

SOUNDSPOTTER / REMIX-TV: FAST APPROXIMATE MATCHING FOR AUDIO AND VIDEO PERFORMANCE

Michael Casey

Media Futures Lab
Goldsmiths,
University of London

Mick Grierson

Music Department
Goldsmiths,
University of London

ABSTRACT

SoundSpotter is an open source software system for real-time matching of an audio input stream to a database of continuous audio or video. Among its novel features are real-time control over audio segmentation, feature selection and match radius. The system uses audio input to control selection of output from a database using similarity-based matching. The low latency methods employed create a feedback loop between the performer and the database, thus it is a type of electronic musical instrument. We employ exact nearest neighbor searching on variable-length sequences of audio features for matching. The feature space is controlled by range selection over the vector dimensions determining which are left out of similarity calculations. The current implementation is capable of real-time matching in tens of hours of audio or video on current laptop hardware. Finally, we describe *SoundSpotter*'s video editing utility in an application called REMIX-TV.

1. INTRODUCTION

Audio matching is a query by example task that has the goal of returning the closest perceptual match to a given query segment. The task is very complex and can be approached in a number of ways. In this paper, we are most interested in methods that preserve the temporal ordering of acoustic information in the query segment when making a match. As such, the proposed system is built using sequences of audio features, called audio shingles that are segmented by one of several events such as fixed window, rhythmic pulse or onsets.

SoundSpotter was designed as a method to play a large database of audio using acoustic control. The mapping from input to output depends on the audio properties of segments stored in the database, the audio properties of the input, the extracted audio features and the matching algorithm.

SoundSpotter is implemented using the FLEXT C++ framework for compatibility with the Max/MSP and PD environments, but a stand-alone version is also available. Being integrated into such environments enables *SoundSpotter* to be used flexibly and integrated with a number of widely used music analysis and synthesis tools such as onset detectors, beat trackers, phase vocoder cross-synthesis and video objects.

In Section 2 we describe relevant prior work in the area of audio matching, we describe the *SoundSpotter* system and implementation details in Section 3 and Section 4 discusses sound driven algorithmic approaches to video editing using *SoundSpotter* in a novel application called *REMIX TV*.

2. PREVIOUS WORK

Among systems that employ music information retrieval techniques for music performance are *Musaicing* [15], *MatConcat* [12], *Mosievius* [8], *BeatBox* [7], *ScrambledHackz*

[16] and *Caterpillar* [11]. Our work extends the previous work in this area in several important respects. 1) *SoundSpotter* employs sequence matching for selectivity of temporal patterns in the multi-dimensional audio features 2) variable length segmentation is employed so that sequence size can be changed on-the-fly during a performance 3) real-time feature selection allows the focus of the match to change on-the-fly without re-computing the source database features 4) *SoundSpotter* is timbre, time and pitch sensitive with complete control given to the user in real-time 5) a match distance parameter allows continuous range of similarity – dissimilarity in the match 6) matching without replacement is implemented by efficient table lookup 7) live matching implements matching in an accumulating real-time source buffer.

There is a wide spectrum of previous work on audio similarity tasks: from very specific fingerprinting work [5][14] to genre recognition [13][10]. *SoundSpotter* falls in the middle of these two extremes that are most represented in the literature. We want to find audio segments that are similar to a query segment, the current audio input, but that are not exact matches. Our approximate audio matching task needs both new features—we don't expect the very specific features used in fingerprinting to work—and a new matching criteria because we want a good perceptual fit to our query that considers temporal, timbral and pitch features.

Audio Shingling is a technique for similarity matching that concatenates audio feature vectors into a long sequence vector and performs matching on the entire sequence. The number of vectors to concatenate varies between 10 and 100 in the literature with feature vector frame rates of the order of 100ms. In previous work we explain how audio shingles can be used to identify remixed music content by spotting specific audio content that is embedded in a new context, [1][2]. A similar technique was employed in [9] for approximate matching of passages from classical music works, thus identifying the same work performed by different artists and even ensembles such as piano reductions of orchestral works.

