# EFFECTIVE COVER SONG IDENTIFICATION BASED ON SKIPPING BIGRAMS

*Xiaoshuo Xu* *Xiaoou Chen* *Deshun Yang*

Institute of Computer Science & Technology, Peking University
128 Zhongguancun North Street, Haidian District, Beijing 100871, P.R.China

## ABSTRACT

So far, few cover song identification systems that utilize index techniques achieve great success. In this paper, we propose a novel approach based on skipping bigrams that could be used for effective index. By applying Vector Quantization, our algorithm encodes signals into code sequences. Then, the bigram histograms of code sequences are used to represent the original recordings and measure their similarities. Through Vector Quantization and skipping bigrams, our model shows great robustness against speed and structure variations in cover songs. Experimental results demonstrate that our model achieves better performance than recent methods and is less computationally demanding.

***Index Terms***— Cover song identification, skipping bigrams, Vector Quantization, inverted index

## 1. INTRODUCTION

Cover song identification is defined as that given a query, the system retrieves all cover renditions that belong to the same song. Audio fingerprinting [1], a technique retrieving an exact copy of a query, cannot be directly applied to this task since great variations such as timbre, tempo and tonality often exist on cover songs.

So far, widespread approaches like Dynamic Programming (DP) or Dynamic Time Warp (DTW) find the optimal matches between queries and cover versions in the database. For example, Ellis and Poliner [2] used DP to extract Beat-chroma features and cross-correlated the representation for sharp peaks indicating local optimal matches. Foote [3] applied DTW to spectral energy features to quantify the similarity between two music pieces. Gómez and Herrera [4] proposed a DTW approach using chroma features. Similarly, Serrà *et al.* [5] utilized $Q_{max}$, a sophisticated DP method, for cover song identification and won Mirex Audio Cover Song Identification contests from 2007 to 2009.

Indeed, these approaches achieve high accuracies when applied to small-scale databases. However, they become time-consuming on large-scale databases. Given a dataset with $N$ recordings with a mean length of $M$, they require the computation of $O(NM^2)$ for each query. To address such difficulty, one intuitive solution is to reduce the complexity of comparison $O(M^2)$ and compensate for time used for iteration. Serrà et al. [6] modeled cover songs with temporal model. Bertin-Mahieux and Ellis [7] applied two-dimension Fourier transform to Beat-chroma features and obtained low-dimensional vectors for calculating similarities. Humphrey et al. [8] projected 2D Fourier Magnitude into very sparse space and applied supervised learning to cover song retrieval. Foster et al. [9] considered an information-theoretic approach to measure the similarity among tracks. These works reduce $O(NM^2)$ to $O(NM)$, but they still have to iterate the whole music collection.

Some scholars attempt to avoid exhaustive iteration through index techniques. In [10], a hash index was constructed to perform efficient retrieval in classical music databases. Casey et al. [11] split songs into chunks, and Local Sensitive Hash (LSH) was built to speed up recognition in remix music set. Yu et al. [12] employed a two-level LSH on a multi-variant audio database. Indeed, these works show enormous potential of index techniques in music retrieval. However, these systems aim at small variations in signals and cannot directly be applied to cover song identification. In [13], the authors utilized hashed landmarks to cover song identification but did not achieve good results. To the best of our knowledge, there are still few successful index approaches used for cover song identification until now. More sophisticated and specific retrieval systems should be studied and adapted to this task.

In this paper, we propose a new retrieval approach based on skipping bigrams robust against musical variations. When retrieving cover songs, our model adopts inverted index for acceleration. With skipping bigrams and an effective index, our approach achieves good performances with relatively low time complexity. Our work extends traditional music retrieval framework into cover song identification. Originally, [14] proposed a N-gram model for folk music retrieval. In our work, we utilize skipping bigrams and Vector Quantization to reduce effects of speed and structure variations in cover songs. Indeed, Vector Quantization and clustering methods were well explored in Music Information Retrieval [15, 16], but these works did not focus on cover song identification. Even though Vector Quantization is a classical technique, we surprisingly find that it shows great robustness in this task.

