AUTOMATIC ALIGNMENT OF AUDIO OCCURRENCES: APPLICATION TO THE VERIFICATION AND SYNCHRONIZATION OF AUDIO FINGERPRINTING ANNOTATION

Mathieu Ramona
IRCAM, Centre Pompidou
1, place Igor Stravinsky, 75004 Paris, France
mathieu.ramona@ircam.fr

Geoffroy Peeters
IRCAM, Centre Pompidou
1, place Igor Stravinsky, 75004 Paris, France
geoffroy.peeters@ircam.fr

ABSTRACT

We propose here an original method for the automatic alignment of temporally distorted occurrences of audio items. The method is based on a so-called *item-restricted fingerprinting* process and a segment detection scheme. The high-precision estimation of the temporal distortions allows to compensate these alterations and obtain a perfect synchronization between the original item and the altered occurrence. Among the applications of this process, we focus on the verification and the alignment of audio fingerprinting annotations. Perceptual evaluation confirms the efficiency of the method in detecting wrong annotations, and confirms the high precision of the synchronization on the occurrences.

1. INTRODUCTION

Audio identification aims at detecting occurrences of known audio tracks in an unknown audio signal or stream. A typical example is the identification of songs in a broadcast recording. An "occurrence" is defined as the presence in an unknown signal of a degraded, or modified, version of an original audio track, that remains recognizable. An audio identification system typically searches for the possible occurrences of a large collection of tracks previously "learned" in a database. Each track contained in the database is called an audio "item".

One of the main challenges of audio identification is the robustness to the possible degradations between the original item signal and the occurrence signal. Typical degradations are audio encodings (MPEG, Real Audio...), filterings, noise addition, etc. These degradations do not alter the temporal evolution of the signal. On the opposite, the so-called *dynamic* degradations, such as time-scale changes, or signal cropping, induce a loss of alignment between the original item and the occurrence. The problem answered by this paper is to estimate very precisely these dynamic degradations, in order to realign both signals. The motivations for this issue will be explained later on.

The subject of audio alignment has been covered in the past, especially in the domaine of audio-to-score alignment, which consists in aligning the audio signal of an execution of a musical piece with the score itself. Joder et al. [1] and Cont [2] both answer this problem with Hidden Markov Models coupled with a tempo model. Müller et al. [3] also propose an algorithm to align a performance of a song on the score, based on a chroma representation. However, these approaches all deal with the alignment with respect to a score, i.e. a symbolic representation of music, whereas the problem here is the alignment of two audio signals. Müller et al. [4] also applied the chroma representation for the matching of two musical interpretations (i.e. two audio signals) of the same piece. But their contribution only focuses on the detection of these matches, not on their temporal alignment.

The method we propose here for the alignment of audio occurrences is based on an original scheme, derived from the audio fingerprinting technique. Indeed, in [5], Casey et al. rank the problems of *audio queries* by order of similarity between the query and the reference. While genre classification and cover song detection connect very different audio signals from their *semantic* musical content, audio fingerprinting is described as "identifying a recording in the presence of a distorting communicating channel". Audio fingerprinting is in fact one of the main methods (along with audio watermarking) to perform audio identification. It consists in computing perceptually relevant numerical codes (the so-called *fingerprints*) that characterize the signal of the audio items of the database. When performing identification, similar codes are computed from the unknown signal, are compared to the codes stored in the database. This similarity search allows to identify the occurrences of the items in the unknown signal.

This paper will show how an audio fingerprinting technology can be exploited for the automatic alignment of audio occurrences, and consequently, for the correction and the refinement of ground truth annotations for audio identification evaluation. We will explain thoroughly in Section 2 why there is a need for such an automated process of annotation verification in the context of audio fingerprinting evaluation. We then present in detail the context and the terminology of the problem in Section 3, before presenting the alignment process in Section 4. A first application of this process on the Quaero audio identification corpus is presented and commented in Section 5, along with some audio examples. Then we comment on the applications and perspectives of this contribution in Section 6.

2. AUDIO FINGERPRINTING EVALUATION

Research on audio fingerprinting has been very active in the last ten years, and commercial applications based on this technology are numerous. However, contrary to other subjects in audio indexing, there is no consensus on the evaluation protocol, nor any public evaluation campaign for audio fingerprinting. One might argue that the main reason that the main commercial system already work very well. However, most companies actually have not published results of their systems on a large public database. The Quaero project has brought a first step in this direction with its first evaluation campaign for audio identification that was held in September 2010. Following this campaign, a collaborative paper from the participants was submitted [6], that discusses the issues of audio fingerprinting evaluation and proposes a public evaluation framework (available at http://pyafe.niderb.fr/).

