# SOUNDSPOTTER – A PROTOTYPE SYSTEM FOR CONTENT-BASED AUDIO RETRIEVAL

*Christian Spevak*
Faculty of Computer Science
University of Karlsruhe, Germany
spevak@ira.uka.de

*Emmanuel Favreau*
Ina-GRM
France
efavreau@ina.fr

## ABSTRACT

We present the audio retrieval system "Soundspotter," which allows the user to select a specific passage within an audio file and retrieve perceptually similar passages. The system extracts frame-based features from the sound signal and performs pattern matching on the resulting sequences of feature vectors. Finally, an adjustable number of best matches is returned, ranked by their similarity to the reference passage. Soundspotter comprises several alternative retrieval algorithms, including dynamic time warping and trajectory matching based on a self-organizing map. We explain the algorithms and report initial results of a comparative evaluation.

## 1. INTRODUCTION

Soundspotter is an audio retrieval system that has been developed as a tool for detecting specific passages within a piece of music or any other audio file. It is based on the *query by example* paradigm, where the user selects a reference passage and asks the system to retrieve perceptually similar occurrences. We refer to this task as "sound spotting." In this context, we interpret "perceptual similarity" as similarity in spectral evolution, which is measured by comparing sequences of feature vectors. The current implementation of our system is a prototype designed to evaluate different retrieval algorithms. Our research fits into the emerging discipline of audio information retrieval [1, 2]. Related work in this area includes content-based retrieval and browsing of sound files in audio databases [3, 4, 5], audio classification [6, 7], and sound source recognition [8].

Soundspotter is based on a modular architecture consisting of four stages. First, a frame-based feature extraction is carried out using mel-frequency cepstral coefficients (MFCCs). Second, the feature vectors are clustered or mapped onto a self-organizing map (only required for some of the implemented algorithms). Third, pattern matching is performed by comparing the given reference sequence with possible test sequences, and finally, a prespecified number of best matches is selected. The former two stages produce an index of the sound file, while the latter perform the actual retrieval. An overview of the implemented algorithms and their specific stages is provided in Figure 1. The most straightforward variant of the implemented sound spotting algorithms, consisting of feature extraction, trajectory matching, and match selection, is described in Section 2, and the alternative algorithms – dynamic time warping, mapping & trajectory matching, clustering & string matching, and clustering & histogram matching – are explained in Section 3. The remaining sections present the graphical user interface (Section 4), some initial evaluation results (Section 5), and plans for further development (Section 6).

## 2. BASIC ALGORITHMS

### 2.1. Feature Extraction

Feature extraction in Soundspotter is based on a variant of the *mel-frequency cepstral coefficient (MFCC)* representation. MFCCs are commonly used in speech recognition systems because they provide a concise representation of spectral characteristics. The standard implementation [9, 10] comprises the following stages: the signal is converted to short frames (approximately 20 ms) using a window function, and for each frame, a discrete Fourier transform is computed. The magnitude spectrum is converted to a logarithmic scale and transformed to a smoothed mel spectrum. Finally, the discrete cosine transform is calculated, and typically the first 13 coefficients are used to form a "cepstral" feature vector.

We adapted the standard MFCC implementation to the specific demands of music retrieval by increasing the frequency range of the Fourier transform and the mel spectrum and by modifying the amplitude scaling. The logarithm conversion, which is supposed to provide an approximation to the perceptual representation of loudness, has been replaced by the power $x^{0.23}$, derived from Stevens' law [11]. This power law is not only more psychophysically plausible, but provides a continuous representation of the signal from silence (magnitude of 0) to the maximum level.

### 2.2. Trajectory matching

The basic pattern matching algorithm implemented in Soundspotter has been termed *trajectory matching* because it compares trajectories in the feature space. The distance between two vector sequences of equal length is calculated by performing a one-to-one comparison of their respective elements. The algorithm takes the given reference pattern and compares it with every possible test pattern of equal length along the total sequence. A distance function, defined over all test patterns, is calculated by taking the average Euclidean distance between corresponding frames in the reference pattern and the test pattern.

