Invariant Audio Prints for Music Indexing and Alignment

Rémi Mignot, Geoffroy Peeters

Invariant Audio Prints for Music Indexing and Alignment

Rémi Mignot
STMS Lab - IRCAM,
Sorbonne Université, CNRS (UMR9912)
75004, Paris, France
ORCID: 0000-0002-9432-5871

Geoffroy Peeters
LTCI - Télécom Paris,
Institut Polytechnique de Paris
91120, Palaiseau, France
ORCID: 0000-0001-5255-3019

***Abstract*—This work deals with music indexing and alignment using audio codes designed to be representative of the music content and robust to sound modifications. First, based on properties of the Fourier Transform and of the logarithm, high-dimensional audio descriptors are designed. Then, a dimension reduction is learned with criteria based on sound discrimination and invariance to transformations. Finally, a binarization is computed to derive codes (integers). This last process allows a fast searching for large catalogs with a hash table, and a Hamming distance on codes makes possible the time alignment using an adapted "Dynamic Time Warping". The contributions of this paper are tested for two different tasks. The goal of the first task is to identify the segments of music medleys with the audio indexing process, and to accurately find the corresponding original time positions. The goal of the second task is to measure the accuracy of the time-alignment with synthesized MIDI files, where the tempo continuously varies, and with modified pitches and instruments. Additionally, the audio indexing is also tested for these data, in order to exhibit some properties of the used audio prints.**

***Index Terms*—Content Analysis and Indexing, Signal processing, Machine learning**

I. Introduction

For music applications, audio-to-audio alignment consists in synchronizing recordings being different performances of the same music work/score; but played by different musicians, and with possibly different instruments, pitches and tempi. As presented in [1], most standard methods in the literature search the time alignment by means of music similarities computed from time sequences of combined content-based short-term audio descriptors. They are for example: spectral centroid, spread, roloff, or flatness, MFCCs, pitch chroma, onset features (e.g. spectral flux). For a more efficient alignment, it is concluded in [1] that using a combination of different features achieves better results. Nevertheless, some of them are highly dependant on the pitch (chroma e.g.), or the timbre (MFCCs e.g.), and there use may negatively affect the results with music transposition or different instruments for example.

In audio indexing, the inputs signals are represented as time sequences of codes which are, ideally, invariant to such transformations. The most efficient and popular audio indexing methods are based on local spectrogram peaks, see e.g. [2]. The provided fingerprints are highly robust to noise addition and filtering. They can also be made robust to pitch shifting and time stretching with some adaptations, see [3], [4]. Nevertheless, these codes are not relevant neither to the acoustics and timbre properties nor to the music content, and then they are not suited for music similarity or distances.

We propose here to use the mid-term audio prints of [5], initially derived for audio indexing, and which are as robust as possible to transformations (e.g. noise addition, filtering, time stretching, pitch shifting) and relevant to the music content (e.g. rhythm, melodies). With an adaptation of the original indexing process of [5], including e.g. time alignment, we present a new method for audio-to-audio alignment. With these modifications, we also propose a segmentation task of medleys in an audio indexing context.

In Sec. II, the audio prints are presented together with their properties, then the hashing process and the search steps are presented in Sec. III, and the two tested tasks in Sec. IV. Finally Section V concludes this paper.

II. Designing and learning of Audio prints invariant to degradation

The audio indexing method proposed in [5] relies on extracting sequences of mid-term audio prints robust to degradation. First, high-dimensional audio descriptors are computed using frame analysis with a window of a few seconds. Using properties, of the Fourier transform e.g., this representation is by design invariant to some sound transformations. Then, an affine reduction is learned to reduce the dimensionality with criteria based on robustness, discrimination and decorrelation. Finally, a binarisation provides integer codes used to index a hash table. The following sections summarize these three steps. Note that more details are available in [5], about the choice of the parameter values for example.

A. Handcrafted High-Dimensional Audio Descriptors

1) Spectrogram: The *Short-Time Fourier Transform* $\mathbb{X}_{[k,\ell]}$ is computed over the whole signal, where k and ℓ denote the indices of the frequency bins (up to 5500 hertz) and the time frames. Then, sub-spectrograms of 3 seconds mid-term windows are extracted: $X^p_{[k,\ell]} = |\mathbb{X}_{[k,\ell_p+\ell]}|$ for all $\ell \in [\![0, L[\![$, with L the window duration expressed in number of frames. The ℓ_p's denote the starting time frames of the p-th mid-term window, their selection is described in Sec. III-B.