In the current work we combine pitch and timbral features in our similarity measure by employing MFCC features, which have been widely used for music information retrieval tasks [13], but not in conjunction with the shingling technique [1][2][9]. Our cepstral coefficients are organized on a strictly logarithmic scale so we call them log-frequency cepstral coefficients (LFCC). In previous work, the first 13 cepstral coefficients are used but in our work we employ the full range of cepstral coefficients for selectivity of both timbral/textural aspects of audio as well as pitch.

3. AUDIO SEQUENCE MATCHING

The system was designed to meet a set of requirements for real-time audio matching. We consider the primary real-time constraint to be low latency-- given an input segment, the time to compute the match should not exceed 20ms, so efficient

algorithms and implementations are required. Additional requirements are that the audio matching should be temporally sensitive, employing sequence matching over variable-length segments, and that it should allow real-time control over the match criteria and the match distance. As such the system implements real-time controls for sequence length, possibly under algorithmic control such as by onset detection, on-the-fly feature selection, adaptive radius-based searching, database range selection, a variable-length queue system for matching without replacement and matching to a real-time accumulating input source.

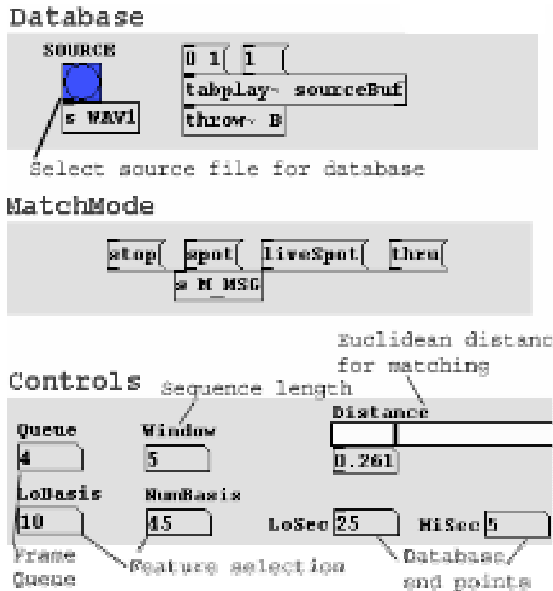


Figure 1: PureData interface to *SoundSpotter*.

3.1. Audio Features

The main requirement for audio sequence matching is that the feature is equipped with a simple norm such as L1 or L2, thus forming a metric space. For example, this is true of spectrum-derived features such as LFCCs and chromagrams, but it is not true of commonly-used statistical features such as GMMs.

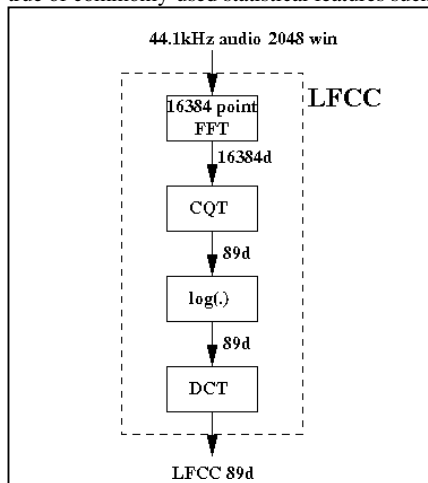


Figure 2: Log-frequency cepstral coefficients feature extraction.

Features are log frequency cepstral coefficients (LFCC) that are like standard MFCCs but use the full range of cepstral coefficients instead of the first 12 to 20 usually retained. As shown in Figure 2, there are 89 elements in the full LFCC feature vector. The features are obtained by applying a 1/16th-octave constant-Q transform to a 16384-point DFT implemented with the FFT for audio sampled at 44.1kHz. The logarithmic spectral frequencies are in the range 62.5Hz to 8kHz, the power spectrum is computed, the real base-10 logarithm is taken and a Discrete Cosine Transform (DCT) applied. The LFCC features are then sequenced into a series of vectors according to the segmentation method.

Feature extraction for the database of audio sources can either be pre-computed or extracted on-line as an accumulating resource during a performance. The time to extract pre-computed features is many hundreds of times faster than real time, the complexity being primarily determined by the FFT.

