SAMPLED MUSIC DETECTION BASED ON DYNAMIC TIME WRAPPING

Tatiana Barrios
KTH Royal Institute of Technology
trbm2@kth.se

Xuehua Fu
KTH Royal Institute of Technology
xuehua@kth.se

1. INTRODUCTION

This project on music informatics is about detecting samples of songs used in other songs based on Dynamic Time Wrapping. This is important because nowadays copyright issues are an ineludible issue with all of the music content being generated, and also because it makes an artist influence more traceable over different places and periods of time. We define a music sample as a portion from a known song used in another song created later. This portion can be as short as a second and as long as the whole song. Our results point that DTW has the capacity of recognizing those samples to certain extent, however it is not very robust when there is noise or tempo changes.

2. RELATED WORK

Detecting similarities in music has been a topic of music informatics since the early 2000s. The most known techniques to detect similarities in music so far are fingerprinting, chromagram comparison, and dynamic time wrapping. The former is the top technology being used by **Shazam** [1]. It creates a hashed time-frequency constellation from a song fragment, which is unique to that song. The fingerprinting algorithm starts by taking the sonogram of the song fragment and then finding out points of interest, which can be the points with the highest energy on a time-frequency window. These points are called anchors and are used to calculate the hashes of the song, each hash is paired with a timestamp. Finally, these hashes are used for music matching. which is the most important use case for **Shazam**.

Another application used in music matching is **SoundHound**. Its difference from **Shazam** is that it also can identify songs from humming and singing. This is explained by the fact that **SoundHound** was previously named Midomi, which is the service that originally got this concept done [2]. **SoundHound** is very likely to use fingerprinting aided with AI for the music queries, especially to identify a song by humming, since the sound is going to be different than the original song by key and tempo. They have a sound database, used to store the hums, since if a user wants to query a song by humming, the hum is going to get compared with others in the same database, rather than with the original song [3].

Although fingerprinting is a robust algorithm, capable to perform with noise and distortion, it only works when two compared fragments of music are basically identical. This is not the case for covers or sampled music. For cover detection, the main used technique is the comparison of chroma-based features [4], which can be done in multiple ways. Chroma features capture harmonic characteristics from an audio excerpt, and these harmonic characteristics are less specific than the fingerprints, hence its use for cover song detection.

One application that ranges from cover song identification to sample identification, which is the goal of this project, is **WhoSampled.com** [5]. However, different from **Shazam** or **SoundHound**, **WhoSampled** uses a team of people to identify covers, samples, or remixes and add those to a database. With that in mind, it is fair to say that automatic sample identification in music informatics is pretty much in its infancy, which makes sense since sampled fragments can be manipulated to become very different on their final songs. In [6], the authors use techniques from fingerprints to non-negative matrix factorization to detect the presence of a sample in a set of songs and its given time, with several grades of success between them.

3. METHOD

The method we implemented comes from [7], which uses chroma features and Dynamic Time Wrapping (DTW) to get similarity on two sound fragments. we take two songs: a song **A** which we call the *sampled song* and a song **B** which we call the *new song*. It is called *new* because it is always launched after the original song.

A sample is a fragment from the sampled song that is taken to be part of the new song. Our approach 1) finds where the sample appears in the new song and 2) takes a short fragment with a fixed length starting from the location found in step 1) and computes the optimal alignment between this fragment and another fragment of the sampled song. Two Python packages *libfmp* and *librosa* are used in the implementation.

3.1 Feature Design

Our approach takes chroma representations as the input features. Our goal is therefore to find similarities between chroma feature sequences from the two songs. However, these chroma features are represented as their normalized statistics instead of the direct chromagram. This is because chromagr ams could be too detailed to detect similarities in two different songs [7] in our case. Indeed, song **B** will use a fragment from song **A**, but this fragment might have pitch or tempo changes. For this, to increase the similarities between chromagrams they should go through a post-processing stage, where they are normalized, quantized,

.

smoothed and down-sampled. After all of these operations, we get a local average overtime for every dimension in the chromagram.

Therefore, we chose $\mathbf{CENS}_d^l$ (chroma energy normalized statistics) features where l is the length of the smoothing window (a Hann window) and d is the downsampling factor. The **CENS** features will still capture essential music information, but it will be better than the original chroma features to show similarities between sound fragments. In Figure 1, you can compare the plot of the chromagram against the plot of the **CENS**. The first one is denser than the latter one but they still resemble each other, even though there is down-sizing on the time scale.