2) Log–log conversion: So far, the time axis ℓ and frequency axis k are expressed in linear scales (homogeneous to seconds and hertz). We convert those to logarithmic scales (log-time and log-frequencies). The new log–log spectrogram $H^p_{[\kappa,\lambda]}$ represents the frequencies from 150 to 5000 Hz, and the times from 0.5 to 2.5 sec, with $\ell = \ell_p$ as origin $t = 0$. The conversion is based on a fast interpolation schema with a good behavior both for over-sampling (low frequencies and times close to 0) and under-sampling (high values). κ and λ are the indices of the sampled axes with a logarithmic profile, between the previously given limits.

3) Sub-band division of the frequencies: We split the log-frequency axis into 5 bands, with an equal quality factor, and with 50% overlapping. We denote the corresponding sub-matrices by $h^{p,b}_{[\kappa,\lambda]} = H^p_{[\kappa+\kappa^b_0,\lambda]}$, with $\kappa + \kappa^b_0 \in [\![\kappa^b_0, \kappa^b_1]\!]$ the frequency range of the band $b \in [\![1, 5]\!]$. Each sub-matrix is of size (32×64). The following steps are identically processed on the 5 matrices of the bands.

4) Amplitude modifications: First, low amplitude values of $\boldsymbol{h}^{p,b}$ are rectified, see (1). Second, a 2D weighting is done to smoothly bring the borders to 0 (both for time and frequency axes). Third, a L_∞ normalization is applied, and fourth, a quasi-logarithmic conversion of the amplitudes is done. The former conversion maps the range $[0, 1]$ to itself, with a linear behavior for low values, and a logarithmic behavior for higher values. The four steps are summarized by:

$$\boldsymbol{g} \leftarrow \max(\sigma, \boldsymbol{h}), \quad \text{where } \sigma = r \max\{\boldsymbol{h} \odot \boldsymbol{w}\}, \tag{1}$$

$$\boldsymbol{g} \leftarrow \boldsymbol{g} \odot \boldsymbol{w}, \tag{2}$$

$$\boldsymbol{g} \leftarrow \boldsymbol{g} / \max\{\boldsymbol{g}\}, \tag{3}$$

$$\boldsymbol{f} \leftarrow \log(1 + a\,\boldsymbol{g}) / \log(1 + a), \tag{4}$$

with $w_{[\kappa,\lambda]}$ the 2D Hamming window, $r = 0.15$ and $a = 10$ two parameters, and $\odot$ the Hadamard product.

5) 2D Discrete Fourier Transform: Finally, a 2D-DFT is computed on each modified log–log spectrogram $\boldsymbol{f}^{p,b}$. Only its modulus is considered. The final representation is then:

$$\mathcal{Y}^{p,b} = \left| \text{2D-DFT}\left\{ \boldsymbol{f}^{p,b} \right\} \right|. \tag{5}$$

Because of the 2D hermitian symmetry, only one half is kept, leading to matrices with size (32×33): 32 for the log-frequency axis, and 33 for the log-time axis. We then obtain a vector of 1056 coefficients for each band b and time ℓ_p.

6) Invariance properties: The previous steps have been designed to get intrinsic invariance to some transformations.

- Time stretching ($t \leftarrow \alpha t$) and pitch shifting ($f \leftarrow \beta f$) are considered. Because of the log–log conversion of the scales (both time and frequencies), they lead to a 2D translation of the content of $H^{b,p}_{\kappa,\lambda}$. Then, because of the shift-invariance of the Fourier Transform magnitude, the final representation is robust to time and pitch compression or expansion. Note that, the DFT magnitude is actually invariant to *circular* shifts, so the 2D-weighting of (2) has the goal to lessen border effects.
- Noise addition is considered. First, with band-pass noises, the sub-band division permits to isolate the noise effect to some bands, and so to leave others unchanged. Second, the rectification of (1) cancels the noise effect if its level is below the floor σ, and leaving higher values unchanged.
- Filtering and gain modulation behave as the products of the spectrum with a time-invariant frequency response, or the time signal with a frequency-independent modulating signal, respectively. Because of the logarithmic behavior of (4), these two modifications provide offsets of $\boldsymbol{f}^{p,b}$ independent from the time or the frequencies respectively. After the 2D-DFT, these modifications only have effects on the first column and the first row of $\mathcal{Y}^{p,b}$. In this work, they are kept and we leave the following projection to choose if this information is useful.

