Real Time Tracking of Musical Performances

Antonio Camarena-Ibarrola* and Edgar Chávez

Universidad Michoacana de San Nicolás de Hidalgo
Morelia, Mich., México
{camarena,elchavez}@umich.mx

Abstract. Real time tracking of musical performances allows for implementation of virtual teachers of musical instruments, automatic accompanying of musicians or singers, and automatic adding of special effects in live presentations.

State of the art approaches make a local alignment of the score (the target audio) and a musical performance, such procedure induce cumulative error since it assumes the rendition to be well tracked up to the current time. We propose searching for the k-nearest neighbors of the current audio segment among all audio segments of the score then use some heuristics to decide the current tracked position of the performance inside the score.

We tested the method with 62 songs, some pop music but mostly classical. For each song we have two performances, we use one of them as the score and the other one as the music to be tracked with excellent results.

Keywords: entropy, index, proximity.

1 Motivation

Real time tracking of musical performances consists in establishing the position of the current short segment of audio of a musical rendition of a test song as is being played in relation to the score (the target audio). Among the possible applications of real time tracking of musical performances we will briefly explain three of them: The virtual music teacher, Automatic accompanying of single players or singers, and automatic adding of special effects in live presentations.

For the virtual music teacher, the score is a well played musical piece interpreted by a trained musician. The music to be tracked is that generated by a student. Normally, a music teacher handles only one student at a time, as the student plays his instrument, the teacher gives him correction indications or gives his approval at the end. A virtual teacher should do the same, it must use the audio signal generated by the student, process it in real time and provide indications to the student accordingly. Of course, the great advantage of a virtual teacher is that it may multiply easily to manage many students simultaneously.

* Corresponding author.

G. Sidorov et al. (Eds.): MICAI 2010, Part II, LNAI 6438, pp. 138–148, 2010.

For automatic accompanying of a singer or a single player of an instrument, the score is an audio signal generated by the single player or singer without accompaniment (e.g. without orchestra) and the music to be tracked is a live performance of the same musician. The system plays the accompanying music and at the same time generates actions (e.g. To introduce delays in the background music). This application can be thought of as an intelligent Karaoke.

For the automatic adding of special effects application, the score is the audio signal recorded during a session of practice of the event, the tracked audio signal is captured in the live event. Examples of actions are turning lights on or off, playing sound effects or even launching fireworks at specific instants marked in the score.

2 Related Work

Real time tracking music systems are characterized by two aspects; The features extracted from the audio signal, and the technique used to align the score with the performance. Features are suppose to have every piece of audio in both the score and the performance characterized. In [1] the dynamic beat period of the signal is determined. In [2] the energy, delta energy, zero crossings, fundamental frequency and delta fundamental frequency are the relevant features extracted from the signal for audio tracking purposes. In [3,4], the global pitch, chroma values, cepstral flux and the spectrum were the collection of features chosen for tracking performances on line.

Once the signal's feature set has been defined, an alignment technique is normally used to relate each time instant within the musical performance to the corresponding time in the score. The classic approach to align two sequences is known as *Dynamic Time Warping* (DTW) [5]. DTW consists of finding the optimal warping function which states how one of the sequences should be shortened or stretched to reduce the differences between both sequences down to a minimum. DTW was originally designed to align two sequences that are both known *a-priori*. For our purpose (tracking musical performances in real time) only one of the sequences is known *a-priori* (the score), the alignment has to be performed as the other sequence arrives (the performance). Recently, in [6] and [7] a variation of DTW, an "on-line time warping" was proposed as an adaptation to allow for tracking musical performances on-line, the onsets of tones and the increases in energy in frequency bins were used as features. The on-line time warping algorithm attempts to predict the optimal warping path by partially filling the matrix of costs with windows of a fixed size (such size is a free parameter of the algorithm), if the minimum cost is found in the rightmost column of the window, then the filling advances column-wise, if it is found in the downmost row of the window, then the filling of the costs' matrix advances row-wise. The algorithm advance both in row and column when the minimum cost is found in the pseudo-diagonal. The real diagonal is not really known since one of the series length (the musical performance under tracking) is unknown. The "on-line time warping" algorithm toggles from row to column and viceversa when too many times the algorithm has incremented the row number (or incremented the column number to many times in a row), the maximum number of times a decision

may be repeated is another free parameter of the algorithm. In practice, the "on-line time warping" algorithm tends to deviate from the optimal path since the error is cumulative, this algorithm may completely loose track of the music.