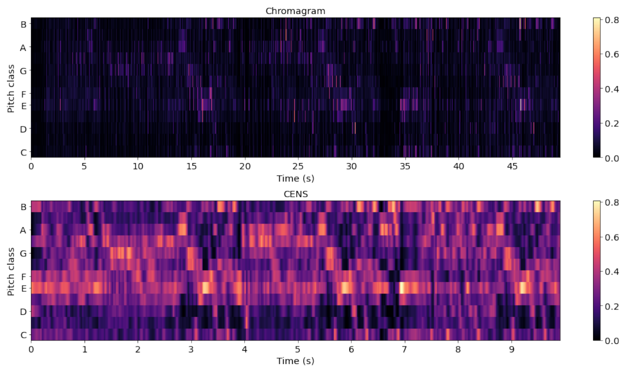

Figure 1. Original chromagram against CENS plot of a song

3.2 Pre-processing

For the pre-processing, we implement DTW over chroma features, to get where in time the songs are the most similar. To calculate these similarities, after getting the **CENS** features, we can calculate the a cost matrix $\boldsymbol{C}$ by using cosine distances c between the feature sequences.

$$\boldsymbol{C}(n, m) := c(x_n, y_m) \tag{1}$$

Given that the song **A** has a **CENS** of dimensions $(12, D_1)$ and song **B** has a **CENS** of dimensions $(12, D_2)$, the cost matrix will have dimensions (D_1, D_2). From this matrix, we get the minimal value and its location (Fig. 2), which signals the point with the least distance over the **CENS** matrix. To find the point in time where the minimal distance happens, this value gets upsized and upscaled, as

$$t_n = C_{min} \cdot d \cdot \frac{Hopsize}{F_s}, \tag{2}$$

where d is the downsampling factor, F_s is the original sample rate, and C_{min} is the column index of the minimum value in the matrix. This gives us the times where both songs are the most similar, i.e. where the sampled fragment is the most likely to appear. We use this location to determine the fragment in song **B** which we want to compare to song **A**.

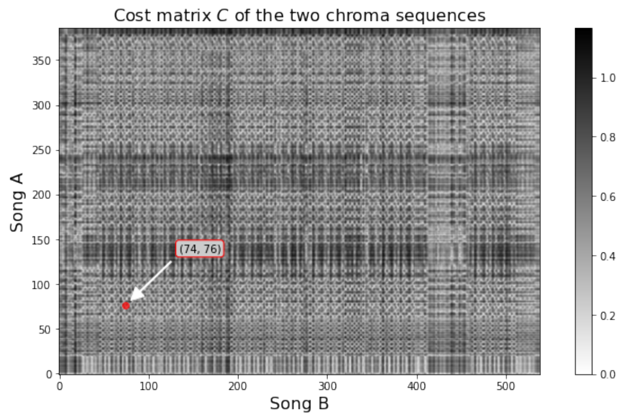

Figure 2. The global minimum point in cost matrix C, using *Informer (Snow)* and *Con Calma (Daddy Yankee ft Snow)* as an example.

3.3 DTW-based Matching

DTW is known to be able to measure the distance between sequences with different lengths, which could be useful for matching two similar fragments of songs with different length. Given two CENS feature sequences, our approach uses a variation of the original DTW algorithm to match the shorter sequence with a subsequence of the longer one.

Let $X = (x_1, x_2, \ldots, x_N)$ and $Y = (y_1, y_2, \ldots, y_M)$ be two feature sequences, where X is assumed to be much shorter than Y. In our case, sequence X is the **CENS** feature sequence obtained from a ten-second fragment of song **B** found in §3.2, while Y is the **CENS** feature sequence taken from the new song. A subsequence of Y is defined as

$$Y(a : b) = (y_a, y_{a+1}, \ldots, y_b),$$

where $a, b \in [1, M], a \leq b$. And the optimal sequence is denoted as $Y(a^\star : b^\star)$, which is the subsequence with the minimal DTW distance (§3.3.1) to sequence X.

3.3.1 Accumulated Cost Matrix

An warping path P between two sequences X and Y refers to a sequence $P = (p_1, p_2, \ldots, p_L)$ with $p_l = (n_l, m_l), l \in [1, L]$ being a pair of indices of components in X and Y. Hence, an alignment between X and Y is indicated by P. An optimal warping path is defined to has minimal accumulated cost along the path among all possible warping paths. The DTW distance denoted as $\mathrm{DTW}(X, Y)$ stands for the total cost of the optimal warping path between sequences X and Y.