Related works on evaluation The main obstacle in audio fingerprinting evaluation lies in the cost of collecting a large real-world corpus, with reliable and precise annotations. The Quaero evaluation campaign is based on a musical track monitoring scenario and is based on a corpus, provided by Yacast[1], containing real-world radio broadcast audio data. Cano et al. [7] also evaluate the identification rate of musical tracks on 12h broadcasted audio streams, but most of the authors measure the robustness of the fingerprinting code on audio items altered by typical audio degradations. Wang (Shazam) [8] evaluates the recognition rate over 250 items, after a GSM compression step and under the addition of noise with controlled SNR. Additive noise over clean items is also used by Weinstein and Moreno (Google) [9]. Haitsma and Kalker (Philips) [10] also propose a protocol involving a large collection of audio degradations (MP3 or Real Media encoding, equalization, various filters, time-scale modification, etc.) applied to four clean items.

Artificial vs Real-world distortions Artificial alterations are indeed preferred in the literature for several reasons: the corpus only consists in a collection of audio tracks and is thus much more easy to collect, the alterations are easily applied, and, most of all, they can be precisely controlled. It is then possible to study the evolution of the robustness with regard to the SNR, or the time-scaling factor. On the other hand, these artificial degradations do not reflect real-world situations. Indeed, in a typical case of broadcast emission, any musical track is generally slightly time-scaled, dynamically compressed, affected by additional noise, and subject to MP3-like encoding or digital-analog conversion.

Real-world annotation Nevertheless, a corpus based on real-world data needs the human annotation of all the occurrences of the items in the stream, and most of the degradation process is unknown to the experimenter. Annotating the start and end times of an occurrence is easy, but the annotation of a large scale corpus will generally imply a low precision (even a one second precision is ambitious). Moreover, it is almost impossible to determine manually the time-scale factor. Finally, a certain amount of mistakes is expected from manual annotation, especially when the item collection involves different edits of the same song.

The method proposed here is an ideal mean for verifying and improving such manual annotations, as we will show in this paper. The detection of missing occurrences is not in the scope of this article, but the detection of wrongly annotated occurrences proves very efficient. The alignment of the occurrence signal with the original item signal (and thus of the fingerprint codes computed from both signals) allows the application of several evaluation schemes used on artificial corpuses. We hope that this new type of annotation post-processing will encourage evaluations of audio fingerprinting techniques on real-world corpuses.

3. CONTEXT

The problem that is raised here is similar to that of audio identification, as explained before: occurrences of known audio items are to be found and located in an audio stream. However, we seek here a much more precise result, which is made possible by the use of prior information, unavailable in a common audio identification scenario. We suppose here that the processed signal has been previously annotated, either manually or during the production of an artificial corpus. In both cases the annotation may be unprecise, and only consists of a collection of item occurrences in the audio streams, characterized by the item index in the database, and the approximate start and end times in the stream. The annotated times can be wrong by a few seconds, the error being compensated by a larger analysis scope.

An audio item can even be a sample of a song (for instance the chorus), instead of the whole track, and its exact position in the song unknown. This situation remains equivalent to using the whole song, as long as the scope of analysis includes the whole song. However, this implies, since a musical track structure is generally repetitive, that the excerpt, or a part of it, may be detected several times in the scope of analysis. Such repetitions must be discarded from the alignment process, in order to focus on the most reliable occurrence.

The item occurrence in the stream can be affected by typical audio distortions, either artificially generated or sampled from a real-world corpus. These distortions fall into two categories:
Static distortions: do not affect the temporality of the signal, i.e. the original and distorted signals are perceived as synchronous on their whole scope. Typical examples are linear filters, equalization, amplification, analog/digital conversions, typical audio encodings (MPEG, OGG ...), noise addition, loudspeaker/microphone loop.
Temporal distortions: affect the synchronization between the original and distorted signals. The process proposed here intends to estimate precisely these temporal distortions in order to be able to correct them and reach a perfect alignment between the original and distorted signals. Temporal distortions considered here are:

- *Shifting*: in the case of frame-sequence analysis, a slight shift (of a few tenth of seconds) between the signals can induce major differences in the content of the frames. It is thus important to synchronize the start time of both signals.
- *Scaling*: for instance, radio stations very often accelerate or slow down musical tracks to fit in a live schedule.
- *Cropping*: the beginning or the end of the audio item can be absent from the occurrence.
- *Insertions*: although this distortion is less common, the item can be interrupted by another signal, and then played again from where it stopped. This induces a slight shift between the item and the occurrence, that also requires a proper correction (i.e. cutting the inserted signal).

Figure 1 sums up the temporal characteristics that we intend to evaluate in this process:

1. **ItemTime:** the time in the stream that corresponds to the beginning of the database item.
2. **StartTime:** the time, relative to the ItemTime, where the occurrence actually starts in the stream. It is positive or zero. When strictly positive, it means that a part of the item beginning is not played in the stream.
3. **EndTime:** the time, relative to the ItemTime, where the occurrence ends in the stream. It is strictly positive, and upper bounded by the ItemDuration $\times$ TimeFactor, when the item is played until its end.