In [2] Hidden Markov Models (HMMs) were used as the alignment tool. An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [8]. Finding HMM's parameters given a sequence of observations is a problem known as training, this problem is solved by the Baum-Welch algorithm [9]. Using an HMM (e.g. for tracking purposes) is the so called "evaluation problem" which is solved by the forward or the backward procedure [10]. Finally the "Viterbi" algorithm is used to try to discover the hidden part of the model (e.g. which state should be connected to which states) unfortunately some critical decisions has to be made by the designer of an HMM, mainly the number of states.

Since alignment is an optimization process, one interesting proposal is to use particle filters and swarm optimization to produce the alignment, as described in [1].

The rest of the paper is organized as follows: In the next section we explain our novel technique for real time tracking of musical performances, there we provide details both for the feature extraction module (audio-fingerprint determination) and for the proximity index that we use instead of any alignment method (the traditional approach for tracking musical performances). In Section 4 we describe the test set, the experiments performed and results obtained, based on which we arrived to some conclusions written in Section 5 where we also propose some future extensions to this work.

3 Our Proposal

State of the art approaches use only local information for tracking performances on-line. This implies a cumulative error, a misalignment in time t_0 cannot be recovered in any $t > t_0$. The inductive hypothesis for dynamic time warping or hidden Markov models is that the alignment is correct from all the previous time frames and use this information to align the current frame. Even if such hypothesis allows for designing of fast algorithms, it is more natural to assume nothing about the past, thus allowing recovery from any possible previous error.

We propose in this paper to improve real time tracking of audio by the novel idea of using a proximity index instead of an alignment tool, using not only local but global information. We combine this idea with the use of an audio-fingerprint of our own design. This two improvements create a very powerful tool able to recover from past alignment errors, it is very fast and even tolerates noise on both the score and the online rendition side.

As we explained in Section 2, any musical performance tracking system has to extract features from the audio signal, we will now explain how we process the signal to determine the more convenient audio signature, the Multi-Band Spectral Entropy Signature which is by the way of our own design and was adapted for the issue of this paper.

3.1 Multi-Band Spectral Entropy Signature

In [11], the extraction of a very robust audio-fingerprint based on instantaneous entropy computed in frequency bands was used for retrieving music by content. The same audio signature was successfully used for automatic radio broadcast monitoring in [12] with excellent results. We adapted our Multi-Band Entropy Signature (MBSES) for the issue of tracking musical performances on-line, the adapted MBSES is determined as follows:

1. The signal is processed in frames of 185 ms, this frame size ensures an adequate time support for entropy computation. The frames are overlapped by 3/4 (75%), therefore, a feature vector will be determined every 46 ms approximately
2. To each frame the Hann window is applied and then its Discrete Fourier Transforms is determined.
3. Shannon's entropy is computed for the first 24 critical bands according to the Bark scale (frequencies between 20 Hz and 7700 Hz). To compute Shannon's entropy, equation 1 is used. σ_{xx} and σ_{yy} also known as σ_x^2 and σ_y^2 are the variances of the real and the imaginary part respectively and $\sigma_{xy} = \sigma_{yx}$ is the covariance between the real and the imaginary part of the spectrum.

$$H = ln(2\pi e) + \frac{1}{2} ln(\sigma_{xx}\sigma_{yy} - \sigma_{xy}^2) \tag{1}$$