7) Embedded music information: Unlike standard audio fingerprints (such as based on local spectrogram peaks [2], [4]), our method relies on audio descriptors which are more relevant to the music. For example, applying a 1D-DFT over the time (instead of the 2D-DFT) provides coefficients similar to the modulation spectrum (see [6], [7]), which are related to the rhythm. Applying a 1D-DFT over the frequencies provides coefficients similar to the cepstrum or MFCCs (see [8], [9]), which are related to the timbre. Here, the last step is a 2D-DFT (DFT both in frequency and time). The obtained coefficients are then meaningful to the music, related to the timbre, the rhythm, and the melody. In this paper, these properties are used for music similarity, indexing, segmentation and alignment tasks.

B. Learning an Invariant Projection

Because the audio descriptors are high-dimensional (1056), we need to reduce their dimensionality. To reduce it, we apply a chain of five linear projections which are learned using criteria related to - robustness to degradation, - discrimination on the original content, and - decorrelation of the output variables. The next paragraphs summarize each projection, more information are available in [5]. Note that the following process is identically repeated for each band.

1) Ill-Conditioned Component Rejection: Our goal is to avoid conditioning issues. For this, we make the variables linearly independent. We do this by computing a projection similar to a *Principal Component Analysis* (PCA), but without initial variable centering and keeping all components associated with non-null singular values. When computed on our training dataset, we obtained a reduction from 1056 coefficients to 1026 (30 directions of the original space are redundant; this is partly due to a residual effect of the hermitian symmetry).

2) Linear Discriminant Analysis: Our goal is to make the descriptions robust to degradations. For this, we perform an LDA [10] in which each class is defined as the set of vectors obtained from the same original signal segment (3 seconds, see Sec. II-A) but various degradations. The LDA both minimizes the within-class covariance, reducing the degradation effects of the new sub-space, and maximizes the between-class covariance, increasing the discrimination of the new representation

for different original signals. The output sub-space is chosen with a size of 80 components.

3) Independent Component Analysis: To improve the uniform filling of the hash table, see Sec. III-A, the chosen path is the use of independent variables. To achieve this, the Fast-ICA algorithm is used, see e.g. [11]. We then obtain a new basis of the same space, with size 80, and with independent variables. Nevertheless, based on informal tests, we remarked a loss of robustness of this representation. For this reason, the next reduction has the goal to recover robustness, and with an orthogonal projection which guarantees the preservation of the variable decorrelation.

4) Orthogonal Mahalanobis PCA: The *Mahalanobis PCA* has the same goal as the LDA (improving the robustness and discrimination), but using a different formulation (similar to metric learning). More details of the method are presented in [5]. To preserve the variable decorrelation, an orthogonal version has been derived, based on a recursive process rather than a Gram-Schmidt orthogonalization. The size of the new sub-space is 40, and the components are ordered with a decreasing order of discrimination power.

5) Hadamard's Transform: The last step is a Hadamard Transform with size 40, see [12]. First, because of its orthogonality, the variable decorrelation is still preserved, and second, because its matrix cells have the same magnitude, with different signs, it has the property to equalize the discrimination power of the output variables.

6) Combined projection: Because all these 5 transformations are linear or affine, they can be combined to get a global affine projection: $Z = \mathbf{P^b}\ X + \mathbf{t^b}$, with X the initial high-dimensional audio descriptors with dimension 1056, and Z the output vector with 40 components which are: centered, normalized, mutually uncorrelated, robust to degradation and discriminant to the original signal. Note that, these projections are separately learned for each band b, leading to 5 different reductions, with the same properties.

C. Binarisation

Audio print codes are finally obtained with a binarisation of the reduced audio keys Z components based on their sign. With $1 \leq k \leq 40$, the k-th bit of the code Γ is given by:

$$\gamma_k = h(z_k) = \quad 0 \text{ if } z_k < 0, \quad 1 \text{ otherwise.} \tag{6}$$

In this work, the codes Γ are used both to define indexes for a hash table, see Sec. III-A, and to compute music similarity with the Hamming's distance, see Sec. III-D. Note that one code Γ is obtained for each analysis time ℓ_p and each band b.