[1] `http://www.yacast.fr/fr/index.html`

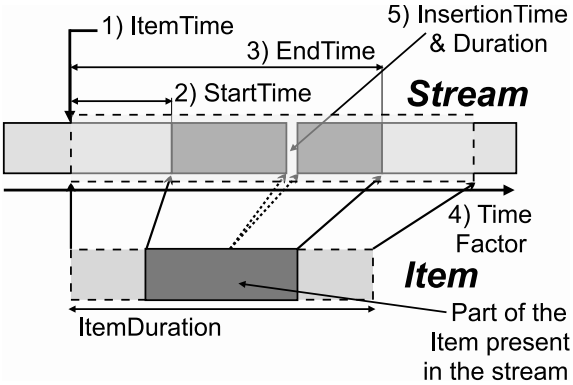


Figure 1: *Characterization of the temporal alterations encountered on the occurrence (at the top) of a given item (at the bottom).*

4. **TimeFactor:** ratio between a time relative to the ItemTime in the stream, and the corresponding time in the item signal.
5. **InsertTime & InsertDuration:** if the insertion happens in the stream (as in Figure 1), the time is relative to the ItemTime ; if it happens in the item, the time is absolute in the item. The same holds for the duration. Several insertions can be observed on a single occurrence.

The ItemDuration (also denoted D) is the duration of the full item signal. All characteristics are expressed in seconds.

4. DESCRIPTION OF THE ALIGNMENT PROCESS

4.1. Item-restricted fingerprinting

The key element of the alignment process is the application of a customized *Item-restricted* fingerprinting technique that we present here.

The normal process of audio fingerprinting is to fill a database will the fingerprint codes computed from a large collection of audio items. Each item is described by multiple code, computed at various time position in the signal, in order to be able to recognize any subset of it. In the present context, for each occurrence, the item is already known (from the annotation). So for each occurrence, a new database is built specifically, that contains only the fingerprint codes computed from this item. This constraint dramatically reduces the size of the database, and thus the search time. Consequently, the result of the search is a sequence of timestamps describing the position of the codes in the original item.

The use of a fingerprinting method ensures a sufficient robustness to static distortions observed in the stream occurrence. The fingerprint method used here is the one developed by the Ircam [11]. It is based on a double-nested Short Term Fourier Transform of the audio signal, over overlapping frames of a few seconds. The original method has recently been upgraded [12] with perceptual scales (a Bark filter-bank for the short-term FFT and a sone scale for the amplitudes of the long-term FFT). The resulting code is a real vector of 36 components.

The latter article [12] also describes an upgrade of the algorithm based on onset detection, but this part of the algorithm is not used here, and we rely on a regular frame scheme. In order to locally reduce the effect of the temporal distortions, the frame size is kept relatively short (2 s). The hop size is much shorter (50 ms), than in the "standard" fingerprint process (originally set to 0.5 s). This implies that the expected temporal shift between corresponding codes is 12.5 m, which represents a negligible portion (0.6%) of the frame size. Theoretically, any other fingerprinting method could be adapted to this item-restricted scheme, but the Ircam is prefered, precisely because it involves larger window and step sizes that most methods, and thus reduces the number of fingerprint codes per item.

The first step of the algorithm consists in computing the fingerprint codes for each item, and storing them with the corresponding timestamps. Each minute of signal generates about 1200 codes. For each annotated occurrence in the stream, a sequence of codes is computed on the scope of analysis. Then a simple nearest neighbor search ($k = 1$ neighbor) is performed among the codes of the item, to collect the resulting sequence of timestamps associated. The so-called *timestamp sequence* is denoted by a set $(x_i, y_i)_{i=1,\dots,n}$, where n is the number of frames, and x_i and y_i are respectively the time of the frame in the stream and the timestamp of the nearest neighbor in the item.