4. For each band, decide if the entropy is increasing or not (e.g. compare with previous frame's band). Equation (2) states how the bit corresponding to band b and frame n of the signature is determined using $H_b(n)$ and $H_b(n-1)$ (The entropy values of frames n and $n-1$ for band b respectively). Only 3 bytes for each 32 ms of audio are needed to store this signature.

$$F(n,b) = \begin{cases} 1 \; if \;\; [H_b(n) - H_b(n-1)] > 0 \\ 0 \; Otherwise \end{cases} \tag{2}$$

After computing the MBSES in each overlapped audio frame the song is represented as a binary matrix with 24 rows (24 bands of Bark) and a number of columns that depends on the duration of the song.

The MBSES corresponding to approximately one second of audio (1110 ms to be precise) is a binary matrix of 24 rows and 24 columns. A single column corresponds to 46.25 ms. The MBSES of the score can be seen as a sequence of 24x24 binary matrixes. Our purpose is to search the current one-second of audio in the online rendition in every position of the score.

The training database stores the audio-fingerprint of each one-second segment and the location of the segment inside the score. We will search for the k-nearest neighbors (KNN) of the current one second segment of the rendition in the score song, and will select among those the closest in time. The above procedure is necessary because it can be the case the current segment correspond to a chorus or a repeating segment of the score song.

The next step consist in speeding up the k-nearest neighbor searching, which can be done using a metric index.

3.2 Metric Indexes

A metric index organize a data set equipped with a distance function (a metric space) to quickly answer proximity queries. We are interested in quickly answering KNN queries in the metric space defined by all the possible segments of the score song. For the proof of the concept detailed in this work we selected the Burkhard-Keller tree or BK-tree implemented in the SISAP library [13,14]. This data structure is well suited for our task.

Fig. 1 shows a BK-tree built from an hypothetical 14-second MBSES shown at the top of Fig. 1, Fourteen binary sub-matrixes are extracted and added to the BK-tree, each correspond to one second of audio and are referred to as a,b,..n. For the sake of space in Fig. 1 a,b,..n are 6x6 binary matrixes instead of the real 24x24 matrixes that result from extracting MBSES from of one-second excerpts.