### 2.3. Match Selection

The *match selection* algorithm receives the distance function produced by the pattern matching stage and returns a list of *M* best matches, i.e. excerpts that are similar to the reference pattern provided in the original query. Soundspotter uses a ranking-based approach, where the user specifies the maximum number of matches, *M*, and the system returns up to *M* matches ranked by their similarity to the reference pattern.

First, the algorithm produces a list of potential matches by recording all local minima of the distance function. To preclude

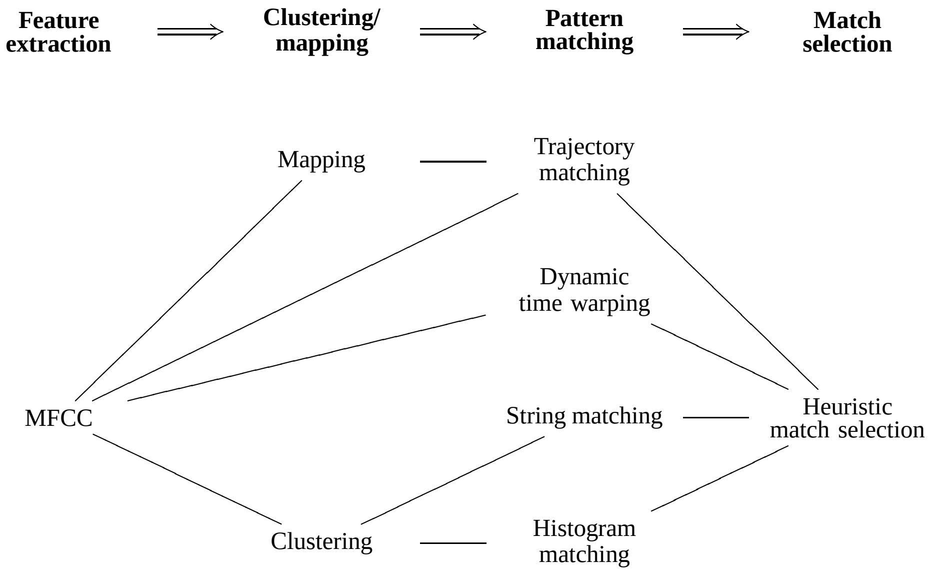

Figure 1: *Overview of the implemented sound spotting algorithms and their respective processing stages.*

the selection of largely overlapping matches, a heuristic minimum distance rule is introduced. It requires adjacent matches to have a minimum inter-onset interval (IOI) of $0.8R$, where $R$ is the length of the reference pattern. The match selection algorithm proceeds along the list of potential matches and checks for each entry if the preceding IOI is large enough. If it is too small, the inferior match is removed. The revised list of potential matches is finally used to extract the $M$ best matches. If the length of the list is less or equal $M$, all potential matches are returned.

## 3. ALTERNATIVE ALGORITHMS

### 3.1. Dynamic Time Warping

*Dynamic time warping (DTW)* is a pattern matching technique originally developed for speech recognition [12]. In Soundspotter, it is utilized to extend the trajectory matching algorithm to sequences of different length by permitting insertion or deletion of elements during alignment. The optimal alignment, where reference and test sequence have minimum distance, is determined using dynamic programming [13].