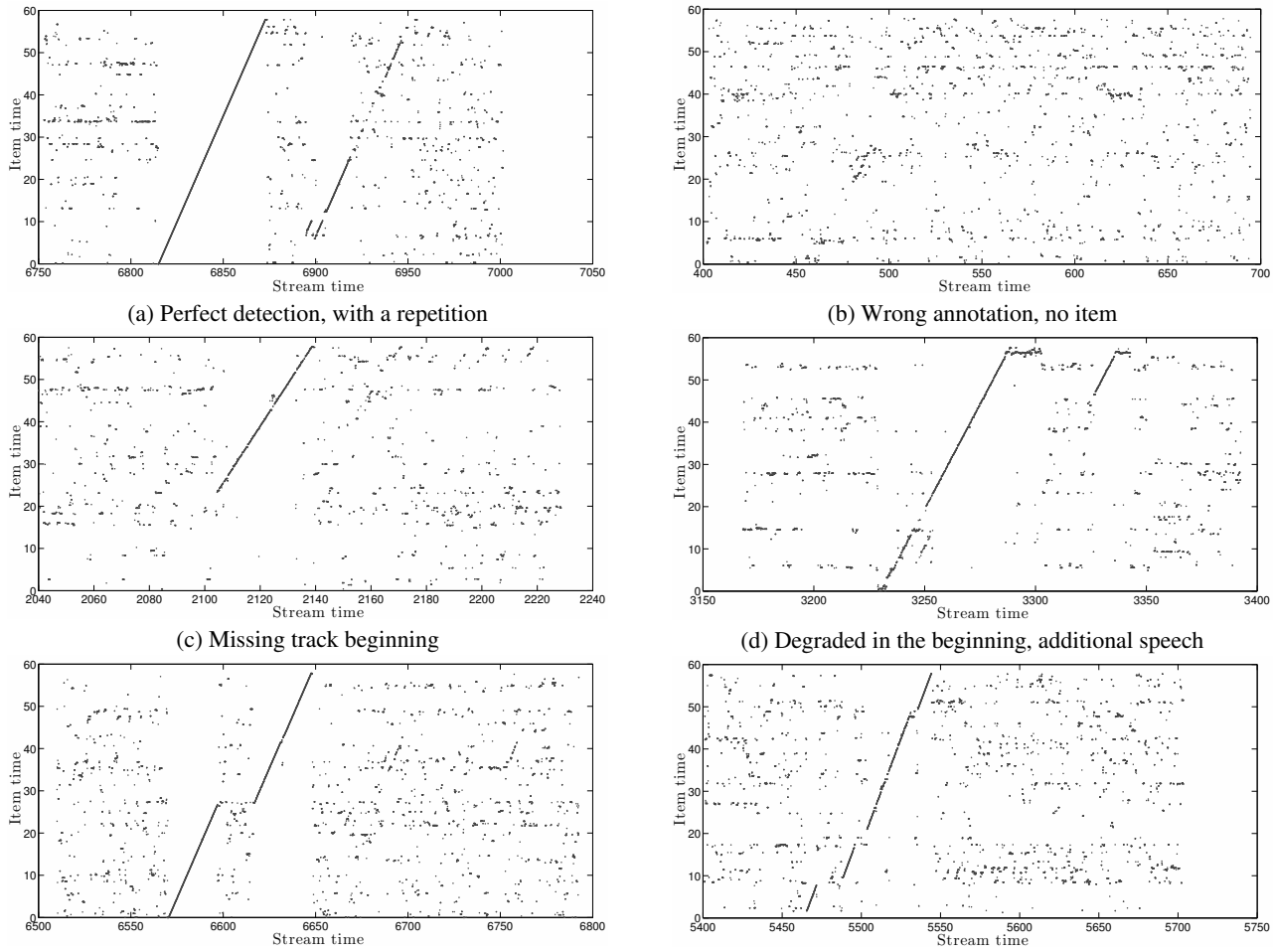
Figure 2 shows several examples of timestamp sequences. The ideal detection of the full item, illustrated by Figure 2(a), implies the presence of a solid line segment of slope close to 1, that binds the ordinates 0 and D (in our case all the audio items are 60 s long). Another fragmented line is also visible on the figure, that denotes a repetition of the item, with alterations. On the opposite, Figure 2(b) shows a clear example of wrong annotation, where the dots are randomly drawn. The dot distribution is clearly not uniform though, and shows higher densities on some constant ordinate lines. This is a simple expression of the classical phenomenon in similarity of "hubs" [13], i.e. examples that are near to all other examples in a distribution. Figure 2(c) shows an example of cropped occurrence where the beginning of the item is missing. Figure 2(d) shows a line degraded in the beginning, illustrating an example of occurrence where the first seconds are covered by another signal (possibly the radio host voice). Figure 2(e) shows a clear example of insertion of a signal snippet in the middle of the original item signal. Finally, Figure 2(f) is an interesting example of fragmentation of the line, that denotes the detection of separated chunks of the original item, probably because of an edit mismatch between the item and the stream occurrence. Indeed, radio stations frequently use specific edits of a song with structures that greatly differ from the original album edit.

These examples show that the alignment process basically consists in a segment detection algorithm in the timestamp sequence. Each segment is described by the following equation:

$$y = ax + b \qquad \forall x \in [x_{\text{start}}; x_{\text{end}}], \tag{1}$$

where

- a is the slope of the line containing the segment, that correspond to the time-factor previously introduced.
- b is the offset of the line containing the segment. Differences between the b values will indicate the presence of insertions.
- $[x_{\text{start}}; x_{\text{end}}]$ are the boundaries in abscissa of the segment. These will determine the ItemTime, StartTime, CutTime and EndTime values introduced in Figure 1.



(a) Perfect detection, with a repetition

(b) Wrong annotation, no item

(c) Missing track beginning

(d) Degraded in the beginning, additional speech

(e) Insertion in the stream occurrence

(f) Fragmented detection, edit mismatch

Figure 2: *Timestamp sequences: ordinates represent the time in the item of the nearest neighbor to the code computed from the stream at the time in abscissa. Several examples of result are shown, ranging from the ideal detection of the item (a) to the absence of detection (b).*

A Hough transform [14] could of course be used for this line detection problem. However, it is more costly than the method proposed below. Moreover, as we will show, the evaluation of the common slope is done jointly on all the segments and is therefore more robust and precise than a fusion of separate estimations from each segment.