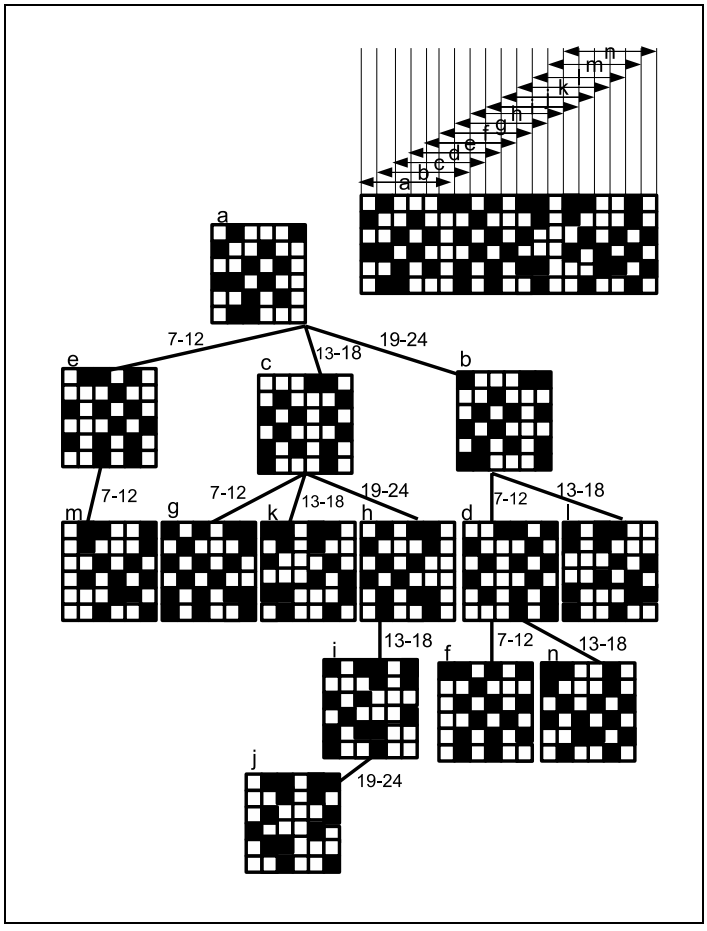

Fig. 1. One-second excerpts are indexed using a BK-tree

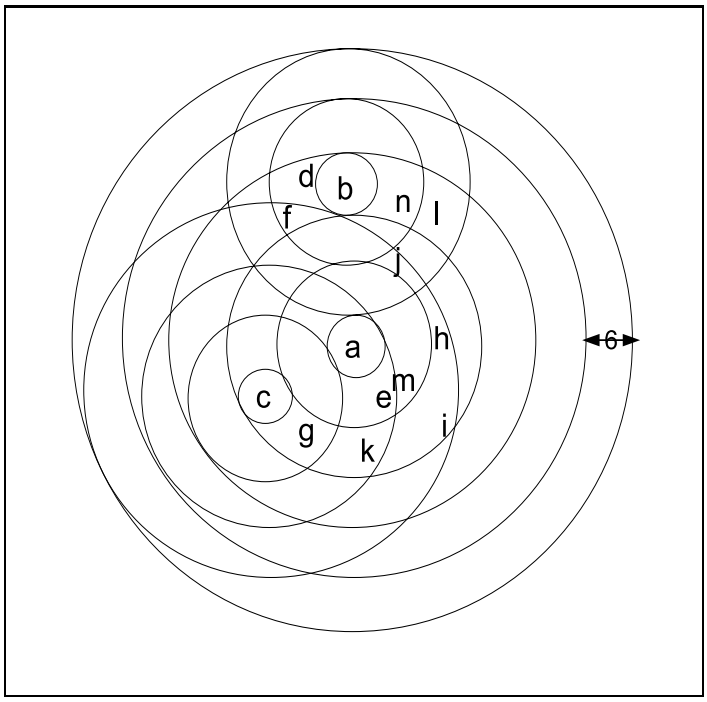

Fig. 2. Metric space related to the BK-tree shown in Fig. 1

The first submatrix read (Matrix a) becomes the root of the BK-tree, the second one is compared to the root using the Hamming distance which turns out to be 21 (21 different bits), so it becomes a son of the root under the link 19-24 since $19 \leq 21 \leq 24$, the third matrix read (matrix c) is next inserted, so it is compared to the root, its Hamming distance to the root is 17 so it becomes another son of the root under link 13-18 since $13 \leq 17 \leq 18$. To insert matrix d, it is first compared to the root, since its distance to the root falls in the rank 19-24 and there is already a node there (matrix b), then matrix d has to be compared with matrix b, it is then added as a son of matrix b under link labeled 7-12 since $7 \leq Hamming(b, d) \leq 12$ (In fact, Hamming(b,d)=12). The rest of the matrixes are inserted in the BK-tree the same way.

For the BK-tree in the example of Fig. 1 the bucket size in the nodes is $bs = 6$, that is, the maximum number of children of a node, also, the ring width is $s = 6$, that is the range of distances grouped in a single child of a node. To understand that, observe Fig. 2, the root of the BK-tree from Fig. 1 (sub-matrix a) is located at the center of the metric space. Six rings are shown around sub-matrix "a", each one has a width of $rw = 6$. Any sub-matrix must lie inside one of these rings since the maximum Hamming distance between two $6x6$ binary matrixes is 36. Any sub-matrix that is a descendant of a son of the root lies inside the same ring, therefore, sub-matrixes "e" and "m" lie inside the same ring, also sub-matrixes "c", "g", "k", "i", "h", and "j" conform a BK-subtree and lie inside the same

ring. Any node that is a son of the root is itself the root of a BK-subtree, then they are at the center of another set of rings, observe for example sub-matrixes "n", "d" and "f", they lie inside the same ring centered at sub-matrix "b" and since these sub-matrixes along with sub-matrix "l" conform a BK-subtree they all lie inside the same ring centered at "a". To find the nearest neighbors in the Bk-tree, the rings whose distance to the center is less and less near the distance between the query and the center.