DTW matches a reference sequence $\mathcal{R} = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_I)$ with subsequences of the total sequence $\mathcal{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_J)$ by creating a distortion matrix (Figure 2), in which each value $D(i, j)$ is associated with a cumulative distortion measure. $D(i, j)$ represents the best-so-far accumulated distance between the first $i$ elements of $\mathcal{R}$ and a subsequence of $\mathcal{A}$ ending at position $j$. It is recursively calculated using the relation

$$\begin{aligned} D(i,j) = \min(&D(i-2, j-1) + w_1 d(i,j), \\ &D(i-1, j-2) + w_2 d(i,j), D(i-1, j-1) + w_3 d(i,j)), \end{aligned} \quad (1)$$

where $d(i, j)$ is the local distance between the $i$th vector of $\mathcal{R}$ and the $j$th vector of $\mathcal{A}$. The parameters are set to $w_1 = w_2 = 2$ and $w_3 = 1$. These settings, equivalent to the path constraint displayed in Figure 2, have been selected after a number of informal trials.

$$\begin{array}{c|cccc} & \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_J \\ \hline \mathbf{r}_1 & D(1,1) & D(1,2) & \cdots & D(1,J) \\ \mathbf{r}_2 & D(2,1) & D(2,2) & \cdots & D(2,J) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{r}_I & D(I,1) & D(I,2) & \cdots & D(I,J) \end{array}$$

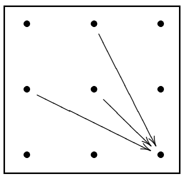

Figure 2: *Distortion matrix and path constraint used in the dynamic time warping algorithm.*

The boundary condition is defined by

$$D(1, j) = d(1, j). \quad (2)$$

The values $D(I, j)$ in the last row of the distortion matrix constitute the distance function to be used for match selection. The values denote the optimal cumulative distance between the reference sequence $\mathcal{R}$ and a subsequence of $\mathcal{A}$ ending at position $j$. In order to determine the beginning of the potential match, its path through the matrix up to the first row has to be determined using *backtracking*. This is implemented as follows: for each $D(i, j)$ in the distortion matrix, a pointer to the respective predecessor is stored as soon as the value is calculated. Starting at $(I, j_{\text{last}})$, the path is traced back by following the pointers until the first row of the matrix is reached at $(1, j_{\text{first}})$. The retrieved subsequence is then given by the elements along the path.

### 3.2. Mapping & Trajectory Matching

Unlike the above algorithms, *mapping & trajectory matching* inserts an additional processing stage between feature extraction and pattern matching. It uses a self-organizing map (SOM) to perform a topology-preserving mapping of the thirteen-dimensional feature vectors onto a two-dimensional surface. The coordinates of the resulting trajectory are passed on to trajectory matching. The application of the SOM is motivated by previous research on self-organizing sound feature maps [14, 15, 16].

A SOM can be visualized as an array of neurons arranged on a typically two-dimensional lattice. Each neuron is associated with an $n$-dimensional weight vector $\mathbf{w} = [w_1, w_2, \ldots, w_n]$, where $n$ corresponds to the dimension of the input signal. During training, the weight vectors adapt to the presented input data and (ideally) converge to a reduced representation [17].

In this case, the SOM is trained on the complete set of feature vectors extracted from the sound file using a batch training algorithm, which provides a computationally efficient way of SOM training [18]. The dimensions of the SOM are heuristically derived from the training data; the number of units is chosen to be approximately the square root of the number of training vectors, i.e. all the feature vectors extracted from the respective audio file. During the update process, each weight vector $\mathbf{w}_i$ is replaced by a weighted average of the data vectors $\mathbf{v}_j$, such that

$$\mathbf{w}_i(t+1) = \frac{\sum_{j=1}^{n} h_{c_j i}(t)\mathbf{v}_j}{\sum_{j=1}^{n} h_{c_j i}(t)}, \tag{3}$$

where $h_{c_j i}(t)$ is a Gaussian neighbourhood function that decreases for subsequent training steps and depends on the distance between the best-matching unit $c_j$ and the current unit $i$.[1]

In order to obtain a finer resolution and a more balanced mapping, the standard *response focus* metric, which maps each input vector to the position of its best-matching unit, has been replaced by the *centroid of activation* metric [19]. For a given input vector $\mathbf{v}$, an activation function $A(\mathbf{v}, i)$ is calculated over all neurons $i$. The function takes the Euclidean distance between the input vector and the respective weight vector,

$$A(\mathbf{v}, i) = \|\mathbf{v} - \mathbf{w}(i)\|, \tag{4}$$

and determines the centroid of activation $\mathbf{c}(\mathbf{v})$,

$$\mathbf{c}(\mathbf{v}) = \frac{\sum_i \mathbf{x}(i)A(\mathbf{v}, i)}{\sum_i A(\mathbf{v}, i)}, \tag{5}$$

where $\mathbf{x}(i)$ represents the two-dimensional coordinates of unit $i$ on the SOM.