The accumulated cost matrix $\boldsymbol{D} \in \mathbb{R}^{N \times M}$ derived from the cost matrix $\boldsymbol{C}$ (Eq. 1) measures the total cost of each warping path, and is computed by Eq. 3. Each value $\boldsymbol{D}(n, m)$ specifies the total cost of an optimal warping path starting at (x_1, y_1) and ending at (x_n, y_m), and hence the distance between any pair of subsequences $X(1 : n)$ and $Y(1 : m)$, i.e. the DTW distance $\mathrm{DTW}(X(1 : n), Y(1 : m)) = \boldsymbol{D}(n, m)$.

$$\begin{aligned}
\boldsymbol{D}(n,1) &= \sum_{k=1}^{m} \boldsymbol{C}(k,1) \qquad \text{for } n \in [1,N],\\
\boldsymbol{D}(1,m) &= \sum_{k=1}^{m} \boldsymbol{C}(1,k) \qquad \text{for } m \in [1,M],\\
\boldsymbol{D}(n.m) &= \boldsymbol{C}(n,m) + \min \begin{cases} \boldsymbol{D}(n-1,m-1)\\ \boldsymbol{D}(n-1,m)\\ \boldsymbol{D}(n,m-1) \end{cases}
\end{aligned} \tag{3}$$

for $n \in [2,N]$ and $m \in [2,M]$.

3.3.2 Optimal Subsequence

The optimal subsequence $Y(a^\star : b^\star)$ of Y is obtained from

$$(a^\star, b^\star) := \underset{(a,b):1 \le a \le b \le M}{\operatorname{argmin}} \operatorname{DTW}(X, Y(a:b)),$$

with $b^\star = \operatorname{argmin}_b \boldsymbol{D}(N,b)$ such that $Y(1:b^\star)$ gives the minimum distance to X. $a^\star$ is determined in a backtracking procedure to construct an optimal warping path [7].

The optimal warping path between CENS feature sequence X and the optimal subsequence is shown in Fig. 3.

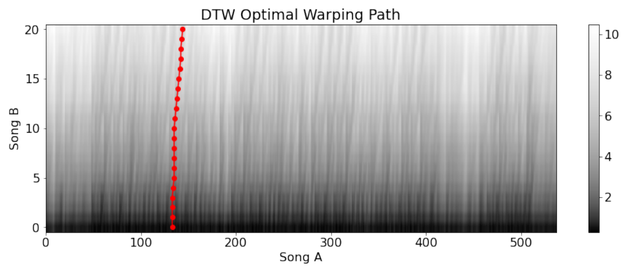

Figure 3. Warping path between X and $Y(a^\star : b^\star)$, using *Informer (Snow)* and *Con Calma (Daddy Yankee ft Snow)* as an example.

A matching function $\Delta_{\text{DTW}}(m)$ shows how well subsequences $Y(1:m), m \in [1,M]$ fit sequence X by measuring the average distances along the paths (Eq. 4).

$$\Delta_{\text{DTW}}(m) := \frac{1}{N}\boldsymbol{D}(N,m) \tag{4}$$

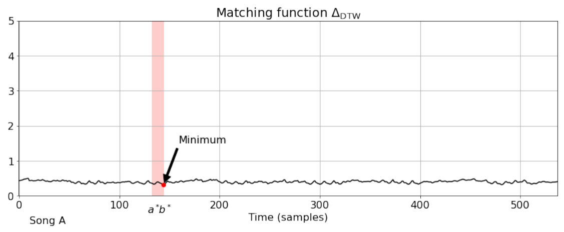

Figure 4. Matching function Δ_{DTW} and the optimal subsequence in Y, using *Informer (Snow)* and *Con Calma (Daddy Yankee ft Snow)* as an example.

Finally, the indices $a^\star$ and $b^\star$ of the subsequence in Y can be converted to time using Eq. 2.

4. EVALUATION

To evaluate the algorithm, we tested on a set of six pairs of songs, each contains a sampled song and a new song. The set of randomly selected songs are in Table 1. We ran the algorithm three times on each pair, once in the unmodified version, once in a noisy version, and the last on a slowed-down version. In the unmodified version, we just ran the algorithms on the songs as they were. In the noisy version, we added white noise to the song **B**, and in the slowed version, we slowed the tempo of also song **B**. The objective of this setup was to show if the algorithm was strong enough to detect the similarities even with the added difficulty.

Once we got the ”similar” fragments to the songs, we listened to them, in order to check if they were similar to human ears, from which we could calculate the number of failures against the total number of attempts.

No.	Song A (original)	Song B
1	Informer (Snow)	Con Calma (Daddy Yankee ft Snow)
2	Misirlou (Dick Dale)	Pump it (Black Eyed Peas)
3	Under Pressure (Queen)	Ice Ice Baby (Vanilla Ice)
4	Gimme Gimme Gimme (ABBA)	Hung up (Madonna)
5	Zamina mina (Zangalewa)	Waka waka (Shakira)
6	Edge of seventeen (Stevie Nicks)	Bootylicious (Destiny’s child)

Table 1. Songs used for evaluation.