4.2. Time-factor estimation

The first step consists in evaluating the slope a. The time-scaling is supposed constant over the whole occurrence, since a varying time-scaling induces audible distortions not acceptable to the listener. All the segments thus share the same a value.

The *point-slope* of a pair of points (x_i, y_i) and (x_j, y_j) is defined as follows:

$$a_{i,j} = \frac{y_j - y_i}{x_j - x_i} \tag{2}$$

We evaluate the distribution of $a_{i,j}$, over the pairs i, j complying with the constraint $1s < x_2 - x_1 < D$, where D stands for the item duration. The complexity of this operation is linear $O(n)$. The lower bound is set because both coordinates are discrete (as shown on the zoom provided Figure 3), which makes the point-slope less precise as the points get closer.

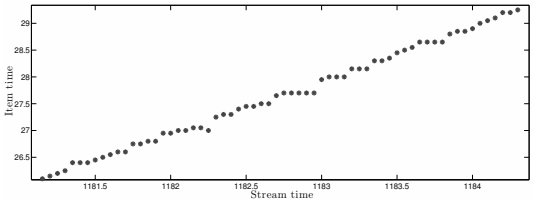


Figure 3: *Illustration of the discrete distribution of the points in the timestamp sequences.*

A histogram distribution is computed between the values 0.8 and 1.2 [2] to identify the expected maximum peak on the a value. The computation of the point-slope indeed strongly amplifies the effect of a real line on the distribution, since aligned points all

[2] considered as large bounds for reasonable (i.e. not too audible) time factors.

contribute to the same bin, whereas unaligned points contribute to different bins. Figure 4 illustrates this on the case of points shared between two lines of same slope. Even in this case, one sees that the pairs from different lines induce different slopes, and will not create local maxima in the distribution.

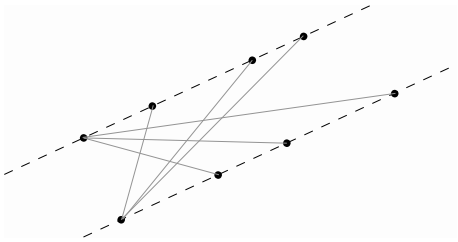


Figure 4: *Unaligned points, even from aligned sets, do not contribute to local maxima in the distribution of slopes.*

However there is in fact a "resonance" of the slopes in the very close vicinity of 1 ($|a-1| < 0.0005$) that we don't explain. In order to avoid this accidental maximum, the distribution is set to zero in this very small interval. Even if the real maximum is precisely at 1, it will spread outside this interval and will be detected. Figure 5 shows an example of the slope distribution, where the time-factor peak is clearly located.

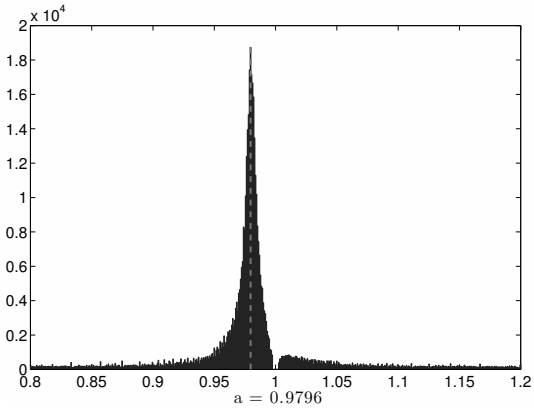


Figure 5: *Example of histogram estimated distribution of the point-slopes. The slope a (i.e. the time-factor) is estimated from the salient peak position (indicated by the dotted line).*

4.3. Offset estimation

We then estimate the *point-offsets* defined as follows:

$$b_i = y_i - a\,x_i. \tag{3}$$

Since all the lines share the same slope, then b_i is constant for all the points (x_i, y_i) on a same line. By estimating the distribution of the point-offsets, disjoint segments from the same line are gathered in the same bin, whereas segments from parallel lines (in the case of insertions or cuts) are separated.

Contrary to the time factor distribution, the search scope for the b values cannot be easily bounded. Moreover, in the case of noisy timestamp sequences, the local maxima are more spread than the peak observed on the time-factor distribution. Instead of histograms, we thus use kernel density estimation [15] (with a gaussian kernel) to get a smoother distribution and gather neighboring peaks[3]. The latter, given a specific number of bins, automatically estimates the optimal bandwidth value for the gaussian kernel. The resulting distribution is then divided by its 90% percentile, in order to fix an absolute threshold (empirically set to 10) for peak detection.