A sequence of feature vectors is thus transformed into a trajectory on the SOM (see Figure 3 for an example). Pattern matching is subsequently performed using the trajectory matching algorithm described in Section 2.2.

## 3.3. Clustering & String Matching

The *clustering & string matching* algorithm uses a SOM to cluster the feature vectors into discrete symbols, which are then subjected to approximate string matching. The SOM is trained as described in Section 3.2, but the actual clustering is based on the standard response focus metric, which maps each input vector to its best-matching SOM unit. Because the resulting sequence of SOM units is interpreted merely as a *string*, i.e. a sequence of symbols belonging to a finite alphabet, all mutual relations between the SOM units, such as distance or similarity, are disregarded. The alphabet is given by the index numbers of the units.

*String matching with k differences* [20] is used to search for approximate occurrences of the reference sequence $\mathcal{R}$ in the total sequence $\mathcal{A}$. The algorithm requires a distance metric such as *edit distance*, which measures the distance between any two strings in terms of the minimum number of edit operations – substitutions, insertions, and deletions – needed to transform one string into the other. The $k$ differences algorithm has been adapted to the Soundspotter environment by combining the distance calculation with the ranking-based match selection process described in Section 2.3.

[1] It is defined as $h_{c_j i}(t) = e^{-d(c_j,i)^2/2\sigma_t^2}$, where $d(c_j, i)$ is the distance on the map between units $c_j$ and $i$, and $\sigma_t$ is the neighbourhood radius at training step $t$.

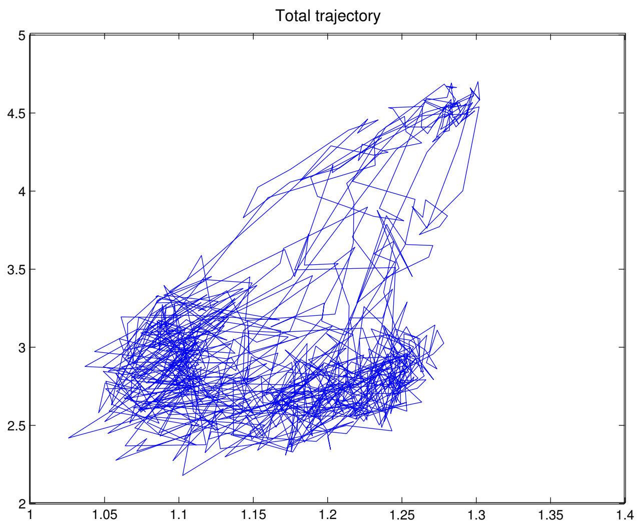

Figure 3: *Mapping & trajectory matching. The diagram illustrates a trajectory on a SOM comprising* $3 \times 10$ *neurons.*

The implementation of the string matching technique is based on dynamic programming, analogous to the DTW algorithm presented in Section 3.1. Figure 4 displays a typical distortion matrix, where

$$\begin{aligned} \mathcal{A}_u &= (07, 09, 17, 17, 17, 17, 18, 17, 08, 09, 09, 17, 20), \\ \mathcal{R}_u &= (09, 09, 17). \end{aligned}$$