III. Indexing and time analysis

A. Approximate Hashing

The audio print codes Γ can be used to index a hash table for a fast search, as in [2], [13]. Nevertheless, if at least one bit of Γ is altered, the whole index is corrupted. To significantly improve the tolerance to bit corruption, an approximative hashing is used, inspired by the *Local Sensitive Hashing* (LSH), see e.g. [14], [15]. To this end, L LSH-functions are derived by selecting 16 bits of Γ. With $l \in [\![1, L]\!]$, we then obtain the lsh-code β_l, composed of a selection of 16 bits of Γ. These bit selections are chosen with a quasi-random drawing to maximise the decorrelation of the β_l's, and are fixed for the whole process. Whereas the alteration of one bit corrupts the whole code Γ, it only affects some lsh-codes, leaving the others unchanged.

Note that, the lsh-codes obtained from the L lsh-functions and the 5 bands cannot be mixed in a same hash table with dimension 2^{16}. Either we build $5L = 255$ different sub-hash tables, or we use a unique hash table by extending the codes to 24 bits, i.e. with 8 additional bits to inform about the band b and the lsh-function l.

B. Adaptive Anchoring Time for a Non-Regular Sampling

The mid-term audio prints presented in Sec. II are computed using frame-analysis with a window of a few seconds anchored in ℓ_p. For indexing, we use a mean time lag of $\bar{\delta} = 0.25\,\text{sec}$ between two adjacent windows. Nevertheless, as explained in [16], a uniform sampling of the time axis may provide time shifts between the query and the corresponding reference song. To solve this issue, the analysis times ℓ_p are synchronised to the maxima of an onset detection function.

For this, we use the frame-wise difference of the L_1 norm with a half-wave rectification $R(x) = \max(0, x)$:

$$\varphi_\ell = R\left(\|\mathbf{X}_\ell\|_\mathbf{1} - \|\mathbf{X}_{\ell-\mathbf{1}}\|_\mathbf{1}\right), \tag{7}$$

which is a good compromise between simplicity and robustness, see [5]. Finally, the analysis times ℓ_p are selected by picking up the positions of the local maxima of φ_ℓ with a sliding window of size $\bar{\delta} = 0.25\,\text{sec}$.

C. Search steps

Indexing is performed in two steps:

STEP1: We first compare the codes of the query and the reference songs of the catalog and select n candidates based on the number of matching code occurrences. This step is accelerated using a hash table, as in [2].

STEP2: We then analyze the time coherence of the matching codes between the query and each candidate. We details this in the next section.

D. Time Alignment

In [2], a linear relation (with a slope of 1) between the query time and the reference time (t_q and t_r) is assumed. For this, they compute the histogram of $t_q - t_r(t_q)$ which is supposed to remain constant. In [5], this process was refined to deal with constant time stretching, i.e. for a linear relation with a slope different than 1. To improve flexibility, and deal with different tasks, such as segmentation and audio-alignment, see Sec. IV, we propose here to adapt the *Dynamic Time Warping* method.

The DTW allows the alignment of two sequences based on local distances (which measure the similarity of each possible element pair). DTW determines the best path joining the start and the end of the sequences, defined as the one which

minimizes the sum of local distances, and with additional transition constraints. In audio, this is used to align time between two signals, see e.g. [17, Chapter 4].

In this work, the audio print sequences of the query and of the reference are used to compute the local distance matrix. The Hamming's distance, which is the number of different bits, is calculated for all binary code pairs. At each time step ℓ_p, we consider the code obtained by concatenating the 5 codes Γ^b of the bands, forming 200-bits integers. To better fit our applications, we propose four adaptations.

1) Transitions: The method proposed in Sec. III-B for selecting the analysis times ℓ_p results in different anchors depending on the time-stretching factor of the audio. Most ℓ_p's remain unchanged, but some disappear with speeding up and others appear with slowing down. In this case the DTW path must be able to "jump" anchor times. That's why we introduce new DTW transitions instead of the traditional horizontal and vertical transitions. The three used transitions are then:

- $(m, n) \rightarrow (m + 1, n + 1)$: *diagonal transition*,
- $(m, n) \rightarrow (m + 2, n + 1)$: *accelerated transition*,
- $(m, n) \rightarrow (m + 1, n + 2)$: *decelerated transition*,

with m and n the time indices of the reference and the query.

2) Free path boundaries: For a standard recognition of audio excerpts, the start and the end of the two sequences may not correspond, which transgresses the standard DTW constraints. To overcome this issue the constraint of the path boundaries is released. So, the path can start and end at any time, with the only constraint to coincide with boundaries of the query or the reference, but not necessarily both. As a summary, the path must start and end on the "borders" of the distance matrix, and not necessarily in the "corners".

3) Mean distance: With the two previous adaptations, the shortest paths may be favored. To avoid this, the adapted DTW algorithm tries to select the path which minimises the mean distance rather than the sum of local distances. Note that it may lead to a sub-optimal solution, but is not a critical issue.

