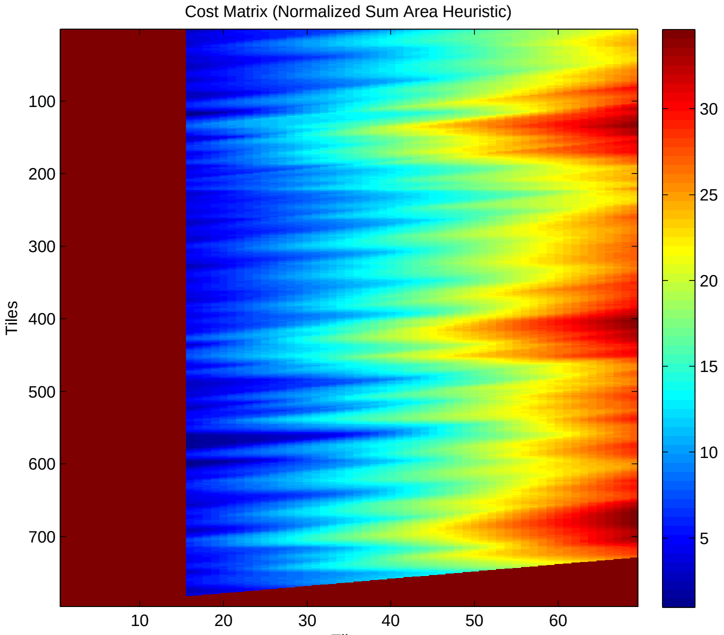
We can compute the full cost matrix (without the normalization) in $O(WT)$ time using the following recursion; set $C(:,1) = \mathrm{diag}(S)$ and then for $f+1 \leq t-1$

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

assuming we pre-compute C for each $1 \leq f \leq t \leq T$.



Cost Matrix Visualization



How many Segmentations?

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

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

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

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

How many Segmentations? (2)

- For large values of T, considering all possible segmentations using brute force is infeasible. For example, a two hour long show would have more than $\left(\frac{60^2\times2}{25}\right)^{24} \approx 1.06 \times 10^{59}$ possible segmentations!
- We can reduce this number by imposing upper and lower bounds on the song length but that only reduces the number of segmentations to about 5.20×10^{56} (see my paper).

Dynamic Programming Solution

We want to compute

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

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

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

Dynamic Programming Solution (2)

To this end, we write the recurrence

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

and for $i \geq 2$

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

In this formula t_i ranges from $(i-1)w + 1$ to $(i-1)W + 1$.

Dynamic Programming Solution (3)

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 $O(TWm)$.

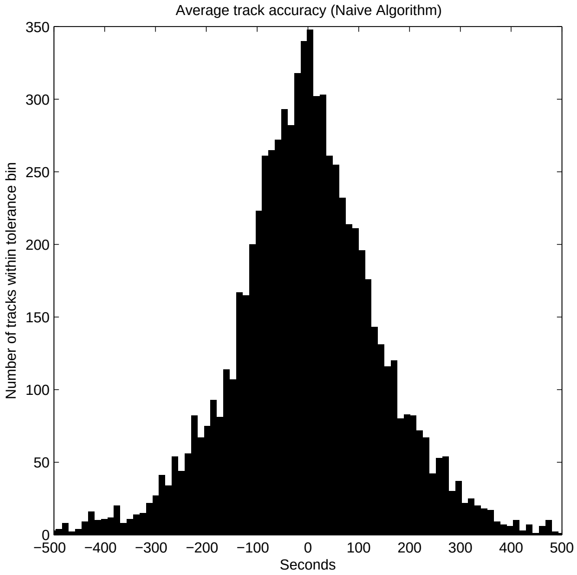
Methodology

- First 10 episodes of each radio show are used to tune the parameters: (*Bandwidth filter b, Tile size M, Minimum track removal C*). The maximum track length W, minimum w.
- These parameters were found with a naive random parameter search (optimised against average track accuracy).