The elements of the matrix denote the cumulative distance $D(i, j)$ between the first $i$ elements of $\mathcal{R}_u$ and a substring ending at the $j$th element of $\mathcal{A}_u$. The matrix is completed row by row using the recursive relation

$$\begin{aligned} D(i, j) = \min(&D(i-1, j-1) + d(i, j), D(i-1, j) + 1, \\ &D(i, j-1) + 1), \end{aligned} \tag{6}$$

where

$$d(i, j) = \begin{cases} 0 & : \ u(i) = u(j) \\ 1 & : \ u(i) \neq u(j) \end{cases}. \tag{7}$$

In terms of edit distance, the three arguments of the min function correspond to substitution, insertion, and deletion, respectively, which result in diagonal, vertical, and horizontal movements in the distortion matrix, respectively. For each $D(i, j)$, a pointer is stored referring to the position $(i', j')$ of the decisive min argument (illustrated by the arrows in Figure 4).[2] The boundary conditions for the recursive relation are given by

$$D(0, j) = 0, \tag{8}$$

$$D(i, 0) = i. \tag{9}$$

[2] If there is more than one minimal value, only the first one is recorded (according to the order given in Equation 6).

| | | 07 | 09 | 17 | 17 | 18 | 17 | 08 | 09 | 09 | 17 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 09 | 1 | $\nwarrow$ **1** | $\nwarrow$ 0 | $\nwarrow$ 1 | $\nwarrow$ 1 | $\nwarrow$ 1 | $\nwarrow$ 1 | $\nwarrow$ 1 | $\nwarrow$ **0** | $\nwarrow$ 0 | $\nwarrow$ 1 | $\nwarrow$ 1 |
| 09 | 2 | $\nwarrow$ 2 | $\nwarrow$ **1** | $\nwarrow$ 1 | $\nwarrow$ 2 | $\nwarrow$ 2 | $\nwarrow$ 2 | $\nwarrow$ 2 | $\nwarrow$ 1 | $\nwarrow$ **0** | $\nwarrow$ 1 | $\nwarrow$ 2 |
| 17 | 3 | $\nwarrow$ 3 | $\nwarrow$ 2 | $\nwarrow$ **1** | $\nwarrow$ 1 | $\nwarrow$ 2 | $\nwarrow$ 2 | $\nwarrow$ 3 | $\uparrow$ 2 | $\uparrow$ 1 | $\nwarrow$ **0** | $\leftarrow$ 1 |

Figure 4: *String matching with k differences using dynamic programming.*

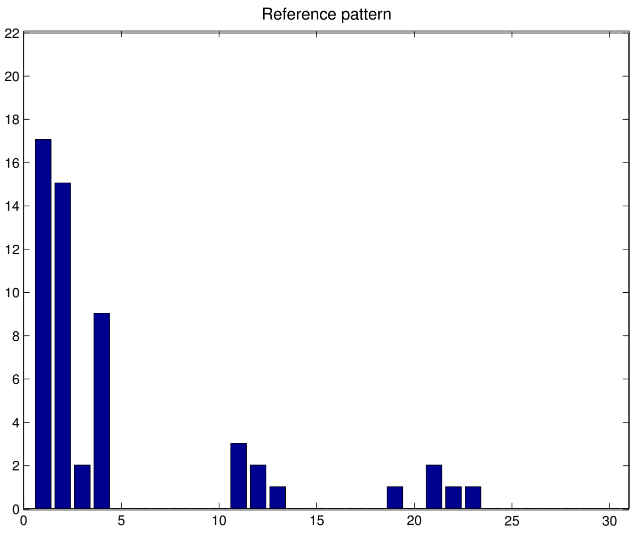

Figure 5: *Clustering and histogram matching: example of a histogram derived from a* $3 \times 10$ *SOM.*

The last row of the distortion matrix constitutes the distance function, which denotes the edit distance between the reference pattern and a best-matching substring ending at the respective column. After the selection of the *M* best matches, the respective start positions are determined by backtracking along the stored pointers. In Figure 4, the two best matches that would have been picked by the match selection algorithm are emphasized in boldface.