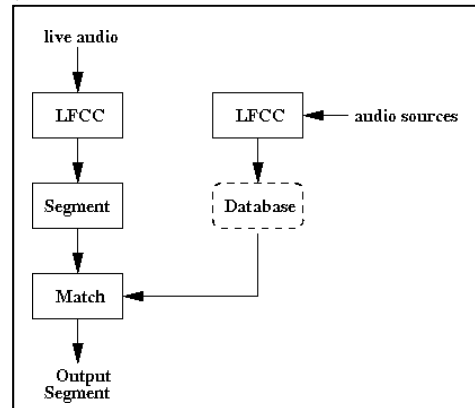


Figure 3: System components and data flow. The live audio is the controller and the audio source can either be the live input (livespot) or a WAV file (spot).

Once the source audio features have been extracted they are stored in a flat file that can be efficiently accessed either as an array in core or by memory mapping from secondary storage to core. For databases that fit in primary store, the features are placed in an array. The array is a pointer to a series of vectors that are indexed by the frame number k . Element i of the k -th d -dimensional vector starting at the database address D is therefore accessible as: $D[k*d+i]$.

One of the controls that *SoundSpotter* offers is real-time feature selection by choosing a low element, a , and high element, b , from the vectors stored in the database. Hence the range of values accessed by the subsequent matching algorithm is: $D[k*d+a+i]$, for $0 < i \leq b-a$.

3.2. Segmentation

Segmentation is a data-dependent stage of the system that divides the database, and live input stream, into chunks that are useful for the given context.

The simplest type of segment in *SoundSpotter* is a variable-length sequence that is triggered at periodic intervals. If the sequence length parameter is constant then the segments are fixed-length. If the sequence-length parameter is varied the segments are non-uniform. Here, matching is unconstrained by boundaries, and exhaustive searching is necessary to find the best alignment between the input segment and the database.

The second type of segment in *SoundSpotter* is an inter-onset interval (IOI). IOI segments attempt to trigger

matching at event boundaries, caused by onsets. A variety of onset detection method can be employed such as PureData's built-in `bonk~` object. The requirement is for SoundSpotter to trigger matching at onset boundaries. The length of the match sequence in this case is the minimum of the last inter-onset interval and a notional maximum sequence length. Here the matching is also aligned to onsets in the audio database, this has the advantage of reducing the number of sequences that must be compared since there are far fewer sequences that start with onsets. The frame index of onsets are stored in a lookup table that is separate from the audio features. This allows fast retrieval by traversing only the list of known onsets in the match algorithm.

The third type of segment is a beat, or tactus interval. Tactus intervals are determined by a common metrical pulse that tracks tempo in the audio input stream. For this type of matching the audio database must also be tempo tracked and tempi and beat locations stored in a lookup table. Here, retrieval is constrained by tempo as well as beats and thus is more efficient than unconstrained matching by variable-length sequences.

3.3. Matching

As in our previous work, [12][13], we use a Euclidean distance metric on the feature space described above to perform matching. For features that are unit norm, the Euclidean norm for audio shingles is efficiently implemented using convolution between the input vector sequence and all overlapping sequences from the database. Figure 4 shows the simplified sequence matching algorithm for an input sequence of length w , starting at memory address V with low element a , high element b , and database of length n starting at address D .

```

void sequenceMatch(
    float* V, /* current input sequence */
    float* M, /* output convolution */
    float* D, /* database sequence */
    int w, /* input sequence length */
    int n, /* database length */
    int a, /* low element selector */
    int b, /* high element selector */
    int d){/* full feature dimensionality */
    int k, i;
    float sum;
    float* tmp = malloc(n*sizeof(float));
    // compute single frame distances
    for(k=0; k<n-w; k++){
        sum=0.0;
        for(i=a; i<b; i++){
            sum+=V[i]*D[k*d+i];
        }
        tmp[k]=sum;
    }
    // compute sequence distances
    M[0]=0.0;
    for(j=0; j< w; j++){
        M[0]+=tmp[j];
    }
    for(k=1; k<n-w; k++){ // recursive computation
        M[k]=M[k-1]-tmp[k-1]+tmp[k+w-1];
    }
    free(tmp);
}

```

Figure 4: Efficient matching using convolution. The multi-dimensional convolution M of the two sequences V and D is proportional to the Euclidean distance between them.