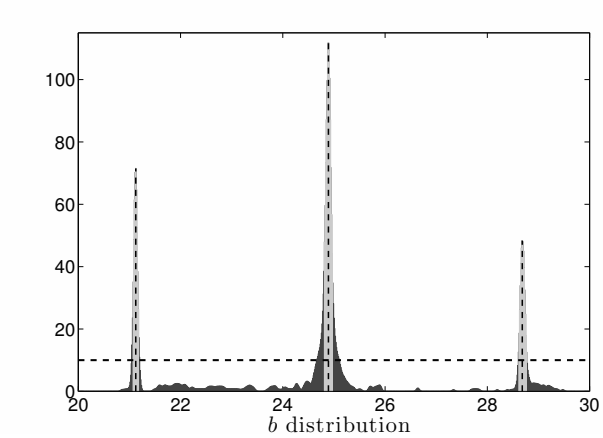


Figure 6: *Distribution of the offset values ($y_i - a\,x_i$), evaluated with gaussian kernel density estimation. Here, three line offset values (indicated by the the vertical dotted lines) are detected above the threshold (the horizontal dotted line). The light areas indicate the range of offsets associated with the peaks.*

The peak values above the threshold are iteratively selected. At each step, the distribution is set to zero in the surroundings of the peak. The kernel bandwidth estimation induces the automatic determination the 3 dB bandwidth of the peak. Finally, all the points (x_i, y_i), with their offset inside the 3 dB cut-off interval, are associated to the selected b value, i.e. to a particular line in the timestamps plane.

Figure 6 shows the result of the offset estimation process applied to the timestamp sequence shown in Figure 2(f). Three peaks are shown, that correspond to the three last segments observed on Figure 2(f) (the forth peak is outside the scope of the figure). Figure 7 shows the results of this operation on the timestamp sequence. Each gray shade represents the points associated with one of the segments.

In the case of no peak detection, the occurrence is discarded and considered as a erroneous annotation.

4.4. Segment estimation

The distribution of the points associated with the segments can be more noisy than on Figure 7, and needs post-processing. Figure 8 shows an example of result with erroneous points. Each ordinate (represented with a different shade) shows a binary signal that indicates the association (or not) of the points to the segment. In this case the segments 3 and 4 are to be discarded, and segments 1 and 2 show a few accidental points outside their boundaries. In fact any point can be wrongly associated with a line, if it is close

[3] In particular, we use the very fast and efficient implementation provided by Botev [16], but this is not essential here.

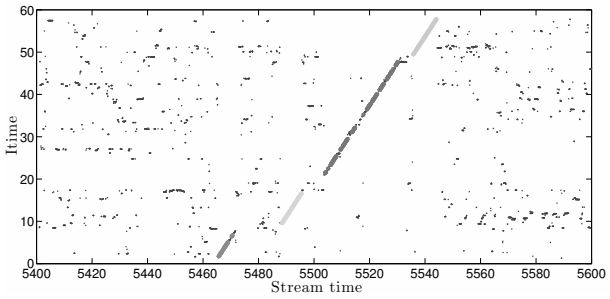


Figure 7: *Each gray scale depicts the collection of points associated with one of the segment offsets detected from the timestamp sequence of Figure 2 (f).*

to it, even if it is far outside the segment boundaries. These are discarded by applying a median filter, with a sliding window of 10 samples, on each segment binary signal. Then, the whole area between the first and the last point of each segment is assigned to it. Finally segments shorter than 5 s are discarded. The dotted boxes on the figure indicate the result of this post-processing, i.e. the estimation of the segment boundaries.

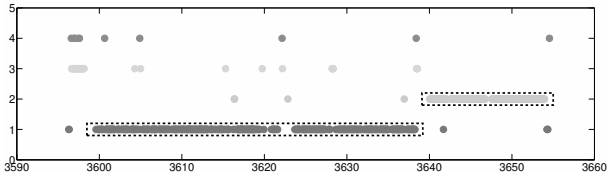


Figure 8: *Each ordinate value corresponds to an offset value, and shows a binary signal indicating the point associated to it. The dotted boxes indicate the boundaries of the segments detected after post-processing. The offset values 3 and 4 are discarded here.*

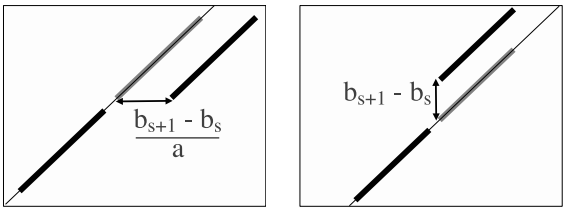
When detecting repetitions of the item (as in Figures 2(a) and (d)), only the most exact occurrence of the item is of interest, the other repetitions are accidental. A simple way to discard these repetitions is to compute the stream time distance between the segments: if a segment s_2 is shorter than s_1, then the segment s_2 is discarded if $|x_{s_1} - x_{s_2}| > 30$ s, where $x_s = -\frac{b_s}{a}$ defines the abcissa of the intersection of the line s with the zero-ordinate axis.

4.5. Estimation of the temporal characteristics

The result of the process so far is a list of S segments ; each segment s is characterized by its offset b_s and its boundaries $[x^s_{\text{start}}; x^s_{\text{end}}]$. The segments are sorted by ascending start boundaries. The slope a is common to all segments.