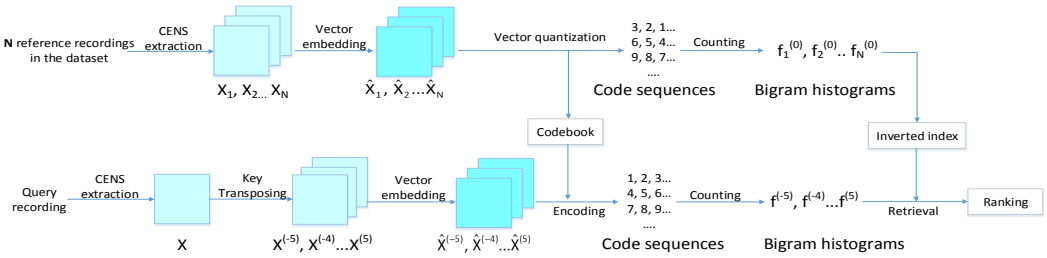


**Fig. 1**. Overview of the approach. There are $N$ recordings in the dataset, and 11 transposed CENS sequences are generated for each query.

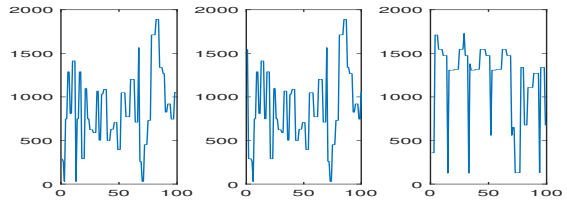

**Fig. 2**. Visualization of code sequences for two versions of "Dancing Queen" and one version of "Rolling in the Deep". We regard the codes as numbers and draw the curve through time.

## 2. APPROACH

As shown in Figure 1, chroma is extracted from audio. Then, chroma sequences are embedded and transformed into code sequences, which are counted to establish bigram histograms and measure the similarity. Besides, inverted hashes are utilized for acceleration.

### 2.1. Feature extraction

Chroma Energy Normalized Statistics (CENS), a kind of enhanced chroma feature, represents the intensity of twelve pitch classes of music [17]. It shows promising applications in recent research [18, 19]. In our case, we use Chroma Toolbox [20] to extract CENS. Under the default setting, we obtain a sequence of 12-dimensional vectors—2 vectors per second, each vector corresponding to 2100ms.

In CENS descriptor, circular shifts are used to represent key transpositions, which are common changes in cover songs. Given a chroma vector $\mathbf{x} = (x_0, x_1 ... x_{11})^T$, the transposed vector is defined as follows:

$$\mathbf{x}^{(i)} = (x_{i\%12}, x_{(i+1)\%12} \dots x_{(i+11)\%12})^T \quad (1)$$

where $i$ denotes the circular shift and $\%$ represents the modulus operator. In our experiments, we consider key transposition within five semitones, i.e. $i \in \{-5, -4 \dots 0 \dots 4, 5\}$.

### 2.2. Vector embedding

We extract a CENS sequence from each recording and use a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2 ... \mathbf{x}_M]$ to represent the sequence. Similarly, the transposed sequence $\mathbf{X}^{(i)} = \left[\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)} ... \mathbf{x}_M^{(i)}\right]$ is constructed from the transposed vectors. Then, similar technique in [5] is utilized to exploit more temporal information. This technique incorporates chroma vectors into high-dimensional vectors. The embedded vector is written as follows:

$$\hat{\mathbf{x}}_j = \left[\mathbf{x}_j^T, \mathbf{x}_{j-1}^T \dots \mathbf{x}_{j-(m-1)}^T\right]^T, j = m, m+1 \dots M \quad (2)$$

where $m$ is the embedded dimension and set to be 19 in the experiments. These embedded vectors are combined to yield an embedded sequence:

$$\hat{\mathbf{X}} = [\hat{\mathbf{x}}_m, \hat{\mathbf{x}}_{m+1}, \hat{\mathbf{x}}_{m+2} \dots \hat{\mathbf{x}}_M] \quad (3)$$

Note $\mathbf{X}$ has a shape of $12 \times M$, and $\hat{\mathbf{X}}$ has a shape of $12m \times (M - m + 1)$. Similarly, we can construct an embedded sequence $\hat{\mathbf{X}}^{(i)}$ from $\mathbf{X}^{(i)}$.