### 3.4. Clustering & Histogram Matching

Unlike all the above matching variants, *clustering & histogram matching* ignores the chronological order of the frames in a pattern. A self-organizing map is used to cluster the feature vectors into discrete units exactly as in the previous algorithm. However, instead of looking at the sequences of index numbers, this algorithm produces a histogram for the selected reference pattern, where each unit corresponds to one bin (Figure 5). The counts reflect the number of times each unit was chosen as a best-matching unit within the given pattern. A similar technique has been suggested in [7].

Pattern matching is carried out by comparing the reference histogram with histograms representing all possible test patterns of equal length along the entire file.[3] The distance between the reference pattern and a test pattern is determined by interpreting the histograms as vectors and taking the Euclidean distance.

[3]By restricting the test patterns to a uniform length, the respective histograms can be computed very efficiently: histograms of adjacent test patterns can be transformed into one another by deleting only one count and inserting another.

## 4. GRAPHICAL USER INTERFACE

The current prototype version of Soundspotter has been implemented in Matlab® and is controlled by a simple graphical user interface (Figure 6). It allows the user to perform a query by example based on any of the implemented algorithms and retrieve the corresponding matches straightforwardly.[4]

On opening an audio file ("Open"), feature extraction is called automatically, and a waveform plot is displayed. The user can play back the entire file or select a particular passage either by clicking in the waveform display or by entering the boundaries explicitly into a pair of text fields. The selection can be monitored using "Play selection," and the boundaries can be adjusted as desired. A pull-down menu allows the user to choose one of the matching algorithms described above. The maximum number of matches to be retrieved is specified in additional text field. "Retrieve matches" invokes the retrieval process. The distances between the selected reference pattern and the test patterns are computed along the entire file. The retrieved matches are indicated in the waveform plot by triangular markers, and a list of the retrieved intervals is printed in the Matlab command window. The retrieved patterns can be monitored one after the other (ranked by their distance to the reference pattern) using "Play matches."

## 5. EVALUATION

Up to now, the system has been subjected to various informal trials as well as a limited quantitative evaluation. In this section, we discuss the results of two comparative tests that have been performed on a recording of the pop song "Fields of Gold" by Sting. The song has a total length of 3 min 39 s. In the first test, we queried for the "rim click"[5] sound that occurs twice in each bar, 182 times altogether (reference passage: 2.19 s–2.22 s). Although it is embedded in different acoustic contexts throughout the song, it is relatively easy to detect because of its characteristic broadband spectrum and its significant power. The short duration of the sound requires a fine time resolution to obtain precise matches; we used a frame rate of 100 Hz. In the second test, we queried for the title phrase "fields of gold," which occurs at the end of each verse, 10 times overall (reference passage: 33.6 s–34.8 s). Table 1 reports the results in terms of true positives (TP) and recall (*R*).

*Recall* is a common evaluation measure in information retrieval, which indicates the fraction of the relevant patterns that has been retrieved [21]. Recall is defined as

$$R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \tag{10}$$

where TP is the number of true positives, i.e. correctly retrieved items, and FN is the number of false negatives, i.e. relevant items that have not been retrieved. Recall is typically used in conjunction with *precision*, which measures the fraction of the retrieved patterns that is relevant. Precision and recall can often be traded off, i.e. one can achieve high precision and low recall or the other way round. For the values reported in Table 1, the number of retrieved patterns was adjusted to the number of relevant patterns, so that precision and recall were equal.

[4]Caveat: the Matlab implementation has not been optimized for speed; larger sound files (e.g. a typical pop song) require a considerable amount of computation time.

[5]A specific drum sound, which is produced by laying the stick across the snare drum and striking the rim of the snare.