4) Even freer boundaries: For the segmentation task of Sec. IV-A, the boundaries of the matching segment may not correspond with the boundaries of neither the query nor the reference, providing a problem similar to the local alignment as in [18]. In this work, first the path is searched with boundaries on the borders of the distance matrix, second the limits of the best segment is searched with a 1D process on the found path.

E. Candidate selection

To speed up the process, rather than selecting a fixed number n of candidates in STEP1 (passed to STEP2), we use an outlier detection. With the candidate songs sorted in decreasing order of matching occurrence count, the idea is to iteratively compute the time alignment with the candidates, and to stop as soon as an abnormally low DTW distance is returned.

In the following experiments, we use the standard *Inter-Quartile Range* method for the outlier detection, but only considering low values as outliers. We use a minimal number of candidates of 10, and a maximal number of 400.

IV. Experiments

In this section, we present two experiments which have been designed to demonstrate on one hand the benefit of the derived audio prints, and their invariance properties, and on the other hand the flexibility of the adapted DTW: the segmentation of medleys and audio-to-audio alignment.

A. Audio Indexing and Segmentation of Medleys

1) Task description: The goal is to recognize the different audio fragments that compose a music medley. We measure the indexing performance, as well as the accuracy of the time mapping.

2) Details of the method: To facilitate this task, the medleys are divided into 30 sec chunks with a 15 sec hop size. For each chunk, the indexing is processed to find segment candidates in the catalog. Because the boundaries of the segment and the chunk may not correspond, the boundary constraint is fully removed, see Sec. III-D4.

Then a post-processing filters out bad segments, with low distances, and tries to merge segments of different chunks from the same reference song. Finally, a linear regression is applied to refine the time mapping, (linear because the time stretching factor is supposed to be constant for each segment).

The time stretching provides a systematic bias of the estimated time. This is explained by the 2 sec windowing and the 0.5 sec time shift of Sec. II-A2. We empirically observed that for a stretching factor 2, the bias is 0.5 sec, then it is removed based on the estimated time stretching factor.

3) Evaluation procedure: We built a large dataset (catalog) made of approximately 40 000 reference songs from [19], and we added the 195 songs of [20]. The full hash table is built. The medleys are automatically built by a random selection among the 195 original songs. The segment durations are randomly drawn between 15 and 30 sec, and the segments are concatenated to form medleys of approximately 2 minutes.

To test the robustness, we consider different levels of degradations as described below. *Time stretching*: The stretch factors of the segments are randomly drawn in the ranges: $\sigma \in [-33, 33]$ cents, or $[-66, 66]$ cents. The *cent* unit, noted (ct), is defined by: $f = 2^{\sigma/100}$, where f is the stretch factor. For example, 100 cents corresponds to a slow down of the tempo by a factor 2. *Pitch shifting*: The pitch shifting factors are: ± 1, ± 2, and ± 4 half-tones, noted (ht). Only the sign is randomly drawn. *Sound degradations*: First, a graphical EQ is used with alternated gains ± 4.5 dB for each octave band. Then, a pink noise is added with 10dB SNR, a 40 kbps MP3 encoding is simulated, and a distortion is simulated with a pre-gain of 3 dB and a sine function[1].

4) Results: The results are given in Tab. I. Each performance measures are computed with 730 medleys. The *Good detection* measure is the time-wise ratio of sum of times where the good song is recognized to the total duration. The *F-measure* presents the medley-wise performance of song

[1]Some sound examples can be heard at the following address: https://anasynth.papers.ircam.fr/2024/CBMI/

TABLE I
MEDLEYS: RESULTS OF SEGMENTATION.

	Time Stretch (cents)	0				[−33, 33]				[−66, 66]			
	Pitch Shift ($\frac{1}{2}$ tones)	0	±1	±2	±4	0	±1	±2	±4	0	±1	±2	±4
No degradation	**Good detection** (%):	96.5	97.3	97.2	94.3	97.2	97.1	96.3	87.5	95.7	94.9	91.6	76.6
	F-measure mean (%):	100.0	100.0	99.9	96.8	99.9	99.8	99.2	90.6	99.5	98.9	96.2	80.6
	Time error: median *(mean)* (ms):	1 *(59)*	2 *(31)*	2 *(95)*	4 *(94)*	36 *(113)*	35 *(60)*	34 *(70)*	38 *(179)*	47 *(108)*	47 *(102)*	51 *(184)*	66 *(334)*
With degradation	**Good detection** (%):	96.6	96.4	93.6	77.2	95.9	93.7	86.7	63.4	92.4	88.9	76.6	50.8
	F-measure mean (%):	99.8	99.3	96.4	79.9	99.3	97.2	90.1	66.8	96.8	93.7	81.3	54.5
	Time error: median *(mean)* (ms):	4 *(104)*	5 *(86)*	6 *(66)*	12 *(275)*	29 *(120)*	30 *(93)*	31 *(261)*	42 *(328)*	47 *(255)*	47 *(190)*	53 *(373)*	75 *(734)*