4 Experiments

For 62 songs or masterpieces we were able to obtain another musical rendition of it, for example for *Beethoven's Symphony Number 5 in C minor* performed by the *Berliner Philharmonische Orchester* conducted by Karajan we obtained the *Viena Philarmonic Orchestra* version conducted by Kleiber, such collection of songs and masterpieces conformed the data set for our tests.

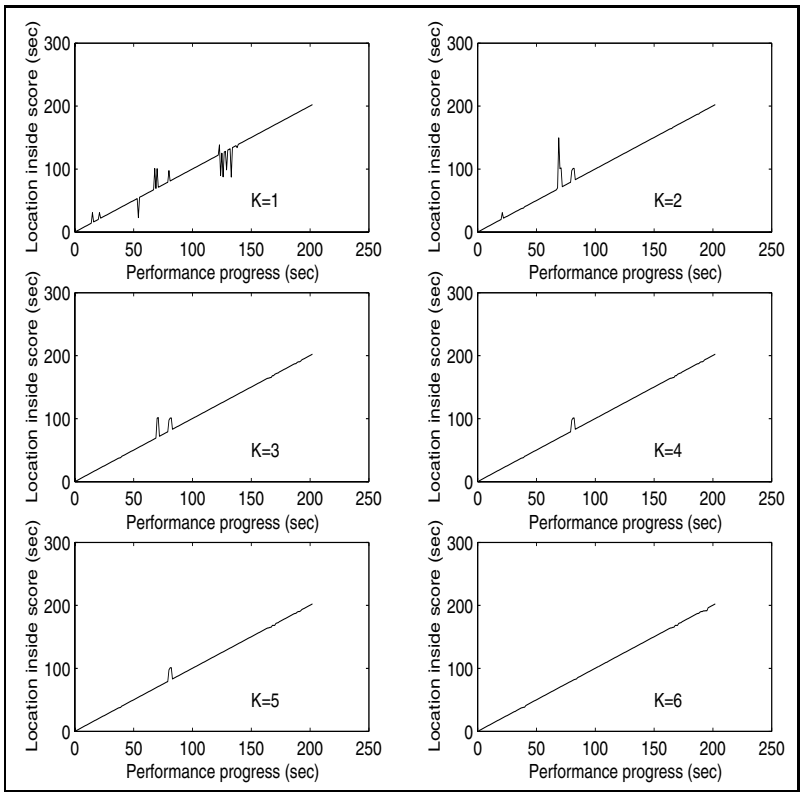

Fig. 3. Tracking of a performance of "All my loving" (The Beatles) for different values of K (Number of nearest neighbors considered). Some tracking failures can be observed for low vales of K.