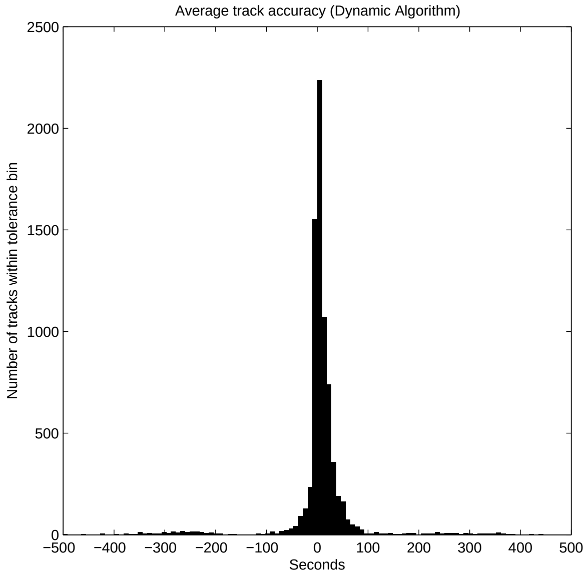
Lack of comparitor

- No clear quantitative way to evaluate the quality of our results
- There are no competing methods for this task
- Let's start by comparing our results with a naive algorithm – evenly spacing the correct number of indices across the show
- Because the track lengths are normally distributed, this approach will work to some extent

Results – Naive Algorithm (112.2s average)



Results – Proposed Algorithm (39.2s average)

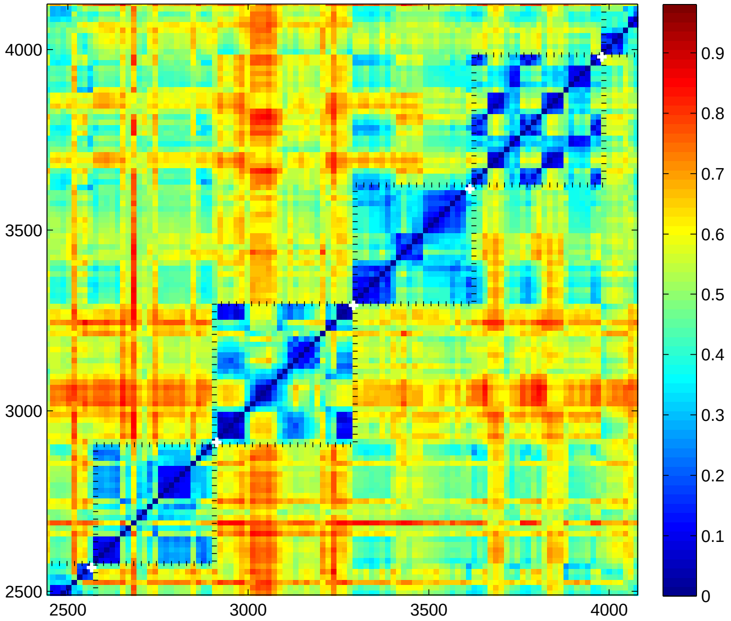


Parameters

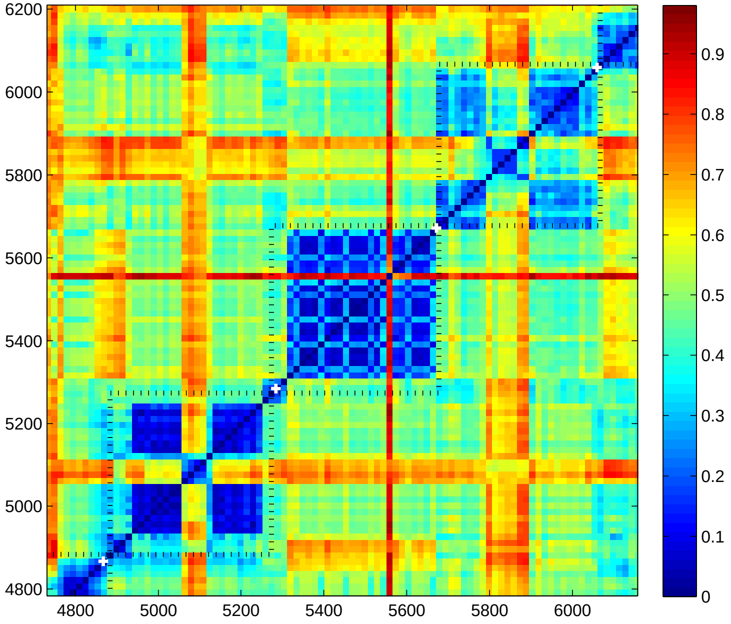
These parameters were found with a naive random search, searching 30 shows (first 10 episodes of each show). These parameters would therefore give similar performance to what is stated here for other shows of the same type.