5. RESULTS

Our algorithm was experimented on six pairs of songs over three different cases as stated in §4: 1) with unmodified versions of songs, 2) with a noisy version of song **B**, in which a Gaussian white noise multiplied by 0.1 was added, and 3) with a slowed-down version of song **B** by extending the length of the song up to 3.5 times. The overall results of our algorithm are shown in Table 2.

No.	Unmodified	Noisy	Slowed-down
1	✓	*only rhythm*	✗
2	✓	✓	✗
3	✗	✗	✗
4	✗	✗	✗
5	✓	✓	✗
6	*only rhythm*	*only rhythm*	✓
Overall	3/6	2/6	1/6

Table 2. Output fragments of songs and results evaluated by human ears over three different test cases. The ”overall” counts the number of successful attempts over the number of total attempts. ”Only rhythm” means that the fragments found by the algorithm share the same rhythm but may have different melodies. ✓is marked if the sampled fragment was found correctly in both songs, and ✗is marked if the resulted fragments sounded completely irrelevant.

One significant observation is that in the case using *Zamina mina (Zangalewa)* and *Waka waka (Shakira)*, the algorithm worked suprisingly well with the slowed-down version while it failed to some extent with the unmodified version. In fact, *Waka waka (Shakira)* uses a faster tempo than *Zamina mina (Zangalewa)*, and the slowed-down version accidentally fits the tempo of the latter. This is also consistent with the cases using song pairs **No. 3** and **No. 4**, with *Ice Ice Baby (Vanilla Ice)* being faster than *Under Pressure (Queen)* and *Hung up (Madonna)* also being faster then *Gimme Gimme Gimme (ABBA).*

Our implementation is attached as a link to the code: `https://github.com/cheriestoner/musinf_project`.

6. DISCUSSION AND CONCLUSION

As we can see, our implementation of DTW over chroma features for sampling detection leaves a lot to be desired, but it worked surprisingly well in some cases. The results indicates that our algorithm is not capable enough to deal with sped-up or slowed-down samples in the new song. Our hypothesis is that the tempo of the songs **A** and **B** must be pretty close for the algorithm to work. For instance, *Misirlou* and *Pump* have similar tempo compared to *Under Pressure* and *Ice Ice Baby*. A possible solution is to use melodic features in addition to chroma, the algorithm would have thrown better results for songs with a different tempo. [8] proposed a method that uses beat tracking to pre-process the data, extracts one chroma feature sequence per beat and then apply DTW on each pair of feature sequences, which would also be useful to improve our approach.

On the other hand, this project was initially about proving that music fingerprinting worked to detect similarities in our use case. However, we worked mainly with polyphonic music, and in this kind of music, the sampled fragments are mixed with other melodies. This makes the audio identification hashing change, thus fingerprinting does not work.

Another approach that would have worked for our project is taking into account that samples may appear more than once in the song. We only considered global optimal point where the sample may appears, so considering local optimal points instead would have yielded more interesting results. Another thing we wished we could have done differently is configuring the algorithm to find out how long the sample lasts in the new song and also dealing with tempo and pitch changes, as we mentioned before. We took the sample duration as 10 seconds or 20 seconds, so when there are speed variations, this does not work anymore.

Furthermore, since we only tested the algorithm in less than 10 songs, we cannot say that our results are really conclusive. We would need a bigger song dataset for that. Also, the songs we used are from different genres, so an improvement would be delimiting the genre for the new songs since the genre of the original song is not controllable.

7. REFERENCES

[1] A. Wang, "An industrial strength audio search algorithm." 01 2003.

[2] "Soundhound inc," https://www.soundhound.com/, accessed: 2021-11-04.

[3] J. Rehmeyer, "Tracking down a tune," https://www.technologyreview.com/2007/04/10/226000/tracking-down-a-tune/, 2007, accessed: 2021-11-04.

[4] L. Maršík, M. Rusek, and K. Slaninová, "Evaluation of chord and chroma features and dynamic time warping scores on cover song identification task," 2007.

[5] "Whosampled: Music dna metadata and technology," https://www.whosampled.com/metadata/, accessed: 2021-11-04.

[6] S. Gururani and A. Lerch, "Automatic sample detection in polyphonic music," in *ISMIR*, 2017.

[7] M. Müller, *Fundamentals of Music Processing*, 2nd ed. Springer, Cham, 2021, ch. "Audio Matching", pp. 377–390.

[8] D. P. Ellis and G. E. Poliner, "Identifying 'cover songs' with chroma features and dynamic programming beat tracking," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 4, 2007, pp. IV–1429–IV–1432.