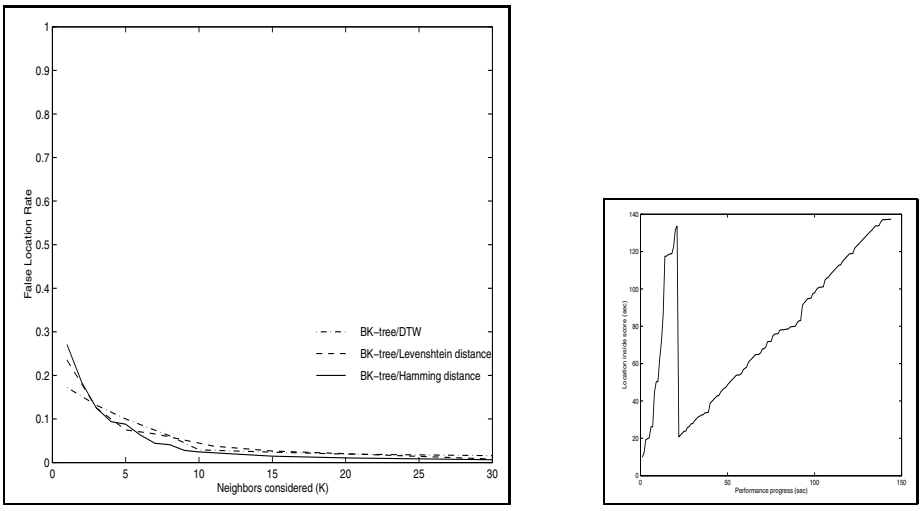

Fig. 4. The False Location Rate falls as K increases (left). An example of the recovery after a false start in *The Nutcracker* using only one nearest neighbor (right).

For every pair of musical performances of each song or masterpiece in our collection, we took the first one as the score and the second one as the rendition. We extracted and indexed the MBSES of the score using the BK-tree with a bucket size of $b = 24$ (the maximum number of children of a node) and a ring width of $s = 24$ (the range of distances grouped in a single node).

For the online tracking we proceed as follows: For every second of audio we extract its MBSES and then search for the K-nearest neighbors of it in the score using the closest one in time. We then read the next second of audio and repeat the process until the end of the performance. Fig. 3 shows the tracking of a performance of the Beatles' song ("All my loving"), the curves shown in Fig. 3 informs for each second of the performance where it was located in the score, normally, if both the score and the performance are very similar in duration as they are in the case of Fig. 3. We observe some peaks in Fig. 3 for low values of K, these peaks are in fact tracking failures since they imply a jump from a tracking position to another position that is not very near, we call this "false locations". We repeated the tracking for different values of K and observe that such false locations disappear as K increases, in the example shown in Fig. 3 the tracking failures disappear for $K = 6$. Observe that the system recovers almost immediately from false locations so they must not be considered has if the system lost track of the music. An extreme example of this recovery is shown in Fig. 4 (rigth).

Repeating the process described above for the rest of the musical performances of our collection we determined the False Location Rate (FLR) for values of K varying from 1 to 30. A false location is considered to appear when the short

audio signal of a one-second length is located too far away in time from the previous tracked piece of audio. The FLR is computed using equation 3

$$FLR = \frac{False\ locations}{False\ locations + True\ locations} \tag{3}$$

In Fig. 4 (left) the FLR resulting from tracking every second musical performance of our collection and then varying the number of nearest neighbors considered (K). We indexed the metric space representing the score using three different distances. The first was the Hamming distance counting the number of bits different in each 24x24 matrix. The second was the DTW distance, and the third was the Levenshtein distance measuring the minimum number of insertions, deletions and substitutions to make the matrices equal; we used a substitution weight of 2, and the insertion and deletion weights were normalized between 0 and 1 using the Hamming distance between frames. We used up to 30 nearest neighbors for the global alignment. Using either of the three distances we were able to lower the false location rate arbitrarily. Since the Hamming distance is the cheaper one of the three, we believe it should be the one we should select to work.

4.1 Avoiding False Locations

For all practical purposes false locations should not lead to execute actions, that is they should be avoided. False locations appear due to the fact that none of the K nearest neighbors occur near the current tracking position in time. A simple way to prevent undesirable actions to take effect as a result of a false location occurrence can be stated like this:

When none of the K nearest neighbors occur near the current tracking position then the tracking position in the score should not move as if the last short segment of audio had not been received.

Once this modification was made to our tracking system the peaks such as those in Fig. 3 no longer occur, not even for K=1.