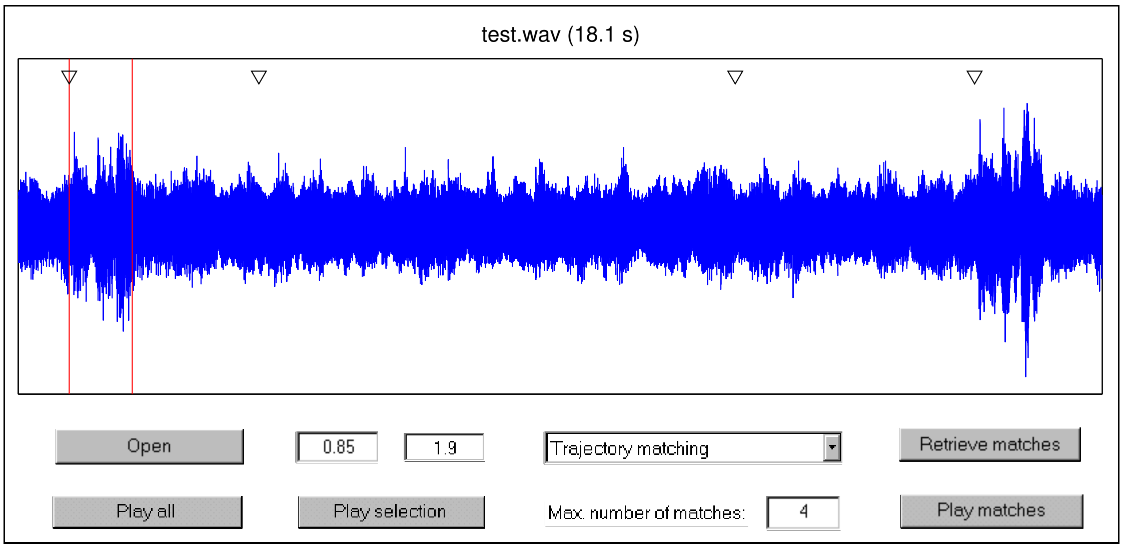

Figure 6: *Graphical user interface of Soundspotter.*

Table 1: *Quantitative evaluation of different sound spotting algorithms. The number of retrieved patterns was adjusted to the number of relevant patterns, i.e. 182 for the rim click and 10 for the title phrase.*

| Algorithm | Rim click | | Title phrase | |
|---|---|---|---|---|
| | TP | *R* | TP | *R* |
| Trajectory matching | 175 | 0.96 | 10 | 1.00 |
| Dynamic time warping | 177 | 0.97 | 10 | 1.00 |
| Mapping & traj. matching | 57 | 0.31 | 4 | 0.40 |
| Clustering & string matching | 134 | 0.74 | 4 | 0.40 |
| Clustering & hist. matching | 118 | 0.65 | 7 | 0.70 |

For both queries, the "direct" matching methods achieve a considerably higher recall (close to or equal 1) than the SOM-based methods. The self-organizing map only seems to deteriorate the retrieval performance as long as the task is relatively straightforward. However, there may be situations where the mapping leads to interesting and useful results. The mapping & trajectory matching algorithm for instance managed to retrieve an instrumental phrase when a corresponding vocal phrase was given as the reference pattern. In order to assess the performance differences between the plain trajectory matching and the computationally more demanding dynamic time warping, further tests are required.

Additional retrieval experiments querying for melodic phrases revealed that the current implementation of Soundspotter is not suitable for melody retrieval because the MFCC feature vectors do not capture enough information about the pitch content, rather, they characterize the broad shape of the spectrum.

## 6. CONCLUSIONS AND FUTURE WORK

In the preceding sections, we explained the architecture and the underlying algorithms of the audio retrieval system Soundspotter and discussed initial evaluation results. These results indicate that in most situations, the plain trajectory matching algorithm is not only the computationally most efficient method, but also the most successful one. We assume that alternative algorithms such as clustering & histogram matching and mapping & trajectory matching may produce favourable results in specific situations, which we will explore in further tests with a particular focus on electroacoustic music.