### 2.3. Vector quantization and encoding

We extract embedded sequences $\hat{\mathbf{X}}_1, \hat{\mathbf{X}}_2 \dots \hat{\mathbf{X}}_N$ from reference recordings in the dataset, and Vector Quantization is used to cluster embedded vectors. It results in code sequences and a codebook for encoding. The codewords represent quantization vector centroids numbered by $\{0, 1, 2 ... K-1\}$ ($K$ is the scale of codebook). As for the query, we transpose its CENS sequences first, and embedded sequences are encoded by the codebook to generate code sequences. Transposing query sequences instead of reference sequences, we do not need to store the transposed sequences for the dataset and thus can lower the storage.

By transforming into code sequences, structural changes in music are eliminated partly since similar embedded vectors might be clustered into the same group. Even though great differences often exist in cover songs, we empirically find that code sequences of cover songs reveal high similarity, while code sequences of different songs show little similarity. An example is displayed in Figure 2.

Furthermore, converting feature sequences into code sequences shows advantages of high efficiency. Many cover song identification methods involve steps to compute some distances like the Euclidean distance between two feature sequences. Representing music with code sequence avoids this sort of computation. Of course, encoding needs extra computation, but it could be dramatically reduced by Nearest Neighbor Searching algorithm. In addition, as the set of codes is finite, it is convenient to build an index for further processing.

### 2.4. Bigram histogram and similarity

After converting audio into code sequences, we represent the recording with its bigram histogram (note that we only count the number of each bigram but not normalize it). When constructing bigrams, we consider not only adjacent codewords but also skipping bigrams with a distance $s$ to reduce the influence of local variations. An example could demonstrate its merits. Given $\{1, 2, 3\}$ and $\{1, 3\}$, measuring the similarity of bigram histogram ends up with no likeness as they do not share any bigrams. However, given the skipping gap $s = 2$, we can construct skipping bigrams as $\{(1, 2), (1, 3), (2, 3)\}$ and $\{(1, 3)\}$ for the two sequences. Indeed, the two sequences share the same pair $(1, 3)$, and they show some kind of similarity.

Then, given two code sequences, their similarity is defined as the overlap of their bigram histograms. Given a query song $u$ and a reference song $v$, we generate 11 code sequences from $u$ in consideration of key transposing. The similarities between these sequences and $v$'s code sequence are computed, and the maximum similarity is defined as the similarity between $u$ and $v$:

$$S(u, v) = \max_{i \in \{-5, -4 \ldots 5\}} \sum_{a,b} \min\{f_u^{(i)}(a, b), f_v^{(0)}(a, b)\} \quad (4)$$

where $f^{(i)}$ represents the histogram of skipping bigrams in the code sequence, $i$ indicates tonal transposition, and $(a, b)$ denotes a specific skipping bigram. Moreover, as the histogram $f^{(i)}$ is often sparse, we are inspired to adopt inverted index for acceleration.

### 2.5. Inverted index

The inverted index is stored in a table which contains mappings from specific bigrams to the recordings that own these bigrams. Each row of the table has a key and a list containing the identifiers of recordings and the corresponding bigram frequencies (see Figure 3). Besides, a hash function is designed to generate key values from bigrams. Given a bigram $(a, b)$, its key value is defined as:

$$h\,(a,\ b) = 2^B a + b \quad (5)$$

where $B$ represents the shift width, and $a, b, B$ satisfy that $0 \leq a, b < K$ and $B = \lceil \log_2 K \rceil$($K$ is the scale of codebook). This function shifts $a$ left by $B$ bits and adds $b$ to it.