4.2 Time Analysis

Two steps are required to find the location inside the score of a one-second segment of audio taken from the tracked musical performance. The first step consists in determining the MBSES (the audio-fingerprint) of the audio signal, the second step is the search of the K nearest neighbors of the 24x24 matrix extracted on the previous step using the BK-tree for that purpose. Using a Dual-core 1.46 GHz pentium Laptop with 1GB of memory, 130 milliseconds are needed to accomplish step 1 and only 10 milliseconds for step 2. Building the BK-tree is performed prior to the tracking process and so its timing is not critical. A real time application such as those discussed in the Motivation section would use approximately 140 ms for every second of audio tracked without any parallelization.

5 Conclusions and Future Work

We successfully performed tracking of musical performances using a robust audio-fingerprint by searching in a metric space using BK-trees as proximity indexes. This approach does not require an iterative training process as Hidden Markov Models (HMMs) where the ideal number of states is unknown. Like HMMs we take advantage of the fact that we know the score a-priori (unlike on-Line DTW). It is very important to remark that our approach is a global alignment tool not accumulating error and able to recover from false locations. A single error cannot be considered has if the system completely lost track of the music when this false location appear. Finally our approach has the additional advantage that the tracking could be started at any time, no alternative approach is capable to begin tracking when the musical performance has already started for example when the tracking system is turned on too late.

There are other alternative metric indexes that should be considered when the metric space is larger than only one song. This can be very useful when a complete collection of audio is indexed instead of a single song. We will investigate the scalability issues of metric indexes for this particular application.

References

1. Sethares, W.A., Morris, R.D., Sethares, J.C.: Beat tracking of musical performances using low level audio features. IEEE Transactions on Speech and Audio Processing (2) (March 2005)
2. Cano, P., Loscos, A., Bonada, J.: Score-performance matching using hmms. In: ICMC 1999, Audiovisual Institute, Pompeu Fabra University, Spain (1999)
3. Orio, N., Déchelle, F.: Score following using spectral analysis and hidden Markov models. In: Proceedings of the ICMC, pp. 151–154 (2001)
4. Orio, N., Lemouton, S., Schwarz, D.: Score following: state of the art and new developments. In: Proceedings of the, conference on New Interfaces for Musical Expression, National University of Singapore, p. 41 (2003)
5. Sakoe, H., Chiba, S.: Dynamic programming algortihm optimization for spoken word recognition. IEEE Transactions on Acoustics and Speech Signal Processing (ASSP), 43–49 (1978)
6. Dixon, S.: Live tracking of musical performances using on-line time warping. In: 8th International Conference on Digital Audio Effects (DAFx 2005), Austrian Research Institute for Artificial Intelligence, Vienna (September 2005)
7. Dixon, S., Widmer, G.: Match: A music alignment tool chest. In: 6th International Conference on Music Information Retrieval (ISMIR), Austrian Research Institute for Artificial Intelligence, Vienna (2005)
8. Rabiner, L., Juang, B.: An introduction to hidden markov models. IEEE ASSP Magazine 3(1), 4–16 (2003)
9. Bilmes, J.A.: A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, Department of Electrical Engineering and Computer Science U.C. Berkeley (April 1998)
10. Rabiner, R.L.: A tutorial on hidden markov models and selected aplications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)

11. Camarena-Ibarrola, A., Chavez, E.: On musical performances identification, entropy and string matching. In: Gelbukh, A., Reyes-Garcia, C.A. (eds.) MICAI 2006. LNCS (LNAI), vol. MICAI, pp. 952–962. Springer, Heidelberg (2006)
12. Camarena-Ibarrola, A., Chavez, E., Tellez, E.S.: Robust radio broadcast monitoring using a multi-band spectral entropy signature. In: 14th Iberoamerican Congress on Pattern Recognition, pp. 587–594. Springer, Heidelberg (2009)
13. Figueroa, K., Chávez, E., Navarro, G.: The sisap metric indexing library, `http://www.sisap.org/library/metricspaces.tar.gz`
14. Burkhard, W.A., Keller, R.M.: Some approaches to best-match file searching. ACM Commun. 16(4), 230–236 (1973)