- TimeFactor is equal to the slope a:
  TimeFactor $= a$.
- ItemTime is equal to the abscissa of the intersection of the first line with the zero-ordinate axis:
  ItemTime $= -\frac{b_1}{a}$.
- StartTime equals the start boundary of the first segment:
  StartTime $= x^1_{\text{start}}$.
- EndTime equals the end boundary of the last segment:
  EndTime $= x^S_{\text{end}}$.
- Successive segments with $b_{s+1} < b_s$ correspond to an insertion in the *stream* signal:
  InsertTime (in the stream) $= x^s_{\text{end}}$,
  InsertDuration (in the stream) $= \frac{b_{s+1}-b_s}{a}$.
- Successive segments with $b_{s+1} > b_s$ correspond to an insertion in the *item* signal:
  InsertTime (in the item) $= a(x^s_{\text{end}} - \text{ItemTime})$,
  InsertDuration (in the item) $= b_{s+1} - b_s$.

Figure 9 illustrates the distinction on insertions. In order to cut the inserted chunks and synchronize both signals, the time factor is used differently when cutting an insertion in the stream (a) or in the item (b). The black line represents the two consecutive segments, and the gray segment represents the synchronized position for the second segment, after correction .



(a) stream cut (b) item cut

Figure 9: *Illustration of the correction of an insertion detected between two consecutive segments. The estimate of the insertion duration is indicated.*

5. EVALUATION

The full evaluation of such a process can only be done by human-checking all the occurrences that have been verified and aligned, as well as the occurrences discarded.

We have thus limited the evaluation to a subset of 100 occurrences randomly extracted from the Quaero corpus annotations of the 2010 campaign of evaluation for audio identification. Some of the training items of the corpus were delivered by Yacast in several edit versions. Among the occurrences, we have then deliberately introduced 30 errors of edit version, in order to verify that edit mismatches are correctly identified as annotation errors. Item mismatch is supposed much more easy to detect than edit mismatch, and is thus not tested here.

After their automatic verification and alignment, the 100 occurrences were human-checked with the help of a small tool we developed prior to this contribution, in order to perform this synchronization manually. The tool interface (developed in Matlab) in shown Figure 10. The user can adjust the characteristics presented earlier (ItemTime, TimeFactor, etc.) and play the item and the aligned stream occurrence simultaneously to check the alignment. The software is meant to be used with headphones, since the item is played on the left channel and the occurrence on the right channel. The full description of the software is not relevant here.

After several hours of annotation we have concluded that the perception of a slight phase shift of d seconds is very consistent:

- $d \approx 0$: When sounds are perfectly simultaneous, one sound is heard and located in the middle of the head.
- $|d| < 0.03$: In a scope of about 30 ms, we still hear one sound, but the latter moves on the side when $|d|$ grows.

Figure 10: *Graphical User Interface of the annotation tool developed for the manual alignment and verification of occurrences.*

- $0.03 < |d| < 0.07$: Between 30 ms and 70 ms, we start hearing two different sounds on sharp onsets (e.g. percussive sounds or some consonants on the singing voice).
- $|d| > 0.07$: Above 70 ms, we hear two different sounds.

These empirical observations corroborate the results in the literature on spacial hearing perception [17].

We have thus limited the precision of the parameters to 0.01 s in the annotation tool, which is still notably heard (through the "perceived" position of the sound in the head) when approaching perfect synchronization.

Using the tool, we have corrected the result of our process to reach the best synchronization possible. Figure 11 shows the distribution of the corrections applied to the ItemTime parameter. Most of the correction amplitudes do not exceed 40 ms. The mean amplitude of the corrections equals 25 ms, which is an expected order of magnitude since the step size between the fingerprint codes was set to 50 ms. Some corrections, though reach higher values, up to 90ms. This is explained by the fact that a slight error on the time-factor can induce a much larger difference on time offsets, especially at the beginning of the occurrence, where the correction was applied.

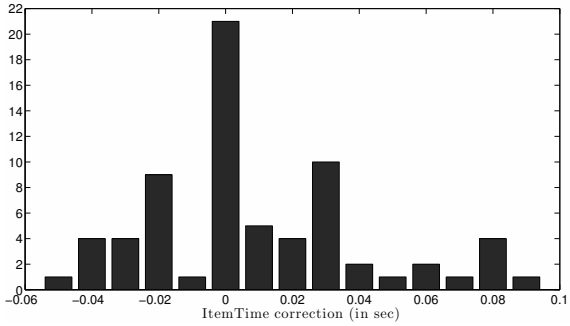


Figure 11: *Histogram distribution of the corrections manually set to the ItemTime estimated with our method.*

The 30 occurrences with edit mismatch were all detected as such, except one where the singing voice part is common to both edits. The detection of erroneous annotation is thus very efficient and reliable.

Finally, we have verified the TimeFactor estimation, by checking that the two sounds are still perfectly synchronized after 20 seconds of signal. Only 4 occurrences over the 70 correct ones (about 6%) were slightly offbeat, the rest remains aligned.

Nevertheless, the best demonstration is still to actually hear the sounds. Several audio samples of the item and stream occurrence (before and after the automatic correction), as well as the stereo mix of the two, can be found on the following webpage: `http://www.mathieuramona.com/wp/data/align`.