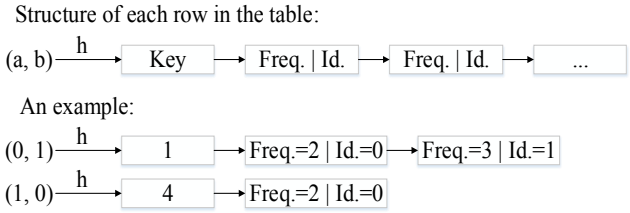


**Fig. 3**. Structure of the table. Given $B = 2$, an example is provided to demonstrate the structure of the table. Consider two songs with the identifier of 0 and 1. The first song has bigrams (0, 1) and (1, 0) with the frequency of 2. The second song has a bigram (0, 1) with the frequency of 3.

Obviously, it is a one-to-one function and maps distinct bigrams to a set of integers with no collisions. Given a pair $(a, b)$, the table helps search the recordings that contain this pair and the corresponding frequencies. In other words, we could find $\{(v, f_v^{(0)}(a, b))|f_v^{(0)}(a, b) > 0\}$ quickly.

### 2.6. Retrieval

Given a query $u$, our model first generates code sequences through transposing and embedding. Then, the bigram histogram $f_u^{(i)}$ is computed. Fixed $i$, for each pair $(a, b) \in \{(a, b)|f_u^{(i)}(a, b) > 0\}$, we find $\{(v, f_v^{(0)}(a, b))|f_v^{(0)}(a, b) > 0\}$ through the table. Then, the similarities between $u$'s code sequences and the reference code sequences can be calculated. Finally, enumerating $i \in \{-5, -4 \ldots 5\}$, the algorithm computes the similarity between the query and the reference and returns a ranking list.

## 3. PREPARATION

In our experiment, the mean average precision (MAP), the precision at 10 (P@10) and the mean rank of first correctly identified cover (MR1) are applied to assess the performances. We use *MC* and *Youtube* to evaluate our approach.

We build up an in-house database called *MC*, consisting of 2475 music recordings and 165 cliques. This collection mainly contains Chinese, Cantonese, Japanese and Korean popular music. This database is used to tune the hyperparameters $s, K$. *Youtube* contains 50 compositions, with 7 recordings of each composition. Following the setting of [18, 19], we compare our approach with recent methods on this database.

## 4. EXPERIMENTS

### 4.1. Influence of hyperparameters

Fixing $K = 1024$, we resample CENS sequences with different rate $r$ to simulate different speed. The experiment is

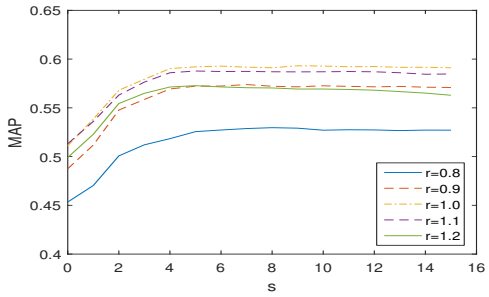


**Fig. 4**. Influence of s on *MC-150*. Note that $s = 1$ represents a bigram model, and for $s = 0$ the model degrades to a monogram model.

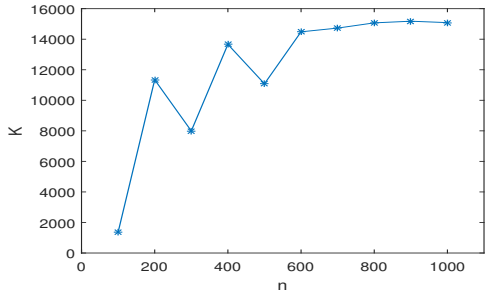


**Fig. 5**. Curve of the optimal $K$ and the extent of database $N$

conducted on *MC-150*, a subset generated from *MC* containing 150 songs (each song has two versions; thus it has 300 recordings totally). Figure 4 shows that under different speed $r$, increasing $s$ improves the performance until $s$ goes higher than 5. By exploiting skipping bigrams, the algorithm outperforms monogram and bigram models, showing robustness against speed variations. Besides, note that constantly increasing $s$ does not weaken the performance. It is due to the sparseness of bigram histograms; even if more skipping bigrams are constructed, these redundant pairs do not affect the calculation of similarity.