w	Minimum Track Length (DP)	240	Seconds
W	Maximum Track Length (DP)	617	Seconds
Y	Minimum Track Removal	120	Seconds
M	Tile Size	9	Seconds
b	Bandwidth Filter	5	Hz
l	Low Pass Filter	1618	Hz
h	High Pass Filter	109	Hz

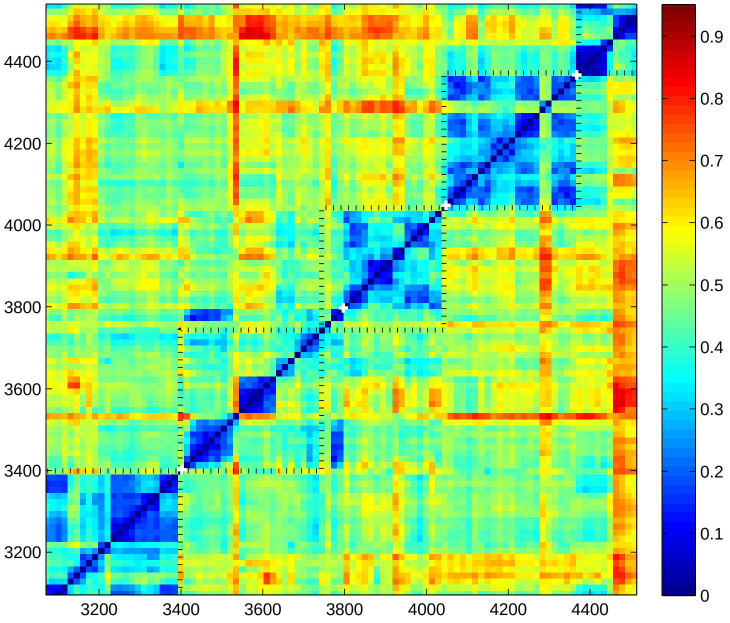
Example reconstruction of indices (1)



Example reconstruction of indices (2)



Example reconstruction of indices (3)



Results

	Dynamic	Naive	Magic	ASOT	TATW
Shows Evaluated	395	395	206	100	89
60 Seconds Precision	95.0%	42.1%	96.4%	91.8%	96.2%
30 Seconds Precision	83.9%	22.0%	88.0%	75.0%	86.0%
20 Seconds Precision	71.5%	14.9%	73.3%	64.3%	76.6%
10 Seconds Precision	53.6%	7.6%	49.6%	53.3%	62.0%
5 Seconds Precision	35.7%	4.0%	29.4%	39.5%	43.7%
3 Seconds Precision	24.2%	2.6%	19.4%	27.6%	29.8%
1 Seconds Precision	8.8%	0.9%	7.0%	10.1%	11.0%
Abs. Avg. Acc. (Sec)	39.2	112.2	39.4	44.0	33.1

What has been done in the literature

- We don't believe an algorithm exists in the literature that would work as well because of our domain knowledge
- Most work on segmentation is based on the construction of a 1d *novelty function* and then peak finding.
- Distance based approaches, Statistical approaches
- Goodwin et al has used a clustering DP for segmentation, intended for structural analysis and focused on short term transients and doesn't consider long term self-similarity as a first class concept.
- J. Foote has done a significant amount of work in this area and was the first to use similarity matrices and consider long term self-similarity.

Conclusion/Further Work

- Our results are significantly better than the naive approach
- Work done on rhythmic features (autocorrelation based)
- Ignoring the percussion seems to improve our performance
- Our algorithm runs fast and is deterministic

The End. Any questions?!