The most suitable algorithms will be implemented in plug-in for Ina-GRM's software *Acousmographe* [22], a tool for creating graphic representations of sound, e.g. graphic records of electronic music. The idea is to facilitate the creation of such representations by automatically detecting multiple occurrences of selected sound "events."

## 7. ACKNOWLEDGEMENTS

Thanks to Karin Höthker and Ralf Schoknecht for their valuable suggestions.

## 8. REFERENCES

[1] Jonathan Foote, "An overview of audio information retrieval," *Multimedia Systems*, vol. 7, pp. 2–10, 1999.

[2] George Tzanetakis, *Manipulation, Analysis and Retrieval Systems for Audio Signals*, Ph.D. thesis, Princeton University, 2002.

[3] Jonathan Foote, "Content-based retrieval of music and audio," in *Conference on Multimedia Storage and Archiving Systems II*, 1997, Proceedings of SPIE, pp. 138–147.

[4] Erling Wold, Thom Blum, Douglas Keislar, and James Wheaton, "Classification, search, and retrieval of audio," in *Handbook of Multimedia Computing*, Borko Furht, Ed., chapter 10. CRC Press, 1999.

[5] George Tzanetakis and Perry Cook, "Audio information retrieval (AIR) tools," In ISMIR [23].

[6] Tong Zhang and C.-C. Jay Kuo, "Content-based classification and retrieval of audio," in *Conference on Advanced*

*Signal Processing Algorithms, Architectures, and Implementations VIII*, San Diego, California, July 1998, vol. 3461 of *Proceedings of SPIE*.

[7] Michael Casey, "General sound classification and similarity in MPEG-7," *Organised Sound*, vol. 6, no. 2, pp. 153–164, Aug. 2001.

[8] Keith D. Martin, *Sound-Source Recognition: A Theory and Computational Model*, PhD thesis, MIT, Cambridge, Massachusetts, June 1999.

[9] Malcolm Slaney, "Auditory Toolbox Version 2," Interval Technical Report 1998-010, Interval Research Corporation, Palo Alto, California, 1998.

[10] Beth Logan, "Mel frequency cepstral coefficients for music modeling," In ISMIR [23].

[11] Eberhard Zwicker and Hugo Fastl, *Psychoacoustics: Facts and Models*, Springer, 1990.

[12] Laurence Rabiner and Biin-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, 1993.

[13] Richard Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

[14] Giovanni De Poli and Paolo Prandoni, "Sonological models for timbre characterization," *Journal of New Music Research*, vol. 26, no. 2, pp. 170–197, 1997.

[15] Petri Toiviainen, "Optimizing self-organizing timbre maps: Two approaches," in *Music, Gestalt, and Computing: Studies in Cognitive and Systematic Musicology*, Marc Leman, Ed., vol. 1317 of *Lecture Notes in Computer Science*, pp. 337–350. Springer-Verlag, Berlin, Heidelberg, 1997.

[16] Christian Spevak, Richard Polfreman, and Martin Loomes, "Towards detection of perceptually similar sounds: investigating self-organizing maps," in *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*, York, Mar. 2001, SSAISB, pp. 45–50.

[17] Teuvo Kohonen, *Self-Organizing Maps*, Springer, third edition, 2000.

[18] Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas, "SOM Toolbox for Matlab 5," Tech. Rep. A57, Helsinki University of Technology, Apr. 2000.

[19] Petri Toiviainen, "Optimizing auditory images and distance metrics for self-organizing timbre maps," *Journal of New Music Research*, vol. 25, no. 1, pp. 1–30, 1996.

[20] Graham A. Stephen, *String Searching Algorithms*, vol. 3 of *Lecture Notes Series on Computing*, World Scientific Publishing, Singapore, 1994.

[21] Christopher D. Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, MA, 1999.

[22] INA-GRM, *Acousmographe*, INA-GRM, Paris, Feb. 2000.

[23] *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Plymouth, Massachusetts, Oct. 2000.