Different from $m$, $K$ possibly depends on the size of the database. It is consistent with an intuitive thought: the more recordings we have, the more codes we need to encode the recordings. Importantly, we care about how many codes we need to ensure good performance. We assume that given a database with $N$ recordings, a $K$ corresponding to the highest MAP is defined as the optimal $K$. To explore the relationship of these parameters, we generate subsets from *MC* with different scales and run our model to find the optimal $K$. Then, the optimal $K$ corresponding to $N$ is drawn and the curve is plotted in Figure 5. Interestingly, Figure 5 demonstrates that sub-linear relationship exists between $N$ and $K$. This result is inspiring since given a dataset with $N$ recordings, we expect a small $K$ to reduce the encoding time.

| | MAP | P@10 | MR1 | Time/s | Complexity |
|---|---|---|---|---|---|
| DTW [19] | 0.425 | 0.114 | 11.69 | 56.50 | $O(NM^2)$ |
| Silva et al. [19] | 0.478 | 0.126 | 8.49 | 3.71 | $O(NMS)$ |
| Serra et al. [21] | 0.525 | 0.132 | 9.43 | 2419.20 | $O(NM^2)$ |
| Silva et al. [18] | 0.591 | 0.140 | 7.91 | 18.72 | $O(NM \log M)$ |
| Rafii CQT [22] | 0.521 | 0.122 | 9.75 | - | $O(NM^2)$ |
| Rafii fingerprint [22] | **0.648** | 0.145 | 8.27 | - | $O(NM^2)$ |
| Skipping bigrams | 0.617 | **0.147** | **7.42** | **3.40** | $O(M \log K)$ |

**Table 1**. Performances of different approaches. Note these baselines are reported on the original papers, [22] does not provide the running time, and $S$ is far smaller than $M$.

### 4.2. Comparision

Based on the analysis above, we employ our approach on *Youtube* with $s = 9, K = 2048$ and run our model for ten times. As the deviation is small, we only report the mean in Table 1. Experimental results show that our approach achieves the highest P@10 and MR1. We only obtain CENS features used in [19, 18, 21][1], and it is fair to compare the performance of these systems. However, the state-of-the-art method [22] extracts sophisticated fingerprints from raw audio and achieves the highest MAP. It is possible that our model could achieve better results by using these features.

As for query time, our approach runs fast and spends 3.40 seconds on a query in average. Note that these methods are implemented in different environmental settings, and thus it is hard to compare their efficiency. In our case, we use a Dell PowerEdge R730 server with an Intel Xeon E5-2640v3 processor and implement the approach in Python and C++[2]. A thorough comparison of the efficiency is to consider their time complexity.

Through using Nearest Neighbor Searching algorithm, it takes $O(M \log K)$ to convert music to code sequences. By exploiting inverted index, the matching process is highly effective, far less time-consuming than the encoding procedure. Overall, the complexity of query time becomes $O(M \log K)$, which avoids exhaustive search and is more efficient than other approaches. Of course, constructing the codebook and the inverted index requires extra computation, but these processes run for one time and do not affect query time.

## 5. CONCLUSION

We have presented an approach based on skipping bigrams for cover song identification. By converting audio into code sequences and applying skipping bigrams, our approach shows robustness against speed and structure variations. Besides, we design an inverted index for fast retrieval. Our approach achieves good performances with low time cost on a recent cover song dataset. In future, we will use other methods to encode raw audio and apply our approach to large-scale databases.