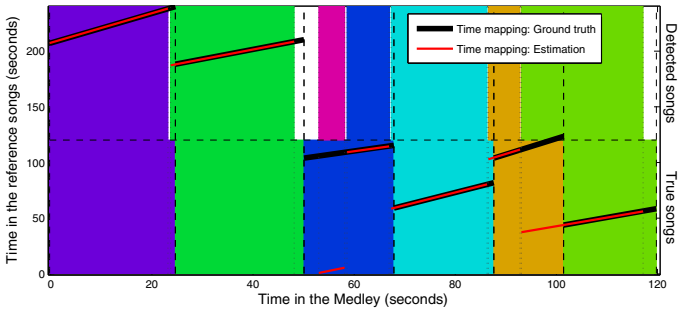

Fig. 1. Medley: Illustration of a medley segmentation. The colored rectangles represent: the true songs on the bottom half, and the detected songs on the top half, and the white spaces of the upper part are for time ranges when no song is detected. The black lines (resp. red) plot the time mapping between the medley time (x-axis) and the time of the true (resp. detected) reference songs (y-axis). The input medley has been created with: time stretching range $[-66, 66]$ (ct), pitch shifting ± 2 (ht), and with sound degradations, providing a difficult configuration. The slope of the black lines inform about the time stretching of the segments. Even if the original songs are well recognized most the time, and the time following is accurate, we can observe small errors at some transitions, providing short bad detections, and a more important inaccuracy at the transition between the 5th and 6th segments. Additionnally, a "unknown" song (coming from the big catalog) is detected during the first half of the third segment, providing also a bad detection, and reducing the precision value (and so the F-measure).

recognition. Then, computed for each time where the reference song is correctly detected, the *Time error* presents the median (and the mean) of the absolute difference between the true and the estimated mapped time.

As expected, the performances decrease when the transformation level increases. For example, without transformation, the *F-measure* is 100%, meaning that all the original songs are detected without False Positive. The time-wise good detection rate is 96.5%: the 3.5% of bad detections are due inaccuracy in the detected transitions, see e.g. Fig. 1 which illustrates the result on one medley.

Whereas the median of the time error does not significantly change with additional sound degradations, the mean and the detection rates are worse. So, we can conclude that the presence of degradations makes the estimations more "instable", but the audio prints are robust enough to degradations to make the time alignment working well. This results is tested in the next experiment for an audio alignment task.

B. Audio-to-Audio Alignment

1) Task description: We test here the time alignment based on the audio prints and the adapted DTW, in the context where the music to recognise is modified with continuously time-varying tempo. The task is then to map the time of the query to the time of the reference, or inversely. An indexing search of the modified songs, among a catalog, is also tested.

2) Details of the method: For indexing, the same procedure is used, but without linear regression.

For time alignment, the mean time lag of anchor times is set to 0.1 sec (instead of 0.25 sec) to get a finer time resolution. To remove the bias, a first alignment is done, and the time stretching is estimated by $\widetilde{\sigma}(t) = \mathrm{d}\widetilde{\tau}(t)/\mathrm{d}t$, with t the time of the reference song and $\widetilde{\tau}(t)$ the estimated time mapping to the modified song. The bias is then empirically estimated as in Sec. IV-A with: $b(t) = 0.5 \log_2 \widetilde{\sigma}(t)$. Note that the mapping $\widetilde{\tau}(t)$ is pre-processed using a symmetrical low-pass filtering (with null group delay) to smooth the evolution of $\widetilde{\sigma}(t)$.

For both sub-task, the DTW path is forced to "touch" the borders, with a duration not lower than 95% of the total duration of the query and the reference.

3) Evaluation procedure: From 238 MIDI files of [21], we modified the time stamps of the MIDI messages to finely control the time-varying time stretching. Additionally, we can easily change the note pitch and the synthesized instruments to test the invariance of the audio prints to different timbres.