6. CONCLUSION AND PERSPECTIVE

We have proposed here an original variant of the fingerprinting scheme, called item-restricted fingerprinting, that is associated with a segment detection method to estimate the temporal distortions between an item occurrence signal and the original item signal. The high precision of the parameters estimation allows the compensation of the temporal distortions and the perfect synchronization of the item and the occurrence.

This method has been used to verify and correct approximative annotations for audio fingerprinting. The short evaluation shows that the incorrect annotation detection works almost perfectly, even on different edits of the same musical track. The estimation of the temporal characteristic proves very precise and on most on the items, the perfect synchronization of the item and stream signal is confirmed perceptually, after compensating the temporal distortions.

This contribution offers many applications. In the field of audio fingerprinting, the alignment of the occurrences allows to reproduce a part of the evaluation protocols generally applied to synthetic alterations of items. Moreover, the precise estimation of the time-factor enables controlled studies on robustness to time-scaling on real-world audio data. The problem of signal alignment answered in this paper can probably be extended to other fields of research in audio processing.

Short-term perspectives would concern the remaining flaws of the algorithm. The peak near the value 1 in the distribution of the point-slopes deserves a proper explanation and should be answered more reliably. The correction of the insertions is also problematic. The duration is correctly estimated through the offset values, but the position is not precise enough, and results in local asynchrony between the signal. Proposing a proper scheme for the detection and the correction of missing occurrences in a fingerprint evaluation corpus is another long-term perspective.

7. ACKNOWLEDGMENTS

This work was partly supported by the “Quaero” Program funded by Oseo French State agency for innovation.

8. REFERENCES

[1] Cyril Joder, Slim Essid, and Gaël Richard, “Hidden discrete tempo model: a tempo-aware timing model for audio-to-score alignment,” in *Proc. ICASSP ’11*, May 22-27 2011, pp. 397–400.

[2] Arshia Cont, “A coupled duration-focused architecture for realtime music to score alignment,” *IEEE Trans. on Pattern*

Analysis and Machine Intelligence, vol. 32, no. 6, pp. 974–987, June 2010.

[3] Meinard Müller, Peter Grosche, and Frans Wiering, "Automated analysis of performance variations in folk song recordings," in *Proc. ACM International Conference on Multimedia Information Retrieval (MIR '10)*, Philadelphia, Pennsylvania, USA, March 29-31 2010, pp. 247–256.

[4] Meinard Müller, Frank Kurth, and Michael Clausen, "Chroma-based statistical audio features for audio matching," in *Proc. IEEE Workshop on Applications of Signal Processing (WASPAA)*, New Paltz, New York, USA, October 2005, pp. 275–278.

[5] Michael Casey, Christophe Rhodes, and Malcolm Slaney, "Analysis of minimum distances in high-dimensional musical spaces," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 16, no. 5, pp. 1015–1028, July 2008.

[6] Mathieu Ramona, Sébastien Fenet, Raphaël Blouet, Hervé Bredin, Thomas Fillon, and Geoffroy Peeters, "Audio fingerprinting: a public evaluation framework based on a broadcast scenario," *submitted to Special Issue on Event Recognition*, 2011.

[7] Pedro Cano, Eloi Batlle, Harald Mayer, and Helmut Neuschmied, "Robust sound modeling for song detection in broadcast audio," in *Proc. AES Convention 112th*, 10-13 mai 2002, pp. 1–7.

[8] Avery Li-Chun Wang, "An industrial-strength audio search algorithm," in *Proc. ISMIR '03*, 2003.

[9] Eugene Weinstein and Pedro Moreno, "Music identification with weighted finite-state transducers," in *Proc. ICASSP '07*, April 15-20 2007, vol. 2, pp. 689–692.

[10] Jaap Haitsma and Ton Kalker, "A highly robust audio fingerprinting system," in *Proc. ISMIR '02*, 13-17 octobre 2002.

[11] Xavier Rodet, Laurent Worms, and Geoffroy Peeters, "Brevet FT R&D/03376: Procédé de caractérisation d'un signal sonore - Patent 20050163325 Method for characterizing a sound signal," brevet international, July 2003.

[12] Mathieu Ramona and Geoffroy Peeters, "Audio identification based on spectral modeling of bark-bands energy and synchronisation through onset detection," in *Proc. ICASSP*, May 22-27 2011, pp. 477–480.

[13] Jean-Julien Aucouturier and François Pachet, "A scale-free distribution of false positives for a large class of audio similarity measures," *Pattern Recognition*, vol. 41, no. 1, pp. 272–284, 2008.

[14] Richard O. Duda and Peter E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, pp. 11–15, January 1972.

[15] Emanuel Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[16] Z. I. Botev, "A novel nonparametric density estimator," Tech. Rep., The University of Queensland, 2006.

[17] Jens Blauert, *Spatial Hearing: The Psychophysics of Human Sounds Localization*, The MIT Press, revised edition edition, October 2 1996.