[1] https://sites.google.com/site/ismir2015shapelets/data
[2] https://github.com/NovaFrost/skipping

## 6. REFERENCES


[1] Rainer Typke, Frans Wiering, Remco C Veltkamp, et al., "A survey of music information retrieval systems.," in *International Society for Music Information Retrieval Conference*, 2005, pp. 153–160.

[2] Daniel PW Ellis and Graham E Poliner, "Identifying-cover songs' with chroma features and dynamic programming beat tracking," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007, vol. 4, pp. IV–1429.

[3] Jonathan Foote, "Arthur: Retrieving orchestral music by long-term structure.," in *International Society for Music Information Retrieval Conference*, 2000.

[4] Emilia Gómez and Perfecto Herrera, "The song remains the same: identifying versions of the same piece using tonal descriptors," in *International Society for Music Information Retrieval Conference*, 2006, pp. 180–185.

[5] Joan Serra, Xavier Serra, and Ralph G Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, no. 9, pp. 093017, 2009.

[6] Joan Serra, Holger Kantz, Xavier Serra, and Ralph G Andrzejak, "Predictability of music descriptor time series and its application to cover song detection," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 514–525, 2012.

[7] Thierry Bertin-Mahieux and Daniel PW Ellis, "Large-scale cover song recognition using the 2d fourier transform magnitude," in *International Society for Music Information Retrieval Conference*, 2012, pp. 241–246.

[8] Eric J Humphrey, Oriol Nieto, and Juan Pablo Bello, "Data driven and discriminative projections for large-scale cover song identification.," in *International Society for Music Information Retrieval Conference*, 2013, pp. 149–154.

[9] Peter Foster, Simon Dixon, and Anssi Klapuri, "Identifying cover songs using information-theoretic measures of similarity," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 6, pp. 993–1005, 2015.

[10] Peter Grosche and Meinard Müller, "Toward characteristic audio shingles for efficient cross-version music retrieval," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 473–476.

[11] Michael Casey and Malcolm Slaney, "Fast recognition of remixed music audio," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2007, vol. 4, pp. IV–1425.

[12] Yi Yu, Michel Crucianu, Vincent Oria, and Lei Chen, "Local summarization and multi-level lsh for retrieving multi-variant audio tracks," in *Proc. of the 17th ACM international conference on Multimedia*, 2009, pp. 341–350.

[13] Thierry Bertin-Mahieux and Daniel PW Ellis, "Large-scale cover song recognition using hashed chroma landmarks," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2011, pp. 117–120.

[14] Stephen Downie and Michael Nelson, "Evaluation of a simple and effective music information retrieval method," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2000, pp. 73–80.

[15] Mark Levy and Mark Sandler, "Music information retrieval using social tags and audio," *IEEE Transactions on Multimedia*, vol. 11, no. 3, pp. 383–395, 2009.

[16] Frank Kurth and Meinard Muller, "Efficient index-based audio matching," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 382–395, 2008.

[17] Meinard Müller, *Information Retrieval for Music and Motion*, Springer Verlag, 2007.

[18] Diego F Silva, Chin-Chin M Yeh, Gustavo Enrique de Almeida Prado Alves Batista, Eamonn Keogh, et al., "Simple: assessing music similarity using subsequences joins," in *International Society for Music Information Retrieval Conference*, 2016.

[19] Diego Furtado Silva, Vinícius Mourão Alves de Souza, Gustavo Enrique de Almeida Prado Alves Batista, et al., "Music shapelets for fast cover song regognition," in *International Society for Music Information Retrieval Conference*, 2015.

[20] Meinard Müller and Sebastian Ewert, "Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features," in *International Society for Music Information Retrieval Conference*, Miami, USA, 2011.

[21] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1138–1151, 2008.

[22] Prem Seetharaman and Zafar Rafii, "Cover song identification with 2d fourier transform sequences," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 616–620.