Different levels of transformations are tested. *Time stretching*: The time stretching continuously changes on short segments with target values drawn in the ranges $[-33, 33]$, $[-66, 66]$ or $[-100, 100]$ (ct). The duration of the segments are also drawn between 1 or 6 sec. The sign of the time stretching alternatively change and the extreme values are favored to provide a strong effect. *Pitch shifting*: For the tests that use pitch shifting, the used transformations are: ± 2, ± 6, and ± 13 (ht). Only the sign of the transformation is randomly drawn. *Instrument changes*: To test the invariance of the audio prints to the timbre, we changed the original instruments by randomly drawing new ones among the proposed instrument list of the *General MIDI* norm, excluding sound effects. Additionally, the drum track is removed.

The MIDI files are modified using the *Pretty_MIDI* python module [21], and the sound is synthesized offline using the

TABLE II
MIDI COVERS: RESULTS OF INDEXING.

	Time Stretch (cents)	[−33, 33]				[−66, 66]				[−100, 100]			
	Pitch Shift ($\frac{1}{2}$ tones)	0	±2	±6	±13	0	±2	±6	±13	0	±2	±6	±13
Same instruments	**STEP1:** rank (full catalog):	1.0	1.2	4.3	17.9	1.0	1.3	5.7	22.7	1.0	1.4	5.9	29.
	STEP2: rank (over the 200 best):	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.1
Changed instruments	**STEP1:** rank (full catalog):	132.3	174.5	310.1	319.9	128.7	185.6	243.0	301.5	129.5	199.5	274.6	302.
	STEP2: rank (over the 200 best):	2.2	5.0	15.3	17.9	2.6	6.3	16.8	20.5	4.3	10.2	24.9	30.3

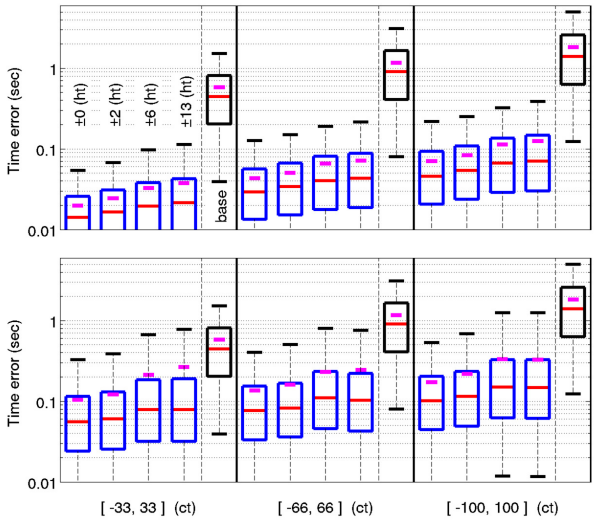

Fig. 2. MIDI Covers: Time accuracy of the alignment. Top figure: instruments unchanged, bottom figure: instruments changed and drums removed. Each figure is splitted into 3 blocks for the 3 time stretching ranges, and each block presents the 4 tested pitch shiftings and the baseline. The box plots display the median, the first and third quartiles, and the 5% and the 95% percentiles. Additionnally, the mean is diplayed with the rose small bar.

FluidSynth software with the Soundfont *FluidR3*, see [22]. For the indexing task the original synthesized MIDI files are added to the same catalog of 40 000 songs, see Sec. IV-A.

4) Results: Table II presents the indexing results. For *STEP1*, and with the same instruments, the mean rank of the true song is below 30 (over 40 000 songs) whatever the time stretching range and the pitch shifting. With instruments changed and drums removed, the mean rank increases but stay below 320 (/40 000). For *STEP2*, the rank over 200 tested songs, which are the best of *STEP1*, is almost perfect with only pitch shifting and time stretching. With modified instruments, the mean rank significantly increases, which demonstrates a certain sensitivity of the audio prints to the timbre. Hence, the tested method is probably not suited to search music covers with big catalogs. Note that the results are quite stable along the 3 time stretching configurations.

For time-alignment, the accuracy of the estimated time mapping is displayed in Fig. 2. Each result is compared with a baseline simply obtained by the time error of the true mapping and a straight line between the start and the end. We can observe a significant benefit of the method, even with instruments changed and drum removed. This demonstrates the ability of the audio prints to efficiently embed music information related to the rhythm and the melody, independently from the pitch, the tempo and the timbres. This is indeed useful when aligning audio with different instruments, and without prior knowledge about the used pitch shifting. Nevertheless, we remark lower performances with instrument change, which still shows sensitivity to timbre. Note that the results are quite stable along the 4 pitch shifting configurations.