3.4 Real-Time Control

Figure 5 shows how audio shingles are used to compare a sequence of input features with the database sequence of features. The user of range parameters a and b allows for on-line feature selection. By selecting ranges of

coefficients, the matching can be made sensitive to different audio properties such as timbre or pitch.

The first 12 LFCC coefficients characterize wide-band, or timbral, spectral information only. The remaining 77 coefficients characterize increasingly narrow-band, or higher-frequency, components of the spectrum. Range selection over the LFCC feature allows matching to be steered toward timbral features, pitch features or both. The matching algorithm is able to work on time-varying feature selection because a range of features defines a complete metric subspace given the Euclidean distance measure described above.

Control over segmentation is affected by a parameter w that is the length of the sequence to match. w can either be controlled directly by the user or by a segmenter which sets sequence length, w , at each onset to be the inter-onset interval. At each onset, the previous w feature vectors of the input stream are used for matching.

The search radius is the Euclidean distance at which to set the origin for the search. For example, when the search radius is zero then those segments that are closest to zero distance are selected for output. If the distance is set to 0.4 then those segments closest to a distance of 0.4 from the input are selected. The range of distance is between 0 (most similar) and 1 (most dissimilar).

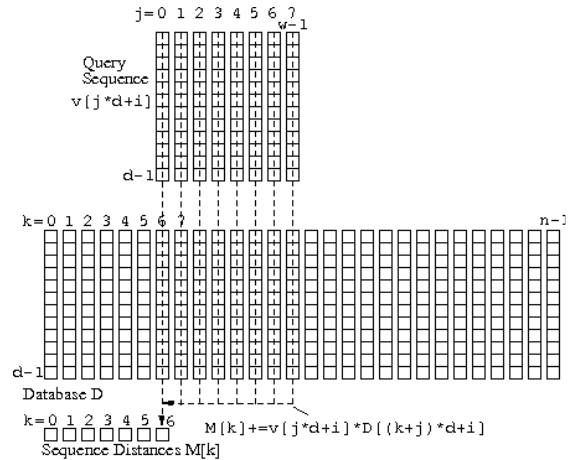


Figure 5: Audio shingling. Each feature vector is a column of a matrix, a sequence of such features is a shingle. The input sequence slides over the database in a multidimensional convolution operation that produces a 1-dimensional sequence of distances.

4. SOUND DRIVEN ALGORITHMIC APPROACHES TO VIDEO EDITING – REMIX TV

Audio analysis and segmentation can be used as an effective method for algorithmic video editing. Representational video material depicting real-world events often large amounts of synchronous material. Events captured on screen are almost always accompanied by direct or related sound events, either as a result or in response to activity that can be seen. Audio material within a video stream can be effectively used as region markers for visual events in a large number of cases. Audio analysis is therefore an acceptable starting point for the development of algorithmic video editing strategies. In addition, it can also be used as a trigger for video manipulation.

Visual material can be re-sequenced using a number of specific strategies, for example, probabilistic methods or

through real-time edit decisions. Significantly, Soundspotter allows for visual material to be re-sequenced through the process of audio matching in live performance.

4.1. Previous Work

Previous work utilising similar approaches includes Weapons of Mass Deconstruction [4], a software suite for the analysis and real-time editing of audiovisual material. In this system, a low pass filter is used for event onset detection. This process can operate in either real-time (using RAM) or non real-time (using the hard disk).

In non-real time mode, video files are prepared first. Audio and video streams are rendered as separate files with identical file names. When a video file is loaded, its audio stream is analysed.

Real-time mode allows for the analysis and resequencing of audiovisual material with a specified delay limited by available RAM, and is equal to the overall length of the material in the region list. This list is continually updated through the use of a dual alternating buffer system – while one set of data is being used for editing, the next is being analysed.

Region start points are stored in a buffer in millisecond units, and are assumed to end one millisecond before the next region starts. The audio region start point is used to calculate the video start frame. For a video file with a frame rate of 25 frames per second, region start points specified in milliseconds are divided by 40 (1000/25), giving the corresponding frame number whilst allowing audio regions to retain their timing accuracy. This maintains perfect audio-video synchronisation even in cases where extreme audio and video manipulation is performed.

4.2. Remix TV

Soundspotter's Remix TV functionality improves on previous systems in two key areas. First, Soundspotter's audio analysis and segmentation approach is significantly more advanced, allowing for re-sequencing to be driven by audio matching. Sonic events in a video stream can be located and restructured in real-time, with connected visual material synchronised automatically. Secondly, Soundspotter's 'livespot' mode dispenses with the need for a real-time dual alternating buffer system, whilst enhancing real-time functionality.

On computing an audio match, Soundspotter frame values are converted to video frame numbers by scaling the output to the frame rate (25 frames per second in most circumstances). Soundspotter outputs synchronisation points related to its hop size. To maintain synchronisation, video frames are matched to the nearest millisecond. The region size is calculated relative to window size, so that playback of video halts on completion of a loop equal to audio segment length.

This system can be used for the remixing of continuous visual material via audio matching. Performers can present specific sections of video material by matching known audio sequences that correspond with specific moments, or present sequences spontaneously. In this way, video and film material can be Re-edited in an improvised fashion, or to a structured score.

An effective use for Soundspotter's remix TV functionality is real-time non-linear re-editing of narratives. Musical motifs are often used to underscore specific characters, events, and themes in narrative cinema [6]. Soundspotter allows for the accurate recall of scenes in Cinema and Television through the real-time live matching of

musical material specific to particular Films. The entire corpus of Hitchcock's Vertigo (1957) can be analysed, complete with Bernard Herrmann's score. If a performer then plays the love theme from Vertigo into Soundspotter, there is a high chance (given the correct settings) that the audience will see an image of Jimmy Stuart kissing Kim Novak. In this way, the entire musical narrative structure of a cinematic work can be navigated by a performer, providing a non-linear approach to cinematic viewing based on musical, thematic, character-based or motif search.

As Soundspotter efficiently encodes elements of texture in addition to note sequences, specific scenes featuring both sound effects and music can be triggered by using a similar sound as a search query. Scenes can be analysed by Soundspotter and used to trigger each other. In this way, similarities in film form can be explored in live performance via Soundspotter's sound analysis and audio matching.

REFERENCES

- [1] Casey, M. and Slaney, M. Song Intersection by Approximate Nearest Neighbor Search. in *Proc. ISMIR*, 2006.
- [2] Casey, M. and Slaney, M. The importance of sequences in music similarity. in *Proc. ICASSP*, 2006.
- [3] Chion, M., *Audiovision Sound on Screen*, Columbia University Press, 1994
- [4] Grierson, M., *Audiovisual Composition*, Thesis, University of Kent, 2005
- [5] Herre, J., Allamanche, E., Hellmuth, O., Kastner, T., Robust identification/fingerprinting of audio signals using spectral flatness features. *Journal of the Acoustical Society of America*, Volume 111, Issue 5, pp. 2417-2417, 2002.
- [6] Kalinak, K., *Settling the score: music and the classical Hollywood film*, University of Wisconsin Press, 1992
- [7] Kapur, A., Benning, M. and Tzanetakis, G. Query-by-Beat-Boxing: Music Retrieval for the DJ, in *Proc. ISMIR*, Barcelona, 2004.
- [8] Lazier, A. and Cook, P. Mosievius: Feature Driven Interactive Audio Mosaicing, in *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, London, 2003.
- [9] Muller, M., Kurth, F. and Clausen, M. Audio Matching via Chroma-Based Statistical Features. In *Proc. ISMIR*, London, Sept. 2005
- [10] Pampalk, E., Flexer, A., Widmer, G. Improvements of Audio-Based Music Similarity and Genre Classification. in *Proc. ISMIR*, pp. 628-633, 2005.
- [11] Schwarz, D. A System for Data-Driven Concatenative Sound Synthesis. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy, 2000.
- [12] Sturm, B. MATConcat: An Application for Exploring Concatenative Sound Synthesis using MATLAB, in *Proc. ICMC*, Miami, 2004.
- [13] Tzanetakis, G. and Cook, P. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):2933-2002, 2002.
- [14] Wang, A., Smith, J. System and methods for recognizing sound and music signals in high noise and distortion. *United States Patent 6990453*, 2006
- [15] Zils, A. and Pachet, F. Musical Mosaicing. *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01)*, Limerick, Ireland, 2001.
- [16] <http://www.popmodernism.org/scrambledhack/>