Finally, let's remark that despite the coarse time resolution of the sampled times, almost 100 ms, the achieved accuracy is better in some cases thanks the special selection of analysis times, see Sec. III-B, based on an onset detection function.

V. CONCLUSION

For music segmentation and alignment, a method initially designed for audio indexing has been adapted. It takes benefit of audio prints relevant to the music and robust to some transformations (e.g. time stretching, pitch shifting, filtering). Representative of rhythm and melodies, they allow the segmentation and the alignment without any prior knowledge about the possible pitch modification, and instrument change.

To measure this, we made two experiments, one based on medley segmentation and the other based on synthesized MIDI covers. Even if the presented results do not seem sufficient for an indexing task with very big catalogs and strong degradations, these preliminary results are very encouraging for the targeted tasks: segmentation and alignment, even with strong transformations and instrument changed.

In a future work, we should focus more on audio-to-audio alignment, with a more in-depth comparison with the literature. Moreover, this work does not take benefit of recent progresses in machine learning, such as *Deep Learning*, and there are many ways for improvement, e.g. using neural metric learning as in [23]. Finally, for audio/music mixes, even if we obtained some interesting results (not presented), the currently method has not been designed for that, and it will be beneficial to specifically deal with this task.

ACKNOWLEDGMENT

This work has been initiated during the BeeMusic Project (FSN-O14703-370483), and new progresses have been funded by the AQUA-Rius project (ANR-22-CE23-0022-01).

REFERENCES

[1] H. Kirchhoff and A. Lerch, "Evaluation of features for audio-to-audio alignment," *Journal of New Music Research*, vol. 40, no. 1, pp. 27–41, 2011.

[2] A. Wang, "An industrial strength audio search algorithm." in *Int. Symposium on Music Information Retrieval (ISMIR'03)*, October 2003, pp. 7–13.

[3] S. Fenet, G. Richard, and Y. Grenier, "A scalable audio fingerprint method with robustness to pitch-shifting." in *Int. Symposium on Music Information Retrieval (ISMIR'11)*, October 2011, pp. 121–126.

[4] R. Sonnleitner and G. Widmer, "Quad-based audio fingerprinting robust to time and frequency scaling," in *Proc. Int. Conf. on Digital Audio Effects (DAFx'14)*, September 2014.

[5] R. Mignot and G. Peeters, "Degradation-invariant music indexing," STMS Lab - IRCAM, Sorbonne Université, CNRS, Tech. Rep., 2024, DOI: https://doi.org/10.48550/arXiv.2403.00688.

[6] L. Worms, "Reconnaissance déxtraits sonores dans une large base de données," Master's thesis, 1998.

[7] L. Atlas and S. Shamma, "Joint acoustic and modulation frequency," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, pp. 1–8, 2003.

[8] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoustics, Speech and Signal Process.*, vol. 28, no. 4, pp. 357–366, 1980.

[9] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Prentice hall, 1993.

[10] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley, 2012.

[11] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[12] Kunz, "On the equivalence between one-dimensional discrete Walsh-Hadamard and multidimensional discrete Fourier transforms," *IEEE Transactions on Computers*, vol. 100, no. 3, pp. 267–268, 1979.

[13] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system." in *Int. Symposium on Music Information Retrieval (ISMIR'02)*, vol. 2002, Octobre 2002, pp. 107–115.

[14] K. Moravec and I. Cox, "A comparison of extended fingerprint hashing and locality sensitive hashing for binary audio fingerprints," in *Proc. of the 1st ACM Int. Conf. on Multimedia Retrieval (ICMR'11)*, April 2011, p. 31.

[15] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the 30th annual ACM symposium on Theory of computing*. ACM, May 1998, pp. 604–613.

[16] M. Ramona and G. Peeters, "Audio identification based on spectral modeling of bark-bands energy and synchronization through onset detection," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'11)*, May 2011, pp. 477–480.

[17] M. Müller, *Information retrieval for music and motion*. Springer, 2007, vol. 2.

[18] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[19] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[20] D. Ellis, A. Berenzweig, and B. Whitman, "The "uspop2002" pop music data set," 2002, http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html.

[21] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016.

[22] "Fluidsynth," https://www.fluidsynth.org/.

[23] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Int. Symposium on Music Information Retrieval (ISMIR'19)